Semester thesis: Understanding the PPSZ algorithm for ClSP

Isabelle Hurbain

Spring Semester 2013

Contents

1 Basic concepts and notation **5**

2 PPSZ for 3-SAT **7**

2.1 The PPSZ algorithm for 3-SAT . . . 7
2.2 Unique satisfying assignment . . . 8
2.2.1 Forced and guessed variables . . . 9
2.2.2 What does it mean for a variable to be forced? . . . 9
2.2.3 Building critical clause trees . . . 11
2.2.4 Placements, critical clause trees and forced variables . . . 12
2.2.5 Random deletion in infinite binary trees . . . 14
2.2.6 From infinite to finite trees . . . 15
2.2.7 From independent to dependent labels . . . 16
2.2.8 Integrating over the place of the root . . . 17
2.2.9 Summary . . . 19
2.3 Multiple satisfying assignments . . . 19
2.3.1 Problem assessment . . . 20
2.3.2 Proof idea . . . 20
2.3.3 Definition of a cost function . . . 21
2.3.4 Proving Lemma 2.19 . . . 24
2.4 Summary . . . 30

3 A weak version of PPSZ for (d, k)-ClSP **32**

3.1 The algorithm . . . 32
3.2 Unique satisfying assignment . . . 33
3.2.1 Forced and guessed variables . . . 34
3.2.2 Building critical clause trees . . . 34
3.2.3 Critical clause trees and forced variables . . . 35
3.2.4 Random deletion in infinite and finite trees of degree $(d-1)(k-1)$ . . . 37
3.2.5 From independent to dependent labels . . . 38
3.2.6 Integrating over the rank of the root . . . 39
3.3 Multiple satisfying assignments . . . 45
3.3.1 Definition of a cost function . . . 45
3.3.2 Proving Lemma 3.19 . . . 46
3.3.3 Using the induction hypothesis . . . 48
3.4 Summary . . . 55

4 A strong version of PPSZ for (d,k)-ClSP **57**
4.1 The algorithm . . . 57
4.2 Unique satisfying assignment . . . 58
4.2.1 The sets $\mathcal{A}(x,\pi,D)$ . . . 58
4.2.2 Building critical clause trees . . . 59
4.2.3 Critical trees and $\mathcal{A}(x,\pi,D)$ . . . 60
4.2.4 Random deletion in one infinite tree . . . 61
4.2.5 From one tree to $(d-1)$ trees . . . 62
4.2.6 From infinite to finite trees . . . 63
4.2.7 From independent to dependent labels . . . 65
4.2.8 Integrating over the rank of the root . . . 67
4.2.9 Summary . . . 70
4.3 Multiple satisfying assignments . . . 70
4.3.1 Definition of a cost function . . . 71
4.3.2 Towards the proof of Conjecture 4.19 . . . 73

A Glossary of classical notions **80**
A.1 Jensen's inequality . . . 80
A.2 Monotone convergence theorem . . . 80
A.3 FKG inequality . . . 81
A.4 Riemann sums approximations . . . 82

B Complementary proofs **83**
B.1 General inequalities . . . 83
B.1.1 An inequality about logarithms . . . 83
B.1.2 An inequality about power functions . . . 83
B.1.3 A power identity . . . 84
B.2 Correlation inequalities . . . 84
B.2.1 Proof of Lemma 2.24 . . . 84
B.3 Proof of Lemma 4.12 . . . 84
B.4 Proof of Lemma 4.8 . . . 85

Introduction

SAT and PPSZ

The satisfiability problem, or SAT, is the archetypical NP-complete problem. It can be formulated as follows: given a boolean formula F on a variable domain V, is there an assignment of the variables of V such that F is satisfied, i.e. it evaluates to true? In this thesis, we will consider boolean formulas in the conjunctive normal form (CNF formulas), and more precisely formulas that have clauses of size at most k, which we call $k-$SAT formulas.

The fastest known algorithm to solve $k-$SAT is called PPSZ, after its authors Paturi, Pudlák, Saks and Zane [4]. In the original paper, the best bounds for the algorithm were proved only for the case where the formula was guaranteed to have a single satisfying assignment. Hertli [2] later used a slight modification of the algorithm to show that these bounds hold in general.

The PPSZ algorithm is itself a modification of the PPZ algorithm, after Paturi, Pudlák and Zane [5]. The idea of PPZ is very simple: process the variables of the domain in a random order. For a given variable x, if there is a clause $\{x\}$ or a clause $\{\bar{x}\}$, set the variable according to that clause; otherwise, pick a value at random. The PPSZ algorithm extends this idea by picking not only unit clauses, but small subsets of clauses. If some small subset of clauses is only compatible with one given assignment of the variable, set the variable accordingly; otherwise, pick a value at random.

Clause satisfaction problems

There is a natural extension of the $k-$SAT formulas, namely what we call $k-$ClSP formulas, where ClSP stands for Clause Satisfaction Problems. Instead of considering boolean values, we consider in ClSP formulas a domain $\{1, ..., d\}$ of d values. The literals of such formulas are of the type $(x \neq c)$ where $x \in V$ and $c \in \{1, ..., d\}$. The rest of the definition stays the same. Once this problem is defined, extending PPZ and PPSZ to solving such problems, and trying to bound the running time of these algorithms on $k-$ClSP is also pretty natural.

Thesis outline

This thesis aims at giving a full, notationally cohesive state of the art of the analysis of PPSZ for $k-$ClSP formulas. In Chapter 1, we will define precisely the setup, terminology and notations used in this thesis. Chapter 2 details the proof of the achieved runtime of PPSZ for 3-SAT. It must be seen as the foundation on which further work is based. The proof outline used in the 3-SAT case will be re-used in the ClSP case. The proofs of this chapter are given in the original PPSZ paper by Paturi, Pudlák, Saks and Zane [4] and in the subsequent paper by Hertli [2]. The framework and write-up of the proof are heavily based on Welzl lecture notes [8]. Chapter 3 and

Chapter 4 are following the proof outline of Chapter 2 to give runtime bounds for two versions of PPSZ adapted to ClSP. Chapter 3 and Chapter 4 are based on the work of Szedlák [7] and Millius [3].

Acknowledgements

Many thanks to Timon Hertli, my advisor, whose advice and great attention to details were greatly appreciated. I also would like to thank Dominik Scheder, May Szedlák, and Sebastian Millius for their previous work that made this thesis at all possible. Finally, the SAT course run in 2012 by Emo Welzl and Robin Moser was truly inspiring; many thanks to them as well.

Chapter 1

Basic concepts and notation

We use the notational framework presented in [8] and used in [2], [7] and [3].

$k-$SAT

Let V be a set of boolean variables. A literal l over $x \in V$ is a variable x or a complemented variable $\bar{x}$. A clause C over V is a finite set of literals over pairwise distinct variables from V. A formula F in conjunctive normal form, or a CNF formula, is a finite set of clauses. We interpret this set of clauses by joining the literals in a clause by a logical OR and by joining the clauses by a logical AND. A $k-$SAT formula F is a CNF formula in which every clause contains at most k literals.

$(d, k)-$ClSP

We extend the above definition to define k-ClSP formulas. Let us denote by $[d] = \{1, ..., d\}$ the set of integers from 1 to d. Let V be a set of variables that take their values in $[d]$. A literal l over $x \in V$ is of the form $(x \neq c)$ for $c \in \{1, ..., d\}$. A clause C over V is a finite set of literals over variables from V. A clause satisfaction problem formula or ClSP formula is a finite set of clauses. We interpret this set of clauses by joining the literals in a clause by a logical OR and by joining the clauses by a logical AND. A $(d, k)-$ClSP formula is a ClSP formula over the domain $[d]$ where every clause contains at at most k literals.

Observe that this definition is equivalent to the SAT definition for $d = 2$, and that a (2, 3)$-$ClSP formula is a 3-SAT formula. In what follows, we will consequently give the definitions in the more general $k-$ClSP setup; they can easily be applied to the standard SAT framework.

Variable sets and assignments

Let $V(C)$ be the set of variables in a constraint C and $V(F)$ be the set of variables in F, i.e. the union of the sets of variables of each constraint. Let $n(F) = |V(F)|$ be the number of variables. We sometimes write n and V instead of $n(F)$ and $V(F)$ if it is clear from the context which formula F we are referring to.

An assignment α is defined as a function $\alpha : V(F) \to [d]$ assigning to each variable an element of the domain. A partial assignment on a set V is a function which maps a subset $W \subseteq V$ to the domain $[d]$. For a ClSP F and a partial assignment α on $V(F)$, we denote by $F^{[\alpha]}$ the restriction of F to α, that is the ClSP that is obtained by replacing the variables in the domain of α by their assigned values. We denote a value assignment l of a variable x to a value c as $l = x \mapsto c$. For a value assignment l we denote by $\mathrm{vbl}(l)$ the variable which l assigns a value to. We can interpret l as a partial assignment on $V(F)$. Similarly, we interpret a partial assignment α on $V(F)$ as a set of value assignments $\{x \mapsto c\}$. We denote, for $l : x \mapsto c$, $\alpha(x) = c$.

Two assignments α and α' are compatible if, for every variable x that is assigned in both α and α', $\alpha(x) = \alpha'(x)$. We can define the union of two compatible assignments $\alpha \cup \alpha'$ as the union of the set of assignments of both assignments. If one of the assignments is a single value assignment $l : x \mapsto c$, we will use the shorthand $\alpha[l]$ to denote $\alpha \cup \{x \mapsto c\}$ or $\alpha \cup \{l\}$.

An assignment α on V is called satisfying if $F^{[\alpha]}$ interpreted as a logical formula in conjunctive normal form evaluates to true. By $\mathrm{sat}(F)$, we denote the set of all satisfying assignments on $V(F)$. A formula F is called satisfiable if $\mathrm{sat}(F) \neq \emptyset$ and unsatisfiable otherwise.

Implication and D-implication

Let F be a ClSP formula, let $D \in \mathbb{N}_0$, and let u be a literal. We say that F implies u, in writing $F \vDash u$, if any assignment α satisfying F also satisfies u. We say that F D-implies u, in writing $F \vDash u$, if there is some $G \subseteq F$ with $|G| \leq D$ having $G \vDash u$. We say that F does not imply (respectively does not D-imply) u, in writing $F \nvDash u$ (respectively $F \nvDash_D u$) if there exists an assignment α satisfying F that does not satisfy u (respectively if no $G \subseteq F$ with $|G| \leq D$ is such that $G \nvDash u$).

Let x be a variable; we say that F does not imply (respectively does not D-imply) x, in writing $F \nvDash x$ (respectively does not D-imply x), in writing $F \nvDash x$ (respectively $F \nvDash x$) if it is not true that there exists $c \in [d]$ such that, for all $c' \neq c$, $F \vDash (x \neq c')$ (respectively $F \vDash_D (x \neq c')$).

Chapter 2

PPSZ for 3-SAT

Before we start studying PPSZ in the ClSP case, we explain its analysis for the (simpler) 3-SAT case. 3-SAT can be seen as a (2, 3)-ClSP; it is hence a restriction of the general (d,k)-ClSP problem. This chapter is heavily based on [8].

2.1 The PPSZ algorithm for 3-SAT

Consider the following informal algorithm. Given a CNF formula F, choose a random variable $x \in \text{vbl}(F)$. If the clause $\{x\}$ or the clause $\{\bar{x}\}$ appears in F, set x so that it satisfies that unit clause. Otherwise, set x randomly. Then iterate.

This randomized algorithm is due to Paturi, Pudlák and Zane [5] and is named PPZ after its authors. We are interested in a generalized version of this algorithm, namely PPSZ (after its authors Paturi, Pudlák, Saks and Zane [4]). The idea of PPSZ is that we are not necessarily looking at a *direct* way to deduce the value of the variable that we treat, but we're allowing to deduce that value from several clauses. An easy example of that would be that, for the formula $\{x, y, z\}, \{x, \bar{y}, \bar{z}\}, \{x, \bar{y}, z\}, \{x, y, \bar{z}\}$, then the value of x is forced to 1 because we cannot satisfy the formula for $x = 0$.

More formally, we define the concept of *D-implication* for SAT as follows:

Definition 2.1. [8] Let F be a CNF formula, let $D \in \mathbb{N}_0$, and let u be a literal. We say that F implies u, in writing $F \vDash u$, if any assignment α satisfying F also satisfies u. We say that F D-implies u, in writing $F \vDash_D u$, if there is some $G \subseteq F$ with $|G| \leq D$ having $G \vDash u$.

Let x be a variable; we say that F does not imply (respectively does not D-imply) x, in writing $F \nvDash x$ (respectively $F \nvDash_D x$) if it is not true that $F \vDash (x = 1)$ or $F \vDash (x = 0)$ (respectively $F \vDash_D (x = 1)$ or $F \vDash_D (x = 0)$).

Checking whether the clause $\{x\}$ or $\{\bar{x}\}$ appears in the formula F (as we do in the PPZ algorithm) is checking for 1-implication.

The PPSZ algorithm can then be stated informally as follows. Given a CNF formula F, choose a random variable $x \in \text{vbl}(F)$. If $F \vDash_D x$, set x to 1; if $F \vDash_D \bar{x}$, set x to 0; otherwise, set x randomly. Then iterate.

To state it formally, we model the choice of the random variable and the choice of the random value as, respectively, a random permutation π and a random assignment β over the variables of the formula.

We also introduce here a parameter α_0 that represents a partial assignment to the variables of V. This isn't needed by the algorithm per se; we would run the algorithm with $\alpha_0 = \emptyset$ to try to get a satisfying assignment of a formula F. However, it simplifies the notation for some parts of the analysis. We also define, again for ease of notation, $\mathcal{U}(\alpha_0) = V \backslash \text{vbl}(\alpha_0)$ and $n(\alpha_0) = |\mathcal{U}(\alpha_0)|$. If α is a total assignment, we say that α and α_0 are *incompatible* if $\exists x \in V : \{\alpha(x), \alpha_0(x)\} = \{0, 1\}$ and *compatible* otherwise. This definition was given for the general case in Chapter 1; here we simply instantiate it for the SAT case.

With these notations defined, we state the algorithm as follows:

```
PPSZ($F, V, \alpha_0, D$)
    $\pi \leftarrow$ a permutation of $\mathcal{U}(\alpha_0)$ chosen u.a.r.;
    $\beta \leftarrow$ an assignment from $\{0,1\}^{\mathcal{U}(\alpha_0)}$ chosen u.a.r.;
    return PPSZ($F, V, \alpha_0, D, \pi, \beta$);

PPSZ($F, V, \alpha_0, D, \pi, \beta$)
    $\alpha_{\text{prog}} \leftarrow \alpha_0$
    for $i \leftarrow 1$ to $n(\alpha_0)$
        do
            $x \leftarrow x_{\pi_i}$
            if $F^{[\alpha_{\text{prog}}]} \vDash_D x$
              then $\alpha_{\text{prog}}(x) \leftarrow 1$
              else if $F^{[\alpha_{\text{prog}}]} \vDash_D \bar{x}$
                     then $\alpha_{\text{prog}}(x) \leftarrow 0$
              else $\alpha_{\text{prog}}(x) \leftarrow \beta(x)$
    if $\alpha_{\text{prog}}$ satisfies $F$
      then return $\alpha_{\text{prog}}$
      else return 'failure'
```

Observe that PPSZ always returns "failure" if the formula is unsatisfiable. Our ultimate goal for this chapter is to prove the following theorem:

Theorem 2.2. For any satisfiable (≤ 3)-CNF formula F on n variables V, PPSZ$(F, V, \emptyset, \log n)$ returns some satisfying assignment with probability $\Omega(1.3071^{-n})$.

This readily implies the following:

Corollary 2.3. There exists a randomized algorithm for 3-SAT with one-sided error that runs in time $\mathcal{O}(1.3071^n)$.

The proof of Theorem 2.2 is quite involved. We will hence, as a warm-up, consider the case of a (≤ 3)-CNF formula F that has exactly one satisfying assignment, and prove, in the first part of this chapter, the following, weaker, theorem:

Theorem 2.4. For any (≤ 3)-CNF formula F on n variables V which has a unique satisfying assignment, PPSZ$(F, V, \emptyset, \log n)$ returns this assignment with probability $\Omega(1.3071^{-n})$.

2.2 Unique satisfying assignment

In this section, we consider that the formula F that we want to satisfy has a unique satisfying assignment α^*. Without loss of generality, we suppose that this satisfying assignment is the all-one assignment, i.e. $\alpha^* = (1, 1, ..., 1)$. We want to prove that, given this formula F, PPSZ$(F, \emptyset, V, \log n)$ does indeed return the all-one assignment with a probability at least $\Omega(1.3071^{-n})$.

In this section, we can also assume that $\alpha_0 = \emptyset$, $\mathcal{U}(\alpha_0) = V$ and $n(\alpha_0) = n$.

2.2.1 Forced and guessed variables

We define the concepts of *forced* and *guessed* variables as follows.

Definition 2.5. [8] Let F be a CNF formula over n variables, $\alpha^* \in \{0,1\}^n$ a satisfying assignment, α_0 a partial assignment that is compatible with α^*, $\pi = (x_1, ..., x_{n(\alpha_0)})$ a permutation of $\mathcal{U}(\alpha_0)$ and $D \geq 0$. A variable x_i is called *forced* with respect to F, α_0, α^*, π and D if $F^{[\alpha_0 \cup \{x_1 \mapsto \alpha^*(x_1), x_2 \mapsto \alpha^*(x_2), ..., x_{i-1} \mapsto \alpha^*(x_{i-1})\}]}$ D-implies x_i or $\bar{x}_i$. Otherwise, the variable is called *guessed.* We denote the set of forced (guessed) variables within $\mathcal{U}(\alpha_0)$ by $\text{Forced}(F, \alpha_0, \alpha^*, \pi, D)$ $(\text{Guessed}(F, \alpha_0, \alpha^*, \pi, D))$.

Observe that $\text{PPSZ}(F, V, \alpha_0, D, \pi, \beta)$ returns the satisfying assignment α^* if and only if $\beta(x) = \alpha^*(x)$ for all $x \in \text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)$.

From this observation, we can write

$$\Pr_{\beta,\pi}[\text{PPSZ returns } \alpha^*] = \mathop{\mathbb{E}}_{\pi}\left[2^{-|\text{Guessed}(F,\alpha_0,\alpha^*,\pi,D)|}\right]$$

We can apply Jensen's inequality (see Appendix A.1) with the convex function $x \mapsto 2^{-x}$ and obtain

$$\Pr_{\beta,\pi}[\text{PPSZ returns } \alpha^*] \geq 2^{-\mathbb{E}_\pi[|\text{Guessed}(F,\alpha_0,\alpha^*,\pi,D)|]} \tag{2.1}$$

By linearity of expectation, we can write that

$$\mathop{\mathbb{E}}_{\pi}[|\text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)|] = \sum_{x \in \mathcal{U}(\alpha_0)} \Pr_{\pi}[x \in \text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)] \tag{2.2}$$

$$= n(\alpha_0) - \sum_{x \in \mathcal{U}(\alpha_0)} \Pr_{\pi}[x \in \text{Forced}(F, \alpha_0, \alpha^*, \pi, D)] \tag{2.3}$$

Most of the work of this section will be devoted to bound these individual probabilities, i.e, for any $x \in \mathcal{U}(\alpha_0), \Pr_\pi[x \in \text{Forced}(F, \alpha_0, \alpha^*, \pi, D)]$.

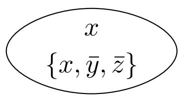
2.2.2 What does it mean for a variable to be forced?

This section aims at giving some intuition of the tool that we will be using to prove the bounds on the probabilities in equation (2.3), namely *critical clause trees.* We will first develop a small example before giving the full formalism of these trees.

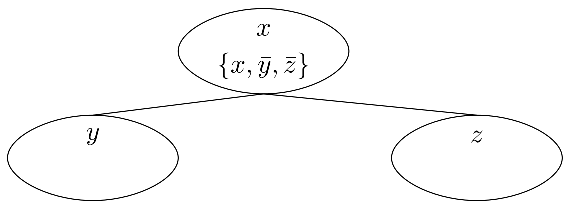
Consider the formula $F = \{\{x, \bar{y}, \bar{z}\}, \{x, \bar{v}, \bar{w}\}, \{x, y, v\}, \{x, z, \bar{a}\}, ...\}$ and the unique satisfying assignment $\alpha^* = (1, 1, ..., 1)$. Suppose, also, that we execute PPSZ with a permutation π that considers the variables in this (relative) order: (a, w, x, z, y, v). Suppose, also, that the PPSZ algorithm hasn't made mistakes so far, and that in particular the variables a and w have been assigned the value 1. Note that if it is not the case, then PPSZ has no chance of succeeding anyway, so that case is not interesting anymore.

We now consider the status (forced or guessed) of the variable x. If the variable is forced, it means that we can disprove that the variable x has the value 0. We do this by exhibiting a series of clauses that, together, prove that the value of x must necessarily be 1. We can, for instance, imagine two people arguing about the value of x, while drawing a tree on a blackboard.

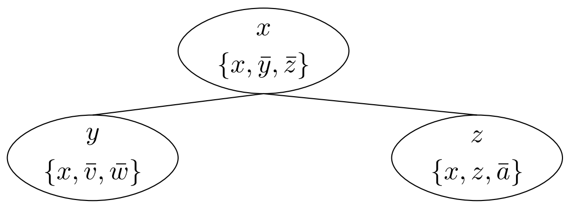
"Look, x cannot have the value 0, because otherwise the clause $\{x, \bar{y}, \bar{z}\}$ wouldn't be satisfied.



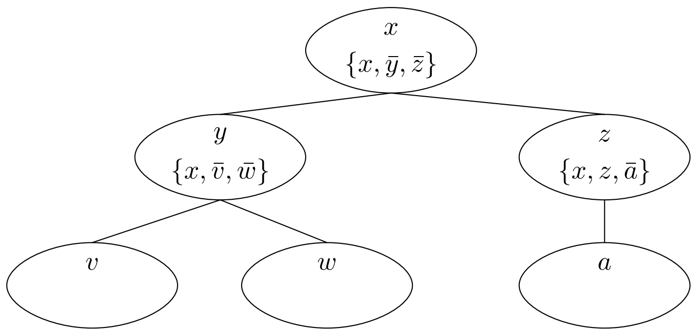
— Well, it could also be that y has the value 0, or that z has the value 0.



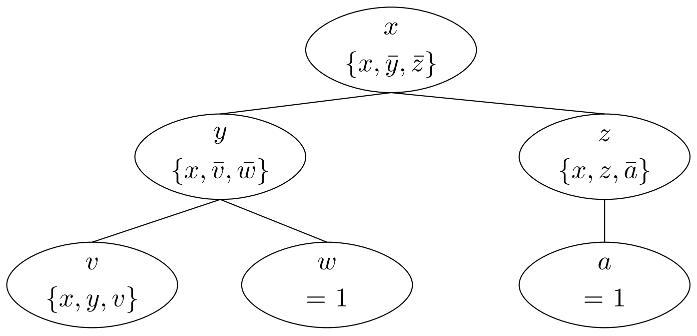
— Ok, let's admit for a moment that y can indeed have the value 0. So I need another argument there... Here it is: if $x = 0$, then $\{x, \bar{v}, \bar{w}\}$ cannot be satisfied. And as for z, well, if both x and z have the value 0, then $\{x, z, \bar{a}\}$ cannot be satisfied either.



— Well again, for the first clause that you give me, I could say that v or w can have the value 0. And for the second one, well, a can have the value 0 as well.



— We could say all this, but we would then be wrong. Let's see: we have already established that, in our run of PPSZ, a and w have the value 1. So we're stuck there: a and w cannot take the value 0. And if, as you say, x, y and v all have the value 0, then the clause $\{x, y, v\}$ cannot be satisfied.



— It looks indeed that you were right, and that x cannot have the value 0."

This reasoning shows that, by looking at 4 clauses, we can establish, given that a and w are before x in the permutation π, the value of x and hence we have that $x \in \text{Forced}(F, \alpha_0, \alpha, \pi, 4)$.

We can see that there are two ways in which we can arrive at a contradiction. We can either exhibit a clause that is incompatible with all the previous hypotheses of the tree building process (that would be the case of clause $\{x, y, v\}$ in our example), or we can have a contradiction on a variable that has already been processed (and set to the correct value) – that would be the case for the variables a and w in our example.

Suppose now that we build such trees for every variable in a formula. We want these trees to be valid for any permutation, so we continue to expand them as much as we can. Applying a given permutation to such trees will in effect make parts of the tree unnecessary, and we may as well cut them. We will show formally that, for a given permutation, if we cut the nodes labelled with the variables that are before x (and their children after that), then x is D-implied, where D is the number of remaining nodes in the tree. From there, we will transform the computation of the probability (over every permutation) that a variable x is forced into a computation about binary trees.

2.2.3 Building critical clause trees

Let's now formalize the above construction in the general case. We construct a collection of binary trees $\{T_x\}_{x \in \mathcal{U}(\alpha_0)}$, each of them called *critical clause tree of* x.

We consider a formula F that has a unique satisfying assignment. Without loss of generality, let this unique assignment be $\alpha^* = (1, 1, ..., 1)$.

T_x is a rooted binary tree, where every node $u \in V(T)$ is labelled both with a variable $x \in V$, which we denote by var-label(u), and a clause $C \in F$, denoted by clause-label(u). Here is how T_x is built for a fixed $x \in \mathcal{U}(\alpha_0)$:

1. Start with T_x consisting of a single root. This root has variable label x, and no clause label yet.
2. As long as there is a leaf $u \in V(T)$ that does not yet have a clause label, do the following:
   (a) Define $W := \{\text{var-label}(v) \mid v \in V(T) \text{ is an ancestor of } u \text{ in } T\}$, where *ancestor* includes u itself and the root.
   (b) Define the total assignment μ as
   $$\mu : \text{vbl}(F) \to \{0, 1\}, z \mapsto \begin{cases} 1 - \alpha^*(z) = 0 & \text{if } z \in W \\ \alpha^*(z) = 1 & \text{otherwise} \end{cases}$$
   (c) Let $C \in F^{[\alpha_0]}$ be a clause not satisfied by μ. Since $\mu \neq \alpha^*$ and α^* is the unique satisfying assignment, such a clause exists. Set clause-label(u) $= C$.
   (d) For each negative literal $w \in C$, create a new leaf, label it with the variable underlying w, and attach it to u as a child. The new leaf does not yet have a clause label.

We denote the resulting tree by T_x. Note that T_x is not unique for a given x. We will still consider the collection $\{T_x\}_{x \in \mathcal{U}(\alpha_0)}$ to be fixed from now on. Remember that we want to bound the expected number of nodes in these trees after doing cuts that depend on the permutation used in PPSZ, and that this number of nodes is related to the number of clauses that imply x for said permutation. Hence, it may be that another tree for T_x would give a better result, but the results we get for an arbitrary critical clause tree of x are certainly a valid bound for the number of clauses that imply x.

Note also that, since α^* satisfies F, every clause in F has at most two negative literals, and thus in step 2(d) we append at most two children, hence the tree is a binary tree. Suppose v is an ancestor of u and var-label$(v) = y$. Since $\mu(y) = 0$, there is no negative literal over y contained in clause-label(u). Therefore,

Observation 2.6. In T_x, no node has the same var-label as one of its proper ancestors.

This also implies that the height of the tree cannot exceed n, thus the process terminates, making T_x well-defined.

2.2.4 Placements, critical clause trees and forced variables

We have tried, in Section 2.2.2 to give some intuition of the connection between critical clause trees and forced variables. We have formalized the definition of said trees in Section 2.2.3; we will now use them to estimate the probability with which a variable is forced.

To simplify the calculations that await us, we introduce the notion of *placements.* A *placement* of the variables in $\mathcal{U}(\alpha_0)$[1] is a function $\pi : \mathcal{U}(\alpha_0) \to [0, 1]$. From now on, we do not think about π as a permutation anymore, but rather as a placement, which is essentially the same. If the values of $\pi(x)$ are chosen independently and uniformly at random from $[0, 1]$ for each $x \in \mathcal{U}(\alpha_0)$, then with probability 1, π is injective and, by sorting the values $\pi(x)$ we obtain a uniformly distributed permutation of $\mathcal{U}(\alpha_0)$. We call $\pi(x)$ the *place* of x.

Let $\gamma \in [0, 1]$ and T_x be the critical clause tree of some fixed variable. We call a node $u \in T_x$ *reachable at time* γ *w.r.t.* π if there exists a path $v_0, v_1, ..., v_m$ such that v_0 is the root of the tree, $v_m = u$ and $\pi(v_i) \geq \gamma$ for all $1 \leq i \leq m$. Let us denote by Reachable(T_x, γ, π) the set of all nodes in T_x reachable at time γ w.r.t. π. Observe that this set is independent of the place of x.

Lemma 2.7. If we have $|\text{Reachable}(T_x, \pi(x), \pi)| \leq D$, then it holds as well that $x \in \text{Forced}(F,$ $\alpha_0, \alpha^*, \pi, D)$.

Proof. Let α' be the restriction of $\alpha^* = (1, 1, ..., 1)$ to the variables $y \in \mathcal{U}(\alpha_0)$ with $\pi(y) < \pi(x)$. By definition, x is forced if there is a formula $F' \subseteq F^{[\alpha_0 \cup \alpha']}$ with $|F'| \leq D$ that implies x. Let $G :=$ clause-label(Reachable$(T_x, \pi(x), \pi)$), i.e. the subformula of F consisting of all the clause labels of reachable nodes in T_x. Since by hypothesis $|\text{Reachable}(T_x, \pi(x), \pi)| \leq D$, then clearly $|G| \leq D$. We claim that $G^{[\alpha_0 \cup \alpha']}$ implies x.

Suppose, for the sake of contradiction, that $G^{[\alpha_0 \cup \alpha']} \not\models x$ (recall that the unique satisfying assignment is $(1, 1, ...1)$). Then we can find an assignment $\nu : V \to \{0, 1\}$ which is compatible[2] with $\alpha_0 \cup \alpha'$, which has $\nu(x) = 0$ and which satisfies G. Choose a maximal path in T_x, starting at the root, x, and containing only nodes v such that $\nu(\text{var-label}(v)) = 0$. Since $\nu(x) = 0$, this path is non-empty. Let u be its endpoint. Since ν is compatible with α^* on all the variables before x, it must be that var-label(u) is either x itself or after x, and hence $\pi(\text{var-label}(u)) \geq \pi(x)$, and so u is reachable, by definition. For all children z of u, we have that $\nu(\text{var-label}(z)) = 1$ (because the path is maximal); all ancestors y of u are such that $\nu(\text{var-label}(y)) = 0$. By definition of T_x, clause-label(u) is unsatisfied by ν and, since clause-label$(u) \subseteq G$, this is a contradiction. $\square$

It follows immediately from Lemma 2.7 that, over the uniform choice of π, we have

$$\Pr_\pi[x \in \text{Forced}(F, \alpha_0, \alpha^*, \pi, D)] \geq \Pr_\pi[|\text{Reachable}(T_x, \pi(x), \pi)| \leq D] \tag{2.4}$$

This reduces the problem to a probabilistic calculation on binary trees: what is the probability that, when sorting the nodes of a fixed binary tree according to a random permutation (caveat:

[1]Recall that we may have a partial assignment α_0 in the call to PPSZ; we defined $\mathcal{U}(\alpha_0) = V \backslash \text{vbl}(\alpha_0)$, where V is the whole set of variables.

[2]Remember that α and α_0 are incompatible if $\exists x \in V : \{\alpha(x), \alpha_0(x)\} = \{0, 1\}$ and compatible otherwise.

some nodes have the same label and are prescribed to get assigned the same place) and deleting all nodes whose place is after the root, there will be at most D nodes reacheable?

We will, in the next section, prove the following theorem:

Theorem 2.8. For any $\varepsilon > 0$ there exists $D_\varepsilon \in \mathbb{N}$ depending only on ε such that the following holds. Let T be a finite binary tree and $\sigma : V(T) \to \{1, ..., r\}$ a labelling of the nodes of T such that on each path from the root to a leaf of T, σ is injective. Let $X_1, X_2, ...X_r$ be real random variables distributed uniformly from $[0, 1]$ and mutually independently. Consider the experiment of drawing $X_1, X_2, ..., X_r$ according to their distribution and then deleting all nodes u from T (along with the corresponding subtrees) for which $X_{\sigma(u)} < X_{\sigma(\text{root})}$.

Then the probability that the resultant tree T' contains more than D_ε nodes is

$$\Pr_{X_1,...,X_r}[|V(T')| > D_\varepsilon] \leq S + \varepsilon$$

where

$$S = \frac{1}{2} - \int_0^{\frac{1}{2}} \frac{p^2}{(1-p)^2} \mathrm{d}p = 2\ln 2 - 1 \leq 0.3863$$

Before we prove Theorem 2.8, we see how this theorem allows us to prove Theorem 2.4.

We introduce the notation

$$S_D := \sup_T \left(\Pr_{X_1,...,X_r}[|V(T')| > D)] \right)$$

where the supremum is over all choices of finite trees with labels T (as in Theorem 2.8) and T' is the random tree arising from T by conducting the experiment described in Theorem 2.8. In this language, the theorem states that

$$\lim_{D\to\infty} S_D \leq S$$

The limit exists because S_D is monotonic (it decreases as D increases) and bounded (because a probability is greater or equal than 0).

Let us recall equation (2.3):

$$\mathbb{E}_\pi[|\text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)|] = n(\alpha_0) - \sum_{x\in\mathcal{U}(\alpha_0)} \Pr_\pi[x \in \text{Forced}(F, \alpha_0, \alpha^*, \pi, D)]$$

Combining it with equation (2.4), we then get

$$\mathbb{E}_\pi[|\text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)|] \leq n(\alpha_0) - \sum_{x\in\mathcal{U}(\alpha_0)} \Pr_\pi[|\text{Reachable}(T_x, \pi(x), \pi)| \leq D].$$

By definition of S_D, this gives

$$\mathbb{E}_\pi[|\text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)|] \leq S_D \cdot n$$

where, by Theorem 2.8, $S_D \to S$ when $D \to \infty$. So on average, when $D \to \infty$, the ratio of variables that need to be guessed converges to 38.63%, while the remaining 61.37% of the variables are forced. This can be achieved by selecting D to be some function that grows slowly in n, for instance $D = \log n$. This still allows us to examine all $G \subseteq F$ such that $|G| \leq D$ and check for D-implications in subexponential time, and allows us to obtain Theorem 2.4.

The next sections aim at proving Theorem 2.8. For this, we will proceed in four high-level steps.

1. We will prove a bound on a much simpler case: a totally symmetric infinite full binary tree from which every node is deleted independently with a fixed probability p. This probability p corresponds, in the PPSZ analysis, to the place of a given variable in the permutation.
2. We will then argue that if the tree is not infinite but finite, the bound still holds.
3. We will show that that bound also holds if we introduce dependencies between the nodes.
4. We will finally consider the case where p is not fixed anymore but the place of the root is also random.

2.2.5 Random deletion in infinite binary trees

In this section, we consider the infinite rooted full binary tree T_∞. Each non-root of T_∞ is deleted (along with its subtree) independently from all other nodes with probability p; this yields the tree T'.

We will prove a bound on the probability that the height of T', $h(T')$, doesn't exceed a given d, by means of three lemmas. In the following sections, we will show that this bound still holds if the conditions of the lemma are relaxed.

Lemma 2.9. Let T_∞ be the infinite rooted full binary tree. Consider the following random experiment: each non-root from T_∞ is deleted (along with its subtree) independently from all other nodes with probability p. Then the probability that the resultant tree T' is of finite size is

$$\Pr[T' \text{ is finite}] \geq \zeta(p)$$

where

$$\zeta(p) = \begin{cases} \dfrac{p^2}{(1-p)^2} & \text{if } p < \dfrac{1}{2} \\ 1 & \text{otherwise} \end{cases}$$

Proof. Let $q = \Pr[T' \text{ is finite}]$. For T' to be finite, then each of the root's children must be either deleted (which happens with probability p) or the root of a finite tree, considering that we subject this infinite full binary tree to the same random experiment – and this happens with probability q. Hence, the following holds:

$$q = (p + (1-p) \cdot q)^2$$

This quadratic equation has two solutions,

$$q = 1 \text{ and } q = \frac{p^2}{(1-p)^2}$$

This proves that q is always at least the smaller of the two solutions, which establishes the claim. □

Let us now denote by $h(T)$ the height of T, i.e. the length of the longest path from the root downwards. For an infinite tree T, let $h(T) = \infty$. We can prove the following lemma:

Lemma 2.10. Let T_∞ be the infinite rooted full binary tree. Consider the following random experiment: each non-root node from T_∞ is deleted (along with its subtree) independently from all other nodes with probability p. Then the probability that the resultant tree T' has height at most $d \geq 1$ converges as

$$\lim_{d\to\infty} \Pr[h(T') \leq d] = \Pr[T' \text{ finite}].$$

Proof. Let B_i be the event that there exists a path from the root to some node at depth i. We first claim that

$$\bigcap_{i\geq 1} B_i = \{T' \text{ infinite}\}.$$

Suppose that an event A is contained in $\bigcap_{i\geq 1} B_i$. This means that T' contains finite paths of arbitrary length, and hence cannot be finite. Suppose that T' is finite. Then it contains paths of arbitrary length, which is what we need for the other direction of the equality.

We obviously have that $B_i \supseteq B_{i+1}$ for all $i \geq 1$ and we can then apply the monotone convergence theorem (see Appendix A.2). It follows that

$$\lim_{d\to\infty} B_d = \Pr[T' \text{ infinite}].$$

Taking complements:

$$\lim_{d\to\infty} \Pr[h(T') \leq d] = \lim_{d\to\infty} \bar{B}_d = 1 - \lim_{d\to\infty} B_d = 1 - \Pr[T' \text{ infinite}] = \Pr[T' \text{ finite}].$$

□

Hence, we have

$$\lim_{d\to\infty} \Pr[h(T') \leq d] \geq \zeta(p).$$

Now, instead of using a limit in this statement, we want to introduce a sequence of errors $\varepsilon_i(p)$. We state this in the following lemma:

Lemma 2.11. There exists a sequence $\varepsilon_1(p), \varepsilon_2(p), ... \in \mathbb{R}_0^+$ of numbers depending only on p, having $\varepsilon_d(p) \to 0$ for $d \to \infty$ such that the following holds. Let T_∞ be the infinite full binary tree. Let $p \in [0, 1]$ be a fixed number, and consider the following random experiment: each non-root node from T is deleted (along with its subtree) independently from all other nodes with probability p.

Then the probability that the resultant tree T' has height at most $d \geq 1$ satisfies

$$\Pr[h(T') \leq d] \geq \zeta(p) - \varepsilon_d(p).$$

Proof. Define, for all $d \geq 1$,

$$\varepsilon_d := \max\{\zeta(p) - \Pr[h(T') \leq d], 0\}$$

Then we find that

$$\Pr[h(T') \leq d] \geq \zeta(p) - \varepsilon_d(p)$$

and, from Lemma 2.10,

$$\lim_{d\to\infty} \varepsilon_d(p) = 0$$

as required. □

2.2.6 From infinite to finite trees

Let us now consider any finite (not necessarily full) binary tree. Consider the following random experiment: each non-root node from T is deleted (along with its subtree) independently from all other nodes with probability p.

We can couple this experiment to a random experiment conducted on T_∞: T is embeddable into T_∞ with the two roots coinciding. If we delete every node from T_∞ independently with probability p, the same experiment is taking place on T. Let T'' be the tree resulting from the deletions in T_∞ and T' the tree resulting from the deletions in T. Since T' is a subtree of T'', we have $h(T') \leq h(T'')$, and therefore

$$\Pr[h(T') \leq d] \geq \Pr[h(T'') \leq d]$$

The lower bounds that we obtain in the case of infinite trees are hence also valid in the case of finite, not necessarily full binary trees. Lemma 2.11 can consequently also be applied to finite, not necessarily full binary trees.

It needs to be stressed here that, in particular, the convergence rate (the sequence of errors ε_i) of the probability is independent of which tree T we consider, and only depends on p.

2.2.7 From independent to dependent labels

In the previous section, we've generalized Lemma 2.11 to take into account the fact that, in our setting, the critical clause trees that we consider have a finite height and are not necessarily full. The current version of Lemma 2.11 also requires that the nodes are deleted independently of all other nodes. This is not the case in the result we are eventually after: in our case, dependencies can occur, in that nodes of the trees are linked to one another and, if one of these linked vertices is deleted, all of them are deleted. Conversely, if one of them is not deleted, then none of them is directly deleted (although it can happen that it's deleted following the deletion of a parent node in the tree). In the critical clause tree setting, this corresponds to the fact that a given variable can appear as var-label at several places in the tree, even though, from Observation 2.6, it can appear only once in any path from the root to a leaf (the labelling is injective on these paths).

We generalize Lemma 2.11 again to obtain the following lemma:

Lemma 2.12. *Let $Z_1, Z_2, ..., Z_r \in \{0, 1\}$ be mutually independent binary random variables, each of which takes value 1 with probability p. Let T be any finite (and not necessarily full) binary tree with a labelling $\sigma : V(T)\backslash\{\text{root}\} \to \{1, ..., r\}$ of the non-root of T with indices that have the property that, on each path from the root to a leaf, σ is injective. Consider the experiment of drawing $Z_1, ..., Z_r$ according to their distribution and then deleting all nodes u from T (along with their subtrees) for which $Z_{\sigma(u)} = 1$. Call the resulting tree T'.*

Juxtapose the experiment where in T, every non-root is deleted independently from all other nodes with probability p. Call the random tree resulting from this experiment T''. Then for any d,

$$\Pr[h(T') \leq d] \geq \Pr[h(T'') \leq d] \geq \zeta(p) - \varepsilon_d(p).$$

Proof. The second inequality of the proof is a direct application of the finite version of Lemma 2.11.

Moreover, the statement is trivial if σ is globally injective, because in that case all the nodes have independent labels and we are in the previous case.

Now we suppose that none of the duplicates are in an ancestor-descendant relation (injectiveness on paths from root to a leaf), and we show that the correlations arising from these duplicate labels cannot decrease this probability. For this, we will use the FKG inequality, which is stated here and proven in Appendix A.3.

Theorem 2.13. *Let $\mathcal{A} = \{A_1, A_2, ..., A_r\}$ be a collection of independent binary random variables and E_1 and E_2 events which are determined by $\mathcal{A}$ and monotonically increasing in $\mathcal{A}$. Then*

$$\Pr[E_1 \wedge E_2] \geq \Pr[E_1] \cdot \Pr[E_2].$$

We will use this theorem in the following proof, where we proceed by induction on d. For $d = 0$, the statement is trivial (because both probabilities are 0, since the root is never deleted).

Let $d > 0$, and suppose that the statement holds for any depth strictly smaller than d. If the root of T has no child, the statement is trivial, since both probabilities are 1. If the root of T has only one child, then the statement is a direct consequence of the induction hypothesis, as the whole tree has height d iff the unique subtree of the root has height $d-1$.

Now suppose that the root of T has two children u and v. Let T_u be the subtree rooted at u and let $i = \sigma(u)$. Let T'_u denote the subtree of T' rooted at u and let T''_u the subtree of T'' rooted at u (T' and T'' are empty trees if u is deleted). Let T_v, j, T'_v, T''_v be the corresponding objects for v. The injectiveness hypothesis on σ entails that no other node in T_u is labelled with Z_i, so whatever happens in the non-root nodes of T_u is independent of whether u itself is being deleted or not. The same holds at v.

The event $\{h(T') \leq d\}$ can be defined as the conjunction of two similar events, one on each subtree:

$$E_1 = \{Z_i = 1 \vee (Z_i = 0 \wedge h(T'_u) \leq d-1)\}$$
$$E_2 = \{Z_j = 1 \vee (Z_j = 0 \wedge h(T'_v) \leq d-1)\}$$
$$\{h(T') \leq d\} = E_1 \wedge E_2$$

E_1 and E_2 are determined by $\{Z_1, ..., Z_r\}$ and are monotonically increasing in those events (because if one of the Z_k goes from 0 to 1 then neither E_1 nor E_2 can go from 1 to 0). Therefore, we can apply the FKG inequality to obtain that

$$\Pr[h(T') \leq d] = \Pr[E_1 \wedge E_2] \geq \Pr[E_1] \cdot \Pr[E_2].$$

Since T_u is independent from Z_i, we have that

$$\begin{aligned}\Pr[E_1] &= \Pr[Z_i = 1] + \Pr[Z_i = 0] \cdot \Pr[h(T'_u) \leq d-1] \\ &= p + (1-p) \cdot \Pr[h(T'_u) \leq d-1],\end{aligned}$$

and, by induction hypothesis,

$$\Pr[E_1] \geq p + (1-p)\Pr[h(T''_u) \leq d-1].$$

A similar statement holds for $\Pr[E_2]$. Combining these two statements, we get

$$\begin{aligned}\Pr[h(T') \leq d] &\geq (p + (1-p)\Pr[h(T''_u) \leq d-1]) \cdot (p + (1-p)\Pr[h(T''_v) \leq d-1]) \\ &= \Pr[h(T'') \leq d].\end{aligned}$$

□

2.2.8 Integrating over the place of the root

We are now ready to prove Theorem 2.8, which we recall here:

Theorem 2.8. For any $\varepsilon > 0$ there exists $D_\varepsilon \in \mathbb{N}$ depending only on ε such that the following holds. Let T be a finite binary tree and $\sigma : V(T) \to \{1, ..., r\}$ a labelling of the nodes of T such that on each path from the root to a leaf of T, σ is injective. Let $X_1, X_2, ...X_r$ be real random variables distributed uniformly from $[0, 1]$ and mutually independently. Consider the experiment of drawing $X_1, X_2, ..., X_r$ according to their distribution and then deleting all nodes u from T (along with the corresponding subtrees) for which $X_{\sigma(u)} < X_{\sigma(\text{root})}$.

Then the probability that the resultant tree T' contains more than D_ε nodes is

$$\Pr_{X_1,...,X_r}[|V(T')| > D_\varepsilon] \leq S + \varepsilon$$

where

$$S = \frac{1}{2} - \int_0^{\frac{1}{2}} \frac{p^2}{(1-p)^2} \mathrm{d}p = 2\ln 2 - 1 \leq 0.3863$$

Proof. We fix ε, and we will, at the end of this proof, fix D_ε appropriately so that the statement holds.

Without loss of generality, suppose $\sigma(\text{root}) = X_r$. This value is independent of all the other values used because the root is part of all paths and σ is injective on each path, so X_r doesn't occur a second time as a label.

Now we condition on $X_r = \gamma$ for some fixed value $\gamma \in [0,1]$ and we consider, for $1 \leq i \leq r-1$ the binary random variables

$$Z_i = \begin{cases} 1 & \text{if } X_i < \gamma \\ 0 & \text{otherwise} \end{cases}$$

The variables $\{Z_1, Z_2, ..., Z_{r-1}\}$ are mutually independent binary random variables, each of which takes value 1 with probability exactly γ. This is exactly the situation of Lemma 2.12, and so we can use this result to conclude that

$$\Pr[h(T') \leq d \mid X_r = \gamma] \geq \zeta(\gamma) - \varepsilon_d(\gamma)$$

To convert this conditional into an unconditional probability, we invoke the law of total probability, which, since Z_r is uniformly distributed, reads

$$\Pr[h(T') \leq d] = \int_0^1 \Pr[h(T') \leq d \mid X_r = \gamma] \, \mathrm{d}\gamma$$

Ideally, we'd be able to evaluate the limit of this integral as $d \to \infty$ directly. However, the rate of convergence of ε could depend on γ, which forbids us to invoke a uniform convergence argument and hence to establish the convergence of the limit of the integral.

We may, however, circumvent the problem by approximating the integral by Riemann sums. The following statement is proven in Appendix A.4:

Lemma 2.14. Let $\phi : [0,1] \to [0,1]$ be a continuous and monotonically non-decreasing function. Then for any $N \geq 1$,

$$\frac{1}{N} \sum_{i=0}^{N-1} \varphi\left(\frac{1}{N}\right) \leq \int_0^1 \varphi(x) \, \mathrm{d}x \leq \frac{1}{N} \sum_{i=0}^{N-1} \varphi\left(\frac{1}{N}\right) + \frac{1}{N}$$

Now observe that $\Pr[h(T') \leq d \mid X_r = \gamma]$ is continuous and non-decreasing as a function of γ, so we can indeed apply this approximation. We obtain:

$$\begin{aligned} \int_0^1 \Pr[h(T') \leq d \mid X_r = \gamma] \, \mathrm{d}\gamma &\geq \sum_{i=0}^{N-1} \frac{1}{N} \Pr\left[h(T') \leq d \middle| X_r = \frac{i}{N}\right] \\ &\geq \sum_{i=0}^{N-1} \frac{1}{N} \zeta\left(\frac{i}{N}\right) - \sum_{i=0}^{N-1} \frac{1}{N} \varepsilon_d\left(\frac{i}{N}\right) \end{aligned}$$

Recall that

$$\zeta(p) = \begin{cases} \dfrac{p^2}{(1-p)^2} & \text{if } p < \dfrac{1}{2} \\ 1 & \text{otherwise} \end{cases}$$

Since ζ is also monotonically non-decreasing on $[0,1]$, we can apply the other inequality of Lemma 2.14 and obtain that

$$\Pr[h(T') \leq d] \geq \int_0^1 \zeta(x)\,\mathrm{d}x - \frac{1}{N} - \sum_{i=0}^{N-1} \frac{1}{N}\varepsilon_d\left(\frac{i}{N}\right)$$

By defining

$$\psi(N,d) = \frac{1}{N} + \sum_{i=0}^{N-1} \frac{1}{N}\varepsilon_d\left(\frac{i}{N}\right)$$

we get that, for an arbitrary N,

$$\Pr[h(T') \leq d] \geq \int_0^1 \zeta(x)\,\mathrm{d}x - \psi(N,d)$$

Note that the error term $\psi(N,d)$ depends only on N and d and not on the choice of T'. Therefore, given $\varepsilon > 0$ as in Theorem 2.8, we can simply pick $N = N(\varepsilon)$ which is large enough such that $1/N < \varepsilon/2$, and the pick $d = d(\varepsilon, N)$ large enough such that $\varepsilon_d(x) < \varepsilon/2$ for all of the N points x where the function is evaluated, yielding that $\phi(N,d) < \varepsilon$ and thus all possible trees have a probability of having at most height d of at least $S - \varepsilon$. Since the trees are binary trees, setting $D_\varepsilon > 2^{d(\varepsilon,N(\varepsilon))+1}$ concludes the proof of Theorem 2.8. $\square$

2.2.9 Summary

Let us summarize the salient points of the proof.

- We first related the probability of success (and hence the expected runtime) of the PPSZ algorithm to the expected number of variables that are "guessed" by the PPSZ algorithm.
- We applied Jensen's inequality be able to bound the probability of success by bounding the probability that a given variable is forced.
- We related the fact that a variable is forced to the probability that a tree to which we apply random cuts has a bounded size.
- We finally bounded said probability and proved that the bound converges, for large trees, to the fixed value that we claimed. To achieve this, we
  - computed said probability for a fixed root place and an infinite tree with independent cuts,
  - added the condition that the tree was not necessarily infinite,
  - added the condition that the tree cuts were not necessarily independent of each other,
  - integrated that result to get the final result for any variable, independently of its place.

2.3 Multiple satisfying assignments

In the previous section, we've proven Theorem 2.4, which deals with the case where F has a unique satisfying assignment. In general, though, we cannot rely on such a strong assumption, and we would like to prove Theorem 2.2, which we re-state here:

Theorem 2.2. For any satisfiable (≤ 3)-CNF formula F on n variables V, $\textsc{ppsz}(F, V, \emptyset, \log n)$ returns some satisfying assignment with probability $\Omega(1.3071^{-n})$.

Intuitively, it shouldn't be harder to find a satisfying assignment when there are several of them than when there is only one of them. The analysis of why it is indeed the case, however, gets significantly more complicated in the general case than in the unique case. The original analysis by Paturi et al. [4] gives worse bounds for multiple satisfying assignments than for a unique satisfying assignment. The gap between both analyses was closed by Hertli [2] and it was shown that the bound for unique SAT is indeed achievable for general SAT.

2.3.1 Problem assessment

The main reason why the analysis doesn't go through like in the unique case is that there is no guarantee that we can build critical clause trees for the formula. If a variable has assignments that assign it to two different values, it can well happen that we cannot find clauses that allow us to build the critical clause tree for that variable. In the extreme case, the empty formula has no clause at all, so building trees in that case would definitely be a problem.

If, however, all the assignments for a given variable send that variable to the same value, it's easy to see that we can apply the exact same argument than in the unique case for that variable and build a critical clause tree for that variable. In a way, the unique case is the multiple case in which all variables are sent to a single value each.

We call these variables "frozen", and we can infer the following lemma:

Lemma 2.15. Let F be a (≤ 3)-CNF formula and x be a frozen variable. Let furthermore α be any satisfying assignment. Then $\Pr[x \in \text{Guessed}(F, \alpha_0, \alpha, \pi, D)] \leq S_D$, with S_D defined as in Theorem 2.8.

The proof of this lemma is exactly the same as the proof for the unique case; there is no argument there that cannot be applied to the case of frozen variables.

On the other hand, the variables that are not frozen are not an issue for this step, because we can set them either way and still stay satisfiable. We could then conclude that non-frozen variables are not an issue, and we have a bound for the probability that a frozen variable is guessed. We are, however, not done. The problem is that this bound on the probability depends on the assignment, and that the assignment that is chosen depends highly on the random choice over the non frozen variables. The analysis needs to take this into account, but the correlations arising between the assignment towards which the PPSZ algorithm ends up steering and the probabilities that variables are guessed make said analysis non trivial to say the least.

2.3.2 Proof idea

In Section 2.2, we proved that the probability that PPSZ returned the unique satisfying assignment α^* was bounded by

$$\Pr_{\beta,\pi}[\textsc{ppsz} \text{ returns } \alpha^*] \geq 2^{-\sum_{x \in \mathcal{U}(\alpha_0)} \Pr_\pi[x \in \text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)]}$$

and we proved that, for each of the n variables of the domain,

$$\Pr_\pi[x \in \text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)] \leq S_D$$

which in turn allowed us to bound

$$\Pr_{\beta,\pi}[\textsc{ppsz} \text{ returns } \alpha^*] \geq 2^{-S_D n}$$

Now we want to bound the probability that PPSZ returns *some* satisfying assignment, and we would like the bounds to, at least, match. So, ideally, we want to write that

$$\Pr_{\beta,\pi}[\text{PPSZ returns some satisfying assignment}] \geq 2^{-S_D n}$$

For the analysis, we will consider the different steps of the PPSZ algorithm. This is where the parameter α_0, corresponding to a partial assignment, really comes into play: we are going to evaluate the probability that PPSZ completes the partial assignment α_0 to any satisfying assignment. Supposing that $F^{[\alpha_0]}$ is satisfiable (because if not, then PPSZ cannot return a satisfying assignment), we want to write that

$$\Pr_{\beta,\pi}[\text{PPSZ}(F, V, \alpha_0, D) \text{ returns some satisfying assignment}] \geq 2^{-S_D n(\alpha_0)}.$$

One way to do this could be to find a function, say "cost", that can be associated to each variable such that $\text{cost}(\alpha_0, x) \leq S_D$ for each variable x at any step α_0 of the algorithm, and to show that $\Pr_{\beta,\pi}[\text{PPSZ}(F, V, \alpha_0, D) \text{ returns any satisfying assignment}] \geq 2^{-\sum_{x\in V(F)} \text{cost}(\alpha_0,x)}$.

Defining this function turns out to be the crucial point of the PPSZ analysis. The name "cost" can become clearer if you think of this function as a "measure of badness" putting a numerical value on the question "when setting this variable, how badly can things go with regard to the completion of the algorithm?".

This is what we will do in the following section, and we will eventually prove a lemma that links this cost function to the probability that PPSZ outputs some satisfying assignment, namely Lemma 2.19.

But first, we need to find a suitable cost function.

2.3.3 Definition of a cost function

Before we define the cost function itself, we need a few additional notations. For any formula F, we partition the variables into three categories:

$$\text{vbl}(F) = \text{V}_{\text{fo}}(F) \,\dot\cup\, \text{V}_{\text{fr}}(F) \,\dot\cup\, \text{V}_{\text{nf}}(F)$$

where

- $\text{V}_{\text{nf}}(F)$ are the variables that are not frozen, i.e. the ones that can be assigned either way keeping the formula satisfiable: $\text{V}_{\text{nf}}(F) := \{x \in \text{vbl}(F) \mid F \nvDash x\}$. We define the shorthand $\text{V}_{\text{nf}}(\alpha_0) = \text{V}_{\text{nf}}(F^{[\alpha_0]})$.

- $\text{V}_{\text{fo}}(F)$ are the variable that are currently forced: they are frozen (because they cannot be forced otherwise), and they follow from F via D-implication, that is $\text{V}_{\text{fo}}(F) := \{x \in \text{vbl}(F) \mid F \vDash_D (x \neq 0) \text{ or } F \vDash_D (x \neq 1)\}$. We define the shorthand $\text{V}_{\text{fo}}(\alpha_0) = \text{V}_{\text{fo}}(F^{[\alpha_0]})$.

- $\text{V}_{\text{fr}}(F)$ are the variables that are frozen, but not currently forced: they follow from F, but D-implication is not enough to see it, that is $\text{V}_{\text{fr}}(F) := \{x \in \text{vbl}(F) \mid F \vDash (x \neq 0) \text{ or } F \vDash (x \neq 1), F \nvDash_D x\}$. We define the shorthand $\text{V}_{\text{fr}}(\alpha_0) = \text{V}_{\text{fr}}(F^{[\alpha_0]})$.

Moreover, we want to actually define the cost function with regard to α_0, x and a given assignment α as the cost of x when completing α_0 to α. The reason for this is that the cost of x may vary depending on which assignment the algorithm steers towards; and we want to be able to take into account these correlations. Thus, we define the cost function that we study as $\text{cost}(\alpha_0, \alpha, x)$.

Now we define the cost function itself, by distinguishing several cases.

- If $x \notin \mathcal{U}(\alpha_0)$, then $\text{cost}(\alpha_0, \alpha, x) = 0$: either x is already in α_0, or it doesn't exist in $\text{vbl}(F)$, but in any case it has no influence on the cost of completing α_0 to α.
- If α_0 and α are incompatible, then $\text{cost}(\alpha_0, \alpha, x) = 0$. This corresponds to the intuitive notion that if α_0 and α are incompatible, then PPSZ is already doomed to fail in completing α_0 to α, so whatever we do with x won't hurt that.
- If α is not a satisfying assignment of F, then $\text{cost}(\alpha_0, \alpha, x) = 0$, because PPSZ will never return α, so we don't need to bother with the cost of completing α_0 to α.
- If $x \in \text{V}_{\text{fo}}(\alpha_0)$, then we also define $\text{cost}(\alpha_0, \alpha, x)$ to be 0: since x is forced, then it will not be a problem, and will be set correctly by PPSZ.
- If $x \in \text{V}_{\text{fr}}(\alpha_0)$, then we define $\text{cost}(\alpha_0, \alpha, x) = \Pr[x \in \text{Guessed}(F, \alpha_0, \alpha, \pi, D)]$. This case is the main reason why we have to distinguish the different values for the final assignment. Intuitively, this corresponds to the probability that something "bad" can happen with regard to outputting the assignment α: if the variable is guessed, then, since it is frozen, we have one chance out of two to end up in an unsatisfiable state when processing it.
- The case $x \in \text{V}_{\text{nf}}(\alpha_0)$ is the hardest to justify intuitively. Since the variable is not frozen, it will be set randomly, but we cannot end up in a situation where the formula becomes unsatisfiable. However, the way we choose the variable can make the satisfiability more "difficult". If there exists a large number of satisfying assignments with $x = 0$ but only one with $x = 1$, then the choice of x may have an impact on the overall success probability. We don't know exactly what happens here; but we know that we want the cost for a variable to be less than S_D. Moreover, the idea behind the cost function is that it doesn't increase when we assign new variables. Since a non-frozen variable can become frozen, but not the other way around, it is somewhat logical to try to choose $\text{cost}(\alpha_0, \alpha, x) = S_D$, and hope that everything works out with this value in the analysis[3].

Now we need to relate this cost function for individual assignments to a cost function that encompasses all of them – and hence the cost, for a given formula and a given initial assignment, to get a satisfying assignment.

To understand why we define that cost function as we do, suppose for an instant that the cost function $\text{cost}(\alpha_0, \alpha, x)$ is exactly the probability that something goes wrong (for some definition of wrong that we won't make explicit here) when processing x, considering that we want to complete α_0 to α. We can also see it as the probability that something goes wrong, knowing that the algorithm steers towards α. If we define $\text{cost}(\alpha_0, x)$ as the probability that something goes wrong, regardless of which assignment we steer towards, by the law of total probability, we get:

$$\text{cost}(\alpha_0, x) = \sum_{\alpha \in \{0,1\}^V} \text{cost}(\alpha_0, \alpha, x) \cdot \Pr[\textsc{ppsz}(F, V, \alpha_0, D) \text{ steers towards } \alpha].$$

Let us stress that while the "intuitive" explanations can give a sense of why and how we defined this cost function, they can only be seen as an approximate meaning of the formal definition of the cost function. The fact that Lemma 2.19 holds is solely due to the formal definition of the cost function and of its analysis, and does not rely on these justifications.

Let us now recapitulate the cost function for a given assignment and properly define the cost function for all assignments.

Definition 2.16. Let α_0 be a partial and α be a total assignment and let $x \in V$ be any variable. We define the *cost of x when completing α_0 to α*, in writing $\text{cost}(\alpha_0, \alpha, x)$ as follows:

[3]Spoiler: it does.

- If $x \notin \mathcal{U}(\alpha_0)$, then $\mathrm{cost}(\alpha_0, \alpha, x) = 0$.
- If α_0 and α are incompatible, i.e. $\exists y : \{\alpha_0(y), \alpha(y)\} = \{0, 1\}$, then $\mathrm{cost}(\alpha_0, \alpha, x) = 0$.
- If α does not satisfy F, then $\mathrm{cost}(\alpha_0, \alpha, x) = 0$.
- Else:
  - If $x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)$, then $\mathrm{cost}(\alpha_0, \alpha, x) = 0$.
  - If $x \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$, then

$$\mathrm{cost}(\alpha_0, \alpha, x) = \Pr_{\pi}[x \in \mathrm{Guessed}(F, \alpha_0, \alpha, \pi, D)].$$

  - If $x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$, then $\mathrm{cost}(\alpha_0, \alpha, x) = S_D$.

To define $\mathrm{cost}(\alpha_0, x)$, we first define the *likelihood* of an assignment as follows.

Definition 2.17. Let $F^{[\alpha_0]}$ be satisfiable and let $\mathcal{S}_{\alpha_0}$ be the set of value assignments $l = \{x \mapsto b\}$ such that $x \in \mathcal{U}(\alpha_0)$ and $F^{[\alpha_0[l]]}$ is satisfiable.

We define the random process $\mathrm{AssignSL}(F, \alpha_0)$ that produces an assignment on $\mathrm{vbl}(F)$ as follows. Start with the assignment α_0, and repeat the following step until $\mathrm{vbl}(F^{[\alpha_0]}) = \emptyset$: Choose a value assignment $l \in \mathcal{S}_{\alpha_0}$ uniformly at random and add l to α_0. At the end, output α_0.

Let α be a total assignment on $\mathrm{vbl}(F)$. Then the *likelihood of completing α_0 to α*, in writing $\mathrm{lkhd}(\alpha_0, \alpha)$ is defined as the probability that $\mathrm{AssignSL}(F, \alpha_0)$ returns α. For completeness, if $F^{[\alpha_0]}$ is not satisfiable, we define $\mathrm{lkhd}(\alpha_0, \alpha) = 0$.

Observe that, in the SAT case (this will *not* be the case in the ClSP sections), the likelihood of completing α_0 to α is the probability that $\textsc{ppsz}(F, V, \alpha_0, D)$ for $D = |F|$ outputs α (this can be seen by checking that the probability distribution is the same, by considering a case distinction on whether the variables are frozen or non-frozen).

Also observe that if α_0 and α are incompatible or if α doesn't satisfy F, then $\mathrm{lkhd}(\alpha_0, \alpha) = 0$.

We now define $\mathrm{cost}(\alpha_0, x)$:

Definition 2.18. Let α_0 be a partial assignment over V. The *cost x when completing α_0 to any satisfying assignment*, in writing $\mathrm{cost}(\alpha_0, x)$, is defined as

$$\mathrm{cost}(\alpha_0, x) = \sum_{\alpha \in \{0,1\}^V} \mathrm{lkhd}(\alpha_0, \alpha) \cdot \mathrm{cost}(\alpha_0, \alpha, x).$$

We call the $\mathrm{lkhd}(\alpha_0, \alpha) \cdot \mathrm{cost}(\alpha_0, \alpha, x)$ term in the sum the *weighted cost of x when completing α_0 to α*, in writing

$$\mathrm{wcost}(\alpha_0, \alpha, x) = \mathrm{lkhd}(\alpha_0, \alpha) \cdot \mathrm{cost}(\alpha_0, \alpha, x).$$

Finally, we define the *total cost of completing α_0 to any satisfying assignment* by summing the costs over all possible variables:

$$\mathrm{cost}(\alpha_0) = \sum_{x \in V} \mathrm{cost}(\alpha_0, x).$$

We can finally state the following lemma:

Lemma 2.19. Let α_0 such that $F^{[\alpha_0]}$ is satisfiable. Then the overall probability of PPSZ to output some satisfying assignment when starting in state α_0 is at least $2^{-\mathrm{cost}(\alpha_0)}$.

Also observe the following:

Observation 2.20. For any α_0, α, x, we have $\text{cost}(\alpha_0, \alpha, x) \leq S_D$. Furthermore, $\text{cost}(\alpha_0) \leq S_D n(\alpha_0)$.

Proof. The statement follows directly from the definition of the cost and from Lemma 2.15. $\square$

Consequently, proving Lemma 2.19 allows us to conclude directly that Theorem 2.2 holds as well and we are done.

2.3.4 Proving Lemma 2.19

The proof of Lemma 2.19 will keep us occupied for quite some time, and we will need to take quite a few detours along the road.

Setup of the proof of Lemma 2.19

We first define $p(\alpha_0)$ as the probability that PPSZ outputs some satisfying assignment when starting from state α_0.

Also recall that the set $\mathcal{S}_{\alpha_0}$ is defined as follows:

$$\mathcal{S}_{\alpha_0} := \{(x, b) \in \mathcal{U}(\alpha_0) \times \{0, 1\} \mid F^{[\alpha_0 \cup \{x \mapsto b\}]} \text{ is satisfiable}\}$$

This set represents all the choices that the PPSZ run can make and still have a satisfying formula in the next step. The set $\mathcal{S}_{\alpha_0}$ contains $2 \cdot |\mathrm{V}_{\mathrm{nf}}(\alpha_0)| + |\mathrm{V}_{\mathrm{fo}}(\alpha_0)| + |\mathrm{V}_{\mathrm{fr}}(\alpha_0)|$ elements.

The proof itself is a proof by induction. We suppose that the claim holds for all α_0 that fix a larger number of variables. If α_0 is total, then the statement holds trivially, because the cost of α_0 is 0, and $p(\alpha_0) = 1$.

Let x and b be random variables: $x \in \mathcal{U}(\alpha_0)$ u.a.r, and b is the forced value by D-implication if $x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)$, and u.a.r. otherwise.

We have:

$$p(\alpha_0) = \mathop{\mathbb{E}}_{x \in_{\text{u.a.r}} \mathcal{U}(\alpha_0); b} [p(\alpha_0 \cup \{x \mapsto b\})]$$

Since, if $(x, b) \notin \mathcal{S}_{\alpha_0}$, then by definition of $\mathcal{S}_{\alpha_0}$, $p(\alpha_0 \cup \{x \mapsto b\}) = 0$, we have that

$$p(\alpha_0) = \Pr[(x, b) \in \mathcal{S}_{\alpha_0}] \cdot \mathop{\mathbb{E}}_{x \in_{\text{u.a.r}} \mathcal{U}(\alpha_0); b} [p(\alpha_0 \cup \{x \mapsto b\}) \mid (x, b) \in \mathcal{S}_{\alpha_0}]$$

(x, b) is in $\mathcal{S}_{\alpha_0}$ if:

- $x \in \mathrm{V}_{\mathrm{nf}}$, because then both $(x, 0)$ and $(x, 1)$ are in $\mathcal{S}_{\alpha_0}$;
- $x \in \mathrm{V}_{\mathrm{fr}}$, and b has the correct value: x is not forced, so b is chosen at random in $\{0, 1\}$, but $\mathcal{S}_{\alpha_0}$ contains only one of these values because x is frozen;
- $x \in \mathrm{V}_{\mathrm{fo}}$, because x is forced, so b is not chosen at random but gets assigned its proper value, and $\mathcal{S}_{\alpha_0}$ contains (x, b).

Hence, the probability that $(x, b) \in \mathcal{S}_{\alpha_0}$ is:

$$\begin{aligned}
\Pr[(x, b) \in \mathcal{S}_{\alpha_0}] &= \Pr[x \in \mathrm{V}_{\mathrm{nf}}] + \Pr[x \in \mathrm{V}_{\mathrm{fo}}] + \frac{1}{2}\Pr[x \in \mathrm{V}_{\mathrm{fr}}] \\
&= \frac{|\mathrm{V}_{\mathrm{nf}}(\alpha_0)|}{n(\alpha_0)} + \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{n(\alpha_0)} + \frac{|\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{2n(\alpha_0)} \\
&= \frac{2|\mathrm{V}_{\mathrm{nf}}(\alpha_0)| + 2|\mathrm{V}_{\mathrm{fo}}(\alpha_0)| + |\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{2n(\alpha_0)} \\
&= \frac{|\mathcal{S}_{\alpha_0}| + |\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{2n(\alpha_0)}
\end{aligned}$$

Hence, our working expression becomes:

$$p(\alpha_0) = \frac{|\mathcal{S}_{\alpha_0}| + |\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{2n(\alpha_0)} \mathop{\mathbb{E}}_{x \in_{\mathrm{u.a.r}} \mathcal{U}(\alpha_0); b} [p(\alpha_0 \cup \{x \mapsto b\}) \mid (x, b) \in \mathcal{S}_{\alpha_0}] \,. \qquad (2.5)$$

Relating to uniform choice over $\mathcal{S}_{\alpha_0}$

Equation 2.5 is somewhat cumbersome to work with, because the choice of (x, b) is not made uniformly at random: we first choose x, and then we choose b. It turns out that this makes the expression significantly harder to work with, and that it's much easier to consider a random choice over $\mathcal{S}_{\alpha_0}$.

Luckily, it's fairly easy to relate both probability distributions. The different cases are summarized in the following table. Consider the event "a given (x, b) is chosen next in the process"; we look at the probabilities of said event within both distributions (observe that (x, b) always belongs to $\mathcal{S}_{\alpha_0}$ if $x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$).

	$x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)$		$x \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$		$x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$	
	$(x, b) \notin \mathcal{S}_{\alpha_0}$	$(x, b) \in \mathcal{S}_{\alpha_0}$	$(x, b) \notin \mathcal{S}_{\alpha_0}$	$(x, b) \in \mathcal{S}_{\alpha_0}$	$(x, b) \notin \mathcal{S}_{\alpha_0}$	$(x, b) \in \mathcal{S}_{\alpha_0}$
x, then b	0	$\frac{1}{n(\alpha_0)}$	$\frac{1}{n(\alpha_0)} \cdot \frac{1}{2}$	$\frac{1}{n(\alpha_0)} \cdot \frac{1}{2}$	–	$\frac{1}{n(\alpha_0)} \cdot \frac{1}{2}$
u.a.r in $\mathcal{S}_{\alpha_0}$	0	$\frac{1}{\lvert\mathcal{S}_{\alpha_0}\rvert}$	0	$\frac{1}{\lvert\mathcal{S}_{\alpha_0}\rvert}$	–	$\frac{1}{\lvert\mathcal{S}_{\alpha_0}\rvert}$

We actually don't care what happens when $(x, b) \notin \mathcal{S}_{\alpha_0}$, because then the corresponding term in the expectation will be 0 anyway. The weight of variables in $\mathrm{V}_{\mathrm{fo}}(\alpha_0)$ is twice the weight of the other variables; we define

$$w(x, b) := \begin{cases} 2 & \text{if } x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0) \\ 1 & \text{otherwise} \end{cases}$$

If we change the expectation of equation (2.5) to $(X, B) \in \mathcal{S}_{\alpha_0}$ u.a.r., each $(X, B) \in \mathcal{S}_{\alpha_0}$ now has weight $w(X, B)$. We still have to normalize the weight: the probability distribution must sum to one. Let W be the normalization factor:

$$\begin{aligned}
\sum_{(x', b') \in \mathcal{S}_{\alpha_0}} \Pr[(x', b') \mapsto (X, B)] &= W \cdot \sum_{(X, B) \in \mathcal{S}_{\alpha_0}} w(X, B) \cdot \frac{1}{|\mathcal{S}_{\alpha_0}|} \\
&= W \cdot \frac{|\mathcal{S}_{\alpha_0}| + |\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|}
\end{aligned}$$

So the normalization factor is $W \cdot \frac{|\mathcal{S}_{\alpha_0}|}{|\mathcal{S}_{\alpha_0}|+|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}$, which in turns allows us to write:

$$\begin{aligned} p(\alpha_0) &= \frac{|\mathcal{S}_{\alpha_0}| + |\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{2n(\alpha_0)} \underset{(X,B)\in_{\mathrm{u.a.r.}}\mathcal{S}_{\alpha_0}}{\mathbb{E}} \left[\frac{|\mathcal{S}_{\alpha_0}|}{|\mathcal{S}_{\alpha_0}| + |\mathrm{V}_{\mathrm{fo}}(\alpha_0)|} w(X,B) p(\alpha_0 \cup \{X \mapsto B\}) \right] \\ &= \frac{|S_{\alpha_0}|}{2n(\alpha_0)} \underset{(X,B)\in_{\mathrm{u.a.r.}}\mathcal{S}_{\alpha_0}}{\mathbb{E}} \left[w(X,B) p(\alpha_0 \cup \{X \mapsto B\}) \right]. \end{aligned}$$

Using the induction hypothesis

We have now expressed $p(\alpha_0)$ in terms of an expectation that depends on the uniform choice over the possible litterals of $F^{[\alpha_0]}$. From now on, we will omit in this proof the $(X, B) \in_{\mathrm{u.a.r.}} S_{\alpha_0}$ for ease of notation and reading.

We first use Jensen's inequality, so that we get expressions with which we can work. This yields:

$$\begin{aligned} p(\alpha_0) &\geq 2^{\log\left(\frac{|S|}{2n(\alpha_0)}\right)} 2^{\mathbb{E}[\log(w(X,B)p(\alpha_0 \cup \{X \mapsto B\})]} \\ &= 2^{\log\left(\frac{|S|}{2n(\alpha_0)}\right)} 2^{\mathbb{E}[\log(w(X,B))] - \mathbb{E}[-\log(p(\alpha_0 \cup \{X \mapsto B\}))]} \end{aligned}$$

by linearity of expectation.

Since $\log(1) = 0$ and $\log(2) = 1$, we have exactly that

$$\mathbb{E}[\log(w(X,B))] = \Pr[w(X,B) = 2] = \Pr[x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)] = \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|}$$

Moreover, we can now apply the induction hypothesis, which states that

$$p(\alpha_0 \cup \{X \mapsto B\}) \geq 2^{-\mathrm{cost}(\alpha_0 \cup \{X \mapsto B\})}$$

to state that

$$\mathbb{E}[-\log(p(\alpha_0 \cup \{X \mapsto B\}))] \leq \mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{X \mapsto B\})]$$

Putting everything together, we obtain:

$$p(\alpha_0) \geq 2^{\log\left(\frac{|S|}{2n(\alpha_0)}\right) + \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} - \mathbb{E}[\mathrm{cost}(\alpha \cup \{X \mapsto B\})]} \tag{2.6}$$

Now the only thing that we need to evaluate is $\mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{X \mapsto B\})]$. To do this, we will first need to establish a few facts about cost and likelihood.

Proving facts about cost and likelihood

We can gather the facts that we want to establish in a lemma:

Lemma 2.21. Let α_0 and α be fixed and compatible. For any fixed variable $x \in \mathcal{U}(\alpha_0)$, if we set x according to α, then

(i) the likelihood of α can only increase, i.e.

$$\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \geq \mathrm{lkhd}(\alpha_0, \alpha)$$

with equality if x is frozen in $F^{[\alpha_0]}$.

(ii) the cost of a fixed variable $y \in V$ w.r.t. α can only decrease, i.e.

$$\mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y) \leq \mathrm{cost}(\alpha_0, \alpha, y)$$

When choosing $x \in \mathcal{U}(\alpha_0)$ uniformly at random and setting it according to α, then

(iii) the likelihood of α increases on average as

$$\mathbb{E}[\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)] = \left(1 + \frac{|\mathrm{V}_{\mathrm{nf}}(\alpha_0)|}{n(\alpha_0)}\right) \mathrm{lkhd}(\alpha_0, \alpha)$$

(iv) the cost of a fixed, frozen, non forced variable $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$ decreases on expectation as

$$\mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)] \leq \mathrm{cost}(\alpha_0, \alpha, y) - \frac{1}{n(\alpha_0)}$$

Proof. (i) We have observed that $\mathrm{lkhd}(\alpha_0, \alpha) = \mathbb{E}[\textsc{ppsz}(F, V, \alpha_0, |F|) = \alpha]$. Assume for the moment that the permutation π is fixed. To output α, every non-frozen variable that PPSZ encounters has to be set to the right value, which happens with probability 1/2. If $D = |F|$, all frozen variables are automatically forced. Hence, for a fixed π, $\mathrm{lkhd}(\alpha_0, \alpha)$ is $2^{-\mathrm{nf}(\pi,\alpha_0,\alpha)}$, where $\mathrm{nf}(\pi, \alpha_0, \alpha)$ denotes the number of non-frozen variables encountered. Therefore we have

$$\mathrm{lkhd}(\alpha_0, \alpha) = \mathbb{E}_{\pi}[2^{-\mathrm{nf}(\pi,\alpha_0,\alpha)}].$$

Moving x to the beginning in π can only decrease the number of non-frozen variables. Furthermore, if x is frozen in $F^{[\alpha_0]}$, the number of non-frozen variables remains the same.

Now observe that if we remove x from π, the resulting permutation π' has a uniform distribution from permutations over $\mathcal{U}(\alpha_0)\backslash\{x\}$. The number of non-frozen variables can only decrease in $\mathcal{U}(\alpha_0)\backslash\{x\}$; removal of x might decrease this even further. Therefore

$$\mathbb{E}_{\pi}[2^{-\mathrm{nf}(\pi,\alpha_0,\alpha)}] \leq \mathbb{E}_{\pi'}[2^{-\mathrm{nf}(\alpha_0 \cup \{x \mapsto \alpha(x))\},\alpha)}],$$

with equality if x is frozen, as in this case x is always assigned $\alpha(x)$. The latter term is equal to $\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)$, and we are done.

(ii) We consider the three cases of Definition 2.16. Note that if $x = y$, the statement holds trivially.

If $y \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$, then $\mathrm{cost}(\alpha_0, \alpha, y) = S_D$. Since, by Observation 2.20, the cost of a variable is always less than S_D, the statement holds.

If $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$ or $y \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)$, then $\mathrm{cost}(\alpha_0, \alpha, y)$ is the probability that y is guessed in the remainder of PPSZ. If we now fix another variable x to $\alpha(x)$, then this probability cannot decrease, because adding a value assignment cannot "un-force" another variable. So $\mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y) \leq \mathrm{cost}(\alpha_0, \alpha, y)$.

(iii) We have

$$\begin{aligned}
\mathrm{lkhd}(\alpha_0, \alpha) &= \sum_{(x,c)\in\mathcal{S}_{\alpha_0}} \frac{1}{|\mathcal{S}_{\alpha_0}|} \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto c\}, \alpha) \\
&= \sum_{x\in\mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{S}_{\alpha_0}|} \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \\
&= \frac{n(\alpha_0)}{|\mathcal{S}_{\alpha_0}|} \mathbb{E}_{x\in_{\mathrm{u.a.r}}\mathcal{U}(\alpha_0)}[\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)],
\end{aligned}$$

which proves the statement, since $|\mathcal{S}_{\alpha_0}| = 2|\mathrm{V}_{\mathrm{nf}}(\alpha_0)| + |\mathrm{V}_{\mathrm{fr}}(\alpha_0)| + |\mathrm{V}_{\mathrm{fo}}(\alpha_0)| = n(\alpha_0) + |\mathrm{V}_{\mathrm{nf}}(\alpha_0)|$.

(iv) The statement tells us that the probability that y is guessed is reduced by $1/n(\alpha_0)$ after one step. This is because with probability $1/n(\alpha_0)$, y comes next in π and is guessed now (since it is not forced now). This $1/n(\alpha_0)$ is counted in $\mathrm{cost}(\alpha_0, \alpha, y)$ but not in $\mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)]$.

□

Evaluating $\mathbb{E}[\mathbf{cost}(\alpha_0 \cup \{X \mapsto B\})]$

In this section, we will prove the following lemma:

Lemma 2.22. If $(X, B) \in \mathcal{S}_{\alpha_0}$ is selected uniformly at random, then

$$\mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{X \mapsto B\})] \leq \mathrm{cost}(\alpha_0) - \frac{|\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} - \frac{2S_D \cdot |\mathrm{V}_{\mathrm{nf}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|}$$

To prove this lemma, we will need the following auxilliary lemma:

Lemma 2.23. Let α be a fixed satisfying assignment and $\alpha_0 \subseteq \alpha$. Let $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$ be a fixed frozen variable. If we select $x \in \mathcal{U}(\alpha_0)$ uniformly at random and assign it according to α, then

$$\mathbb{E}[\mathrm{wcost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)] \leq \mathrm{wcost}(\alpha_0, \alpha, y) \cdot \frac{\mathcal{S}_{\alpha_0}}{n(\alpha_0)} - \frac{\mathrm{lkhd}(\alpha_0, \alpha)}{n(\alpha_0)}$$

Proof. We use the following correlation inequality, proven in Appendix B.2.1:

Lemma 2.24. Let $A, B \in \mathbb{R}$ be random variables and $a, b, \bar{a}, \bar{b} \in \mathbb{R}$ fixed numbers such that $A \geq a$ and $B \leq b$ always, and $\mathbb{E}[A] = \bar{a}$ and $\mathbb{E}[B] = \bar{b}$. Then

$$\mathbb{E}[A \cdot B] \leq a\bar{b} + b\bar{a} - ab.$$

To apply that lemma, we recall that $\mathrm{wcost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y) = \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \cdot \mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)$, so we define $A = \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)$ and $B = \mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)$. Recall that, in Lemma 2.21, we have proven the following facts:

- For any fixed variable $x \in \mathcal{U}(\alpha_0)$, if we set x according to α, then $\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \geq \mathrm{lkhd}(\alpha_0, \alpha)$ and $\mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y) \leq \mathrm{cost}(\alpha_0, \alpha, y)$.
- When choosing $x \in \mathcal{U}(\alpha_0)$ u.a.r. and setting it according to α, the likelihood of α becomes, on average
$$\mathbb{E}[\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\})] = \left(1 + \frac{|\mathrm{V}_{\mathrm{nf}}(\alpha_0)|}{n(\alpha_0)}\right) \mathrm{lkhd}(\alpha_0, \alpha),$$
and the cost of a fixed variable $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$ becomes, on average,
$$\mathbb{E}[\mathrm{cost}(\alpha \cup \{x \mapsto \alpha(x)\}, \alpha, y)] = \mathrm{cost}(\alpha_0, \alpha, y) - \frac{1}{n(\alpha_0)}.$$

With these results, we can set, in Lemma 2.24, $a = \mathrm{lkhd}(\alpha_0, \alpha)$, $\bar{a} = \left(1 + \frac{|\mathrm{V}_{\mathrm{nf}}(\alpha_0)|}{n(\alpha_0)}\right) \mathrm{lkhd}(\alpha_0, \alpha)$, $b = \mathrm{cost}(\alpha_0, \alpha, y)$ and $\bar{b} = \mathrm{cost}(\alpha_0, \alpha, y) - \frac{1}{n(\alpha_0)}$. Applying our correlation inequality, we deduce that, for $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$, when selecting $x \in \mathcal{U}(\alpha_0)$ uniformly at random and assigning it according to α, we have

$$\begin{aligned}
& \mathbb{E}[\mathrm{wcost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)] \\
\leq\ & a\bar{b} + b\bar{a} - ab \\
=\ & \mathrm{lkhd}(\alpha_0, \alpha)\left(\mathrm{cost}(\alpha_0, \alpha, y) - \frac{1}{n(\alpha_0)}\right) + \mathrm{cost}(\alpha_0, \alpha, y)\left(1 + \frac{|\mathrm{V}_{\mathrm{nf}}(\alpha_0)|}{n(\alpha_0)}\right)\mathrm{lkhd}(\alpha_0, \alpha) \\
& - \mathrm{lkhd}(\alpha_0)\mathrm{cost}(\alpha_0, \alpha, y) \\
=\ & \mathrm{wcost}(\alpha_0, \alpha, y)\left(1 + \frac{|\mathrm{V}_{\mathrm{nf}}(\alpha_0)|}{n(\alpha_0)}\right) - \frac{\mathrm{lkhd}(\alpha_0, \alpha)}{n(\alpha_0)} \\
=\ & \mathrm{wcost}(\alpha_0, \alpha, y) \cdot \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} - \frac{\mathrm{lkhd}(\alpha_0, \alpha)}{n(\alpha_0)}.
\end{aligned}$$

□

We can now prove Lemma 2.22.

Proof of Lemma 2.22. We analyze the cost decrease contribution of the different types of variables separately. Each variable forced in state α_0 contributes zero cost both before and after the step.

For a non-frozen variable $y \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$, note that $\mathcal{S}_{\alpha_0}$ features two pairs containing y, in contrast to the other types of variables of which $\mathcal{S}_{\alpha_0}$ features only one pair each. This means that with probability $2|\mathrm{V}_{\mathrm{nf}}(\alpha_0)|/|\mathcal{S}_{\alpha_0}|$, a pair featuring y is selected. In that case, the cost contribution of y drops from S_D to zero in all assigments. No matter what happens to these costs otherwise (they can certainly not increase by definition), the non-frozen variables hence contribute the last term of the claimed inequality.

Now consider the frozen variables. If we fix some satisfying and α_0-compatible assigment $\alpha \in \{0,1\}^V$, and condition the experiment on the event that $\alpha(X) = B$, then X becomes uniformly at random among $\mathcal{U}(\alpha_0)$ because $\mathcal{S}_{\alpha_0}$ contains exactly one pair (x', b') per variable $x' \in \mathcal{U}(\alpha_0)$ such that $\alpha(x') = b'$. Now we can directly apply Lemma 2.23 to find that, conditioning on $\alpha(X) = B$, the cost of each frozen variable drops on average as:

$$\mathbb{E}[\mathrm{wcost}(\alpha_0 \cup \{X \mapsto B\}, \alpha, y) \mid \alpha(X) = B] \leq \mathrm{wcost}(\alpha_0, \alpha, y) \cdot \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} - \frac{\mathrm{lkhd}(\alpha_0, \alpha)}{n(\alpha_0)}.$$

The condition itself is satisfied with probability $n(\alpha_0)/|\mathcal{S}_{\alpha_0}|$. If it does not apply, then the cost contribution of α drops to zero altogether. Therefore, the unconditional change can be obtained by multiplying the right-hand side by $n(\alpha_0)/|\mathcal{S}_{\alpha_0}|$, which then yields

$$\mathbb{E}[\mathrm{wcost}(\alpha_0 \cup \{X \mapsto B\}, \alpha, y)] \leq \mathrm{wcost}(\alpha_0, \alpha, y) - \frac{\mathrm{lkhd}(\alpha_0, \alpha)}{|\mathcal{S}_{\alpha_0}|}.$$

If we sum over all assignments $\alpha \in \{0,1\}^V$, the claim follows. □

Putting everything together

We are back to the main track of the proof of Lemma 2.19, and we can now introduce the result from Lemma 2.22 into equation (2.6). We obtain:

$$p(\alpha_0) \geq 2^{\log\left(\frac{|\mathcal{S}_{\alpha_0}|}{2n(\alpha_0)}\right) + \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} - \mathrm{cost}(\alpha_0) + \frac{|\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} + \frac{2S_D \cdot |\mathrm{V}_{\mathrm{nf}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|}}$$

Since we want to show that $p(\alpha_0) \geq 2^{-\mathrm{cost}(\alpha_0)}$, we need to show that:

$$\log\left(\frac{|\mathcal{S}_{\alpha_0}|}{2n(\alpha_0)}\right) + \frac{|\mathrm{V_{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} + \frac{|\mathrm{V_{fr}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} + \frac{2S_D \cdot |\mathrm{V_{nf}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} \geq 0.$$

We write

$$\begin{aligned}
\log\left(\frac{|\mathcal{S}_{\alpha_0}|}{2n(\alpha_0)}\right) &= -1 + \log\left(\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\right) \\
&= -1 + \log\left(\frac{|\mathrm{V_{fr}}(\alpha_0)| + |\mathrm{V_{fo}}(\alpha_0)| + 2|\mathrm{V_{nf}}(\alpha_0)|}{n(\alpha_0)}\right) \\
&= -1 + \log\left(1 + \frac{|\mathrm{V_{nf}}(\alpha_0)|}{n(\alpha_0)}\right)
\end{aligned}$$

We now use an inequality about logarithms, namely that for $x \geq 0$,

$$\log(1+x) \geq \log(e)\frac{x}{1+x}$$

This inequality is proven in Appendix B.1.1.

This yields:

$$\begin{aligned}
\log\left(\frac{|\mathcal{S}_{\alpha_0}|}{2n(\alpha_0)}\right) &\geq -1 + \log(e)\frac{\frac{|\mathrm{V_{nf}}(\alpha_0)|}{n(\alpha_0)}}{1 + \frac{|\mathrm{V_{nf}}(\alpha_0)|}{n(\alpha_0)}} \\
&= -1 + \log(e) + \frac{|\mathrm{V_{nf}}(\alpha_0)|}{n(\alpha_0) + |\mathrm{V_{nf}}(\alpha_0)|} \\
&= -1 + \log(e)\frac{|\mathrm{V_{nf}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|}
\end{aligned}$$

Inserting this into the inequality, we get:

$$\begin{aligned}
&-1 + \log(e)\frac{|\mathrm{V_{nf}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} + \frac{|\mathrm{V_{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} + \frac{|\mathrm{V_{fr}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} + \frac{2S_D \cdot |\mathrm{V_{nf}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} \geq 0 \\
\Leftrightarrow\ &-|\mathcal{S}_{\alpha_0}| + \log(e)|\mathrm{V_{nf}}(\alpha_0)| + |\mathrm{V_{fo}}(\alpha_0)| + |\mathrm{V_{fr}}(\alpha_0)| + 2S_D \cdot |\mathrm{V_{nf}}(\alpha_0)| \geq 0
\end{aligned}$$

Since $\log(e) + 2S_D > 1.44 + 2 \cdot 0.38 > 2$, the left hand side of the inequality is at least 0 as $\mathcal{S}_{\alpha_0} = 2|\mathrm{V_{nf}}(\alpha_0)| + |\mathrm{V_{fo}}(\alpha_0)| + |\mathrm{V_{fr}}(\alpha_0)|$ and we are done proving Lemma 2.19.

2.4 Summary

To prove Theorem 2.2, we went through the following steps:

1. We first proved Theorem 2.4, which makes the analysis of the algorithm somewhat easier by considering a single satisfying assignment.
2. We used the bound obtained in this analysis to define a cost function for
   - variables in an assignment,
   - a single assignment,
   - and the formula itself.

3. By definition, this cost is bounded by the value that we need to establish the same bound on the runtime of PPSZ as in the unique case.

4. We proved that, indeed, the probability that PPSZ returns any satisfying assignment is at least $2^{-\text{cost}(F)}$, where $\text{cost}(F)$ is the cost of returning a satisfying assignment when starting with an empty assignment.

Chapter 3

A weak version of PPSZ for (d,k)-ClSP

We have seen in Chapter 2 how to prove running time bounds for the PPSZ algorithm applied on 3-SAT formulas. We have introduced in Chapter 1 the notion of ClSP formulas, which can be seen as an extension of SAT formulas: the only thing that changes is the domain over which the variables are defined. In the following two chapters, we will extend PPSZ and its analysis to get bounds for the runtime of PPSZ for (d,k)-ClSP formulas.

The results presented in the following two chapters have been developed by Szedlák [7] for the unique case and Millius [3] for the general case; these chapters are heavily based on these two theses.

In this chapter, we will focus on a "weak" version of the PPSZ algorithm (we'll see in what sense in Section 3.1); next chapter is devoted to a "strong" version of the algorithm that has better results than the "weak" version. We still present the weak version for two reasons. The first is that the analysis is somewhat simpler than the strong version – in a sense, it can serve as a warm-up for the strong version. The second reason is that the analysis itself is, at the time of this write-up, more complete for the weak version than for the strong version. In the case of the weak version, Millius [3] has been able to prove that the general case bound is at least as good as the unique case bound. In the case of the strong version, no such proof has been fully established yet, although it is our belief that it is indeed feasible.

3.1 The algorithm

In this chapter, we will consider the following algorithm for (d,k)-ClSP formulas.

```
PPSZ-WEAK$(F, V, \alpha_0, D)$
    $\pi \leftarrow$ a permutation of $\mathcal{U}(\alpha_0)$ chosen u.a.r.;
    $\beta \leftarrow$ an assignment from $[d]^{\mathcal{U}(\alpha_0)}$ chosen u.a.r.;
    **return** PPSZ-WEAK$(F, V, \alpha_0, D, \pi, \beta)$;
```

```
PPSZ-WEAK$(F, V, \alpha_0, D, \pi, \beta)$
    $\alpha_{\text{prog}} \leftarrow \alpha_0$;
    for $i \leftarrow 1$ to $n(\alpha_0)$
            do
                $x \leftarrow x_{\pi_i}$
                if there exists $c \in \{1, ..., d\}$ such that $\forall c' \neq c$, $F^{[\alpha_{\text{prog}}]} \vDash_D (x \neq c')$
                    then $\alpha_{\text{prog}}(x) \leftarrow c$
                    else  $\alpha_{\text{prog}}(x) \leftarrow \beta(x)$;
    if $\alpha_{\text{prog}}$ satisfies $F$
        then return $\alpha_{\text{prog}}$;
        else  return 'failure';
```

The reader may have spotted an obvious improvement to this algorithm. In this algorithm, we choose values u.a.r. among all the values of the domain, i.e. $[d]$, when there is no "forced" value. However, it may be that some values of $[d]$ can already be excluded when looking at D clauses. Intuitively, reducing the choice domain at each step to consider only the allowed values can only increase the probability of returning a satisfying assignment, because with some positive probability this algorithm will make a value assignment it "knows" to be forbidden.

This is exactly the sense in which the algorithm that we present here is "weak". We will consider the suggested improvement in the next chapter.

We will follow the same structure for this algorithm as we did for the 3-SAT case: we will first prove a theorem concerning the unique case, and we will then show that the considered bound holds for the general case.

The probabilities and runtimes that we achieve to prove here depend on d and k; we define the constant $S_{(d,k)}$ to encompass this dependency:

$$S_{(d,k)} = \int_0^1 \frac{t^{\frac{1}{(d-1)(k-1)}} - t}{1-t} \, \mathrm{d}t$$

We will prove the following theorem.

Theorem 3.1. For any satisfiable (d, k)-ClSP formula F on n variables V, PPSZ-WEAK$(F, V, \alpha_0, \log \log n)$ returns some satisfying assignment with probability $\Omega(d^{-S_{(d,k)}n-o(n)})$.

This theorem imply the following:

Corollary 3.2. There exists a randomized algorithm for (d, k)-ClSP with one-sided error that runs in time $\mathcal{O}(d^{S_{(d,k)}n+o(n)})$.

As we did in the 3-SAT case, we will first prove a weaker version of this theorem and consider first the case of a unique satisfying assignment.

Theorem 3.3. For any (d, k)-ClSP formula F on n variables V which has a unique satisfying assignment, PPSZ-WEAK$(F, V, \alpha_0, \log \log n)$ returns this assignment with probability $\Omega(d^{-S_{(d,k)}n-o(n)})$.

3.2 Unique satisfying assignment

In this section, we consider that the formula F has a unique satisfying assignment α^*. Without loss of generality, we suppose that this satisfying assignment is the all-d assignment, i.e. $\alpha^* = (d, d, ..., d)$. In this section, we can also assume that $\alpha_0 = \emptyset$, $\mathcal{U}(\alpha_0) = V$ and $n(\alpha_0) = n$.

3.2.1 Forced and guessed variables

We extend the notion of forced and guessed variables defined in Chapter 2 so that it can be used for the ClSP case.

Definition 3.4. Let F be a (d, k)-ClSP formula over n variables, α^* a satisfying assignment, α_0 a partial assignment that is compatible with α^*, $\pi = x_1, ..., x_{n(\alpha_0)}$ a permutation of $\mathcal{U}(\alpha_0)$ and $D \geq 0$. A variable x_i is called *forced* with respect to F, α_0, α^*, π and D if there exists $c \in [d]$ such that for all $c' \in [d], c' \neq c$, $F^{[\alpha_0 \cup x_1 \mapsto \alpha^*(x1), x_2 \mapsto \alpha^*(x2), ..., x_{i-1} \mapsto \alpha^*(x_{i-1})]}$ D-implies the literal $(x \neq c')$. Otherwise, the variable is called *guessed*. We denote the set of forced (guessed) variables within $\mathcal{U}(\alpha_0)$ by $\text{Forced}(F, \alpha_0, \alpha^*, \pi, D)$ $(\text{Guessed}(F, \alpha_0, \alpha^*, \pi, D))$.

Observe that PPSZ-WEAK$(F, V, \alpha_0, D, \pi, \beta)$ returns the satisfying assignment α^* if and only if $\beta(x) = \alpha^*(x)$ for all $x \in \text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)$.

From this observation, we can write:

$$\Pr_{\beta,\pi}[\text{PPSZ-WEAK returns } \alpha^*] = \mathop{\mathbb{E}}_{\pi}\left[d^{-|\text{Guessed}(F,\alpha_0,\alpha^*,\pi,D)|}\right].$$

We apply Jensen's inequality (see Appendix A.1) with the convex function $x \mapsto d^{-x}$ and we obtain

$$\Pr_{\beta,\pi}[\text{PPSZ-WEAK returns } \alpha^*] \geq d^{-\mathbb{E}_\pi[|\text{Guessed}(F,\alpha_0,\alpha^*,\pi,D)|]}.$$

By linearity of expectation, we can write that

$$\begin{aligned}
\mathop{\mathbb{E}}_{\pi}[|\text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)|] &= \sum_{x \in \mathcal{U}(\alpha_0)} \Pr_{\pi}[x \in \text{Guessed}(F, \alpha_0, \alpha^*, \pi, D)] \\
&= n(\alpha_0) - \sum_{x \in \mathcal{U}(\alpha_0)} \Pr_{\pi}[x \in \text{Forced}(F, \alpha_0, \alpha^*, \pi, D)]
\end{aligned}$$

3.2.2 Building critical clause trees

So far, the reasoning is exactly the same as in the 3-SAT case. The first main difference when building critical clause trees. In the 3-SAT case, we had binary trees. Here, we are not restricting k to 3 (which was the reason why the trees were binary), and we are considering domains that have an arbitrary number of values, which will make the construction slightly more complicated. The core reasoning stays the same as in the 3-SAT case, though.

We construct a collection of trees $\{T_x\}_{x \in \mathcal{U}(\alpha_0)}$, each of them called a *critical clause tree of* x.

We consider a formula F that has a unique assignment α^*, and let α^* be, without loss of generality, the all-d assignment.

Definition 3.5. We call T a *rooted tree with children into j directions* if the following holds. T is a tree with a designated root, $\text{root}(T)$. The children of a vertex v are partitioned into j groups which we denote $\text{Children}_1(v)$, $\text{Children}_2(v)$,..., $\text{Children}_j(v)$. Each child belongs to exactly one group, i.e. $\text{Children}_i(v)$ and $\text{Children}_k(v)$ are disjoint sets whenever $i \neq k$.

T_x is a rooted tree in $(d-1)$ direction, where every node $u \in V(T)$ is labelled both with a variable $x \in V$, which we denote by var-label(u), and a set of clauses $\mathcal{C} \in F^{[\alpha_0]}$, denoted by clause-label(u). Here is how T_x is built for a fixed $x \in \mathcal{U}(\alpha_0)$:

1. Start with T_x consisting of a single root. This root has variable label x, and an empty clause label.

2. As long as there is a leaf $u \in V(T)$ that has an empty clause label, do the following:

   (a) Define $W := \{\text{var-label}(v) \mid v \in V(T) \text{ is an ancestor of } u \text{ in } T\}$, where *ancestor* includes u itself and the root.

   (b) Let the path from the root to u be $\{y_0, y_1, y_2, ..., y_m, u\}$ be such that var-label$(y_0) = x$, $y_1 \in \text{Children}_{\ell_1}(x)$, $y_2 \in \text{Children}_{\ell_2}(y_1)$, ..., $y_m \in \text{Children}_{\ell_m}(y_{m-1})$, $u \in \text{Children}_{\ell_{m+1}}(y_m)$. Then $\ell_1, ..., \ell_{m+1}$ are well-defined, and uniquely defined. We define the partial assignment μ_0 as follows:

$$\mu_0 : \text{vbl}(F) \to \{0,1\}, \begin{cases} \mu_0(\text{var-label}(y_i)) = \ell_{i+1} & \forall i \in \{0, ..., m\} \\ \mu_0(z) = \alpha^*(z) = d & \forall z \notin \{\text{var-label}(y_0), ..., \\ & \text{var-label}(y_m), \text{var-label}(u)\} \end{cases}$$

   (c) For $j = 1$ to $d-1$, we define $\mu_j = \mu_0[\text{var-label}(v) \mapsto j]$. For each j, let C_j be a constraint that is not satisfied by μ_j. Since μ_j is not compatible with α^* and α^* is the unique satisfying assignment, such a clause exists. Add C_j to clause-label(u).

   (d) For each literal $(y \neq d)$ in C_j, add a node to Children$_j(v)$, which is var-labeled with the variable the literal is over.

We denote the resulting tree by T_x. Note that T_x is not unique for a given x. We still consider the collection $\{T_x\}_{x \in \mathcal{U}(\alpha_0)}$ to be fixed from now on.

Any given node has at most $(d-1)(k-1)$ children: for each $j = 1$ to $d-1$, we add a group of at most $(k-1)$ children: it cannot happen that a clause has k literals $(y \neq d)$ because the all-d assignment is a satisfying assignment.

Suppose v is an ancestor of u and var-label$(v) = y$. Since $\mu_0(y) \neq d$, it cannot happen that clause-label(u) contains a clause that has a literal $(y \neq d)$ (otherwise this clause would be satisfied). Therefore:

Observation 3.6. In T_x, no node has the same var-label as one of its proper ancestors.

This also implies that the height of the tree cannot exceed n, and thus the process terminates, making T_x well-defined.

3.2.3 Critical clause trees and forced variables

As in Section 2.2.4, we now consider π as a placement; the values $\pi(x)$, called place of x, are chosen independentely and uniformly at random from $[0,1]$ for each $x \in \mathcal{U}(\alpha_0)$.

Let $\gamma \in [0,1]$ and T_x be the critical clause tree for some fixed variable. We can use the same definition as in Section 2.2.4 for *reachable nodes*: a node $u \in T_x$ is reachable at time γ w.r.t. π if there exists a path $v_0, v_1, ..., v_m$ such that v_0 is the root of the tree, $v_m = u$ and $\pi(v_i) \geq \gamma$ for all $1 \leq i \leq m$. Let us denote Reachable(T_x, γ, π) the set of all nodes in T_x reachable at time γ w.r.t. π. Observe that this set is independent of the place of x.

A slightly adapted version of Lemma 2.7 can be proven:

Lemma 3.7. *If we have* $|\text{Reachable}(T_x, \pi(x), \pi)| \leq D$, *then it holds as well that* $x \in \text{Forced}(F, \alpha_0, \alpha, \pi, (d-1)D)$.

Observe that these two lemmas are equivalent for $d = 2$.

Proof. Let α' be the restriction of $\alpha^* = (d, d, ..., d)$ to the variables $y \in \mathcal{U}(\alpha_0)$ with $\pi(y) < \pi(x)$. By definition, x is forced if there is a formula $F' \subseteq F^{[\alpha_0 \cup \alpha']}$ that implies all the literals $(x \neq c)$ for all $c \in [d], c \neq d$. Let $G :=$ clause-label(Reachable$(T_x, \pi(x), \pi)$), i.e. the subformula of F

consisting of the union of all the clause-labels sets of reachable nodes in T_x. Since by hypothesis $|\text{Reachable}(T_x, \pi(x), \pi)| \leq D$, and each node's clause-label contains exactly $(d-1)$ clauses, then clearly $|G| \leq (d-1)D$.

Suppose that there exists c such that G does not imply $(x \neq c)$. Then we can fix an assignment $\nu : V \to [d]$ which is compatible with $\alpha_0 \cup \alpha'$, which has $\nu(x) = c$ and which satisfies G. Choose a maximal path in T_x, starting at the root, x, and containing only nodes v such that $\nu(\text{var-label}(v)) \neq d$, following the directions Children_i defined by ν. Since $\nu(x) = c$, this path is non-empty. Let u be its endpoint. Since ν is compatible with α^* on all the variables before x, it must be that var-label(u) is either x itself or after x, and hence $\pi(\text{var-label}(u)) \geq \pi(x)$, and so u is reachable, by definition. For all children z of u, we have that $\nu(\text{var-label}(z)) = d$ (because the path is maximal); all ancestors y of u are such that $\nu(\text{var-label}(y)) \neq d$. By definition of T_x, at least one clause-label of u is unsatisfied by ν and, since clause-label$(u) \subseteq G$, this is contradiction. □

It follows immediately from Lemma 3.7 that, over the uniform choice of π, we have

$$\Pr_{\pi}[x \in \text{Forced}(F, \alpha_0, \alpha^*, \pi, D)] \geq \Pr_{\pi}\left[|\text{Reachable}(T_x, \pi(x), \pi)| \leq \frac{D}{d-1}\right].$$

The same reasoning as in Section 2.2.4 can be applied here: we reduce the problem to a probabilistic computation on trees: when sorting the nodes of a fixed $(d-1)(k-1)$-degree tree according to a random permutation (some nodes have the same label and are prescribed to get assigned the same place) and deleting all nodes whose place is after the root, what is the probability that there will be at most $\frac{D}{d-1}$ nodes reachable?

We will, in the next section, prove the following theorem:

Theorem 3.8. For any $\varepsilon > 0$, there exists $D_\varepsilon \in \mathbb{N}$ depending only on ε such that the following holds. Let T be a finite tree of degree at most $(d-1)(k-1)$ and $\sigma : V(T) \to \{1, ...r\}$ a labelling of the nodes of T such that on each path from the root to a leaf of T, σ is injective. Let $X_1, X_2, ..., X_r$ be real random variables distributed uniformly from $[0, 1]$ and mutually independently. Consider the experiment of drawing $X_1, .X_2, ..., X_r$ according to their distribution and then deleting all nodes u from T (along with the corresponding subtrees) for which $X_{\sigma(u)} \leq X_{\sigma(\text{root})}$.

Then the probability that the resultant tree T' contains more than D_ε nodes is

$$\Pr_{X_1,...,X_r}[|V(T')| > D_\varepsilon] \leq S_{(d,k)} + \varepsilon$$

where

$$S_{(d,k)} = \int_0^1 \frac{t^{\frac{1}{(d-1)(k-1)}} - t}{1-t}\,\mathrm{d}t$$

The next sections aim at proving Theorem 3.8, and we will proceed in the same four high-level steps as we did in the 3-SAT case:

1. We will prove a bound on a much simpler case: a full infinite tree of degree $(d-1)(k-1)$ from which every node is deleted independently with fixed probability p. This probability p corresponds, in the PPSZ analysis, to the place of a given variable in the permutation.

2. We will then argue that if the tree is not infinite but finite, the bound still holds.

3. We will show that the bound also holds if we introduce dependencies between the nodes.

4. We will finally consider the case where p is not fixed anymore but the place of the root is also random.

3.2.4 Random deletion in infinite and finite trees of degree $(d-1)(k-1)$

In this section, we consider the infinite rooted full tree T_∞ of degree $(d-1)(k-1)$. Each non-root of T_∞ is deleted (along with its subtree) independently from all other nodes with probability p; this yields the tree T'.

Let $q = \Pr[T' \text{ is finite}]$. For T' to be finite, each of the root's children must be either deleted (which happens with probability p), or the root of a finite tree, considering that we subject this infinite full tree of degree $(d-1)(k-1)$ to the same random experiment – and this happens with probability q. Hence, the following holds:

$$q = (p + (1-p)\cdot q)^{(d-1)(k-1)}.$$

Let $R_{d,k}(p)$ be the smallest $q \geq 0$ that satisfies this equation. This is well-defined since the equation is trivially satisfied for $q = 1$ and $(p + (1-p)\cdot q)^{(d-1)(k-1)} - q$ is polynomial in q.

This reasoning and this definition yield the following lemma:

Lemma 3.9. Let T_∞ be the infinite rooted full tree of degree $(d-1)(k-1)$. Consider the following random experiment: each non-root from T_∞ is deleted (along with its subtree) independently from all other nodes with probability p. Then the probability that the resultant tree T' is of finite size is

$$P[T' \text{ is finite}] \geq R_{d,k}(p)$$

where $R_{d,k}(p)$ (which we will characterize later) is the smallest q greater or equal than 0 satisfying the equation

$$q = (p + (1-p)\cdot q)^{(d-1)(k-1)}.$$

Observe that, since 1 is a solution of this equation, the probability is always valid.

We will prove the following lemma:

Lemma 3.10. Let T_∞ be the infinite rooted tree of degree $(d-1)(k-1)$. Consider the following random experiment: each non-root node from T_∞ is deleted (along with its subtree) independently from all other nodes with probability p. Then the probability that the resultant tree T' has height at most $H \geq 1$ converges as

$$\lim_{H\to\infty} \Pr[h(T') \leq H] = \Pr[T' \text{ is finite}].$$

The proof of this lemma is exactly the same as the proof of lemma Lemma 2.10 using the monotone convergence theorem; the only difference between these lemmata is the degree of the considered tree (binary tree vs. tree of degree $(d-1)(k-1)$).

Hence, we have

$$\lim_{H\to\infty} \Pr[h(T') \leq H] \geq R_{d,k}(p).$$

Instead of using a limit in this statement, we introduce a sequence of errors $\varepsilon_i(p)$. We state this in the following lemma:

Lemma 3.11. There exists a sequence $\varepsilon_1(p), \varepsilon_2(p), \ldots \in \mathbb{R}_0^+$ of numbers depending only on p, having $\varepsilon_H(p) \to 0$ for $H \to \infty$ such that the following holds. Let T_∞ be the infinite full tree of degree $(d-1)(k-1)$. Let $p \in [0,1]$ be a fixed number, and consider the following random experiment: each non-root node from T is deleted (along with its subtree) independently from all other nodes with probability p. Then the probability that the resultant tree T' has height at most $H \geq 1$ satisfies

$$\Pr[h(T') \leq H] \geq R_{d,k}(p) - \varepsilon_H(p).$$

Proof. Define, for all $H \geq 1$,

$$\varepsilon_H := \max\{R_{d,k}(p) - \Pr[h(T') \leq d], 0\}.$$

The we find that

$$\Pr[h(T') \leq H] \geq R_{d,k} - \varepsilon_H(p)$$

and, from Lemma 3.10,

$$\lim_{H\to\infty} \varepsilon_H(p) = 0$$

as required. $\square$

The arguments of Section 2.2.6 can be applied directly to the infinite tree of degree $(d-1)(k-1)$ to show that considering any finite (not necessarily full) trees of degree $(d-1)(k-1)$ does not hurt the bound from this section and that it can be applied to finite, not necessarily full trees as well.

3.2.5 From independent to dependent labels

We generalize Lemma 3.11 again to obtain the following lemma:

Lemma 3.12. Let $Z_1, Z_2, ..., Z_r \in \{0, 1\}$ be mutually independent binary random variables, each of which takes value 1 with probability p. Let T be any finite tree of degree $\leq (d-1)(k-1)$ with a labelling $\sigma : V(T)\backslash\{\text{root}\} \to \{1, ..., r\}$ of the non-root of T with indices that have the property that, on each path from the root to a leaf, σ is injective. Consider the experiment of drawing $Z_1, ..., Z_r$ according to their distribution and then deleting all nodes u from T (along with their subtrees) for which $Z_{\sigma(u)} = 1$. Call the resulting tree T'.

Juxtapose the experiment where in T, every non-root is deleted independently from all other nodes with probability p. Call the random tree resulting from this experiment T''. Then for any H,

$$\Pr[h(T') \leq H] \geq \Pr[h(T'') \leq H] \geq R_{d,k}(p) - \varepsilon_H(p).$$

Proof. The second inequality of the proof is a direct application of the finite version of Lemma 3.11.

Moreover, the statement is trivial if σ is globally injective, because in that case all the nodes have independent labels and we are in the previous case.

Now we suppose that none of the duplicates are in an ancestor-descendant relation (injectiveness on the paths from root to a leaf), and we show that the correlations arising from these duplicate labels cannot increase this probability. For this, we will use the FKG inequality, which we recall here and proven in Appendix A.3.

Theorem 2.13. Let $\mathcal{A} = \{A_1, A_2, ..., A_r\}$ be a collection of independent binary random variables and E_1 and E_2 events which are determined by $\mathcal{A}$ and monotonically increasing in $\mathcal{A}$. Then

$$\Pr[E_1 \wedge E_2] \geq \Pr[E_1] \cdot \Pr[E_2].$$

We prove Lemma 3.12 by induction on H. For $H = 0$, the statement is trivial (because both probabilities are 0, since the root is never deleted).

Let $H > 0$, and suppose that the statement holds for any depth strictly smaller than H. If the root of T has no child, the statement is trivial, since both probabilities are 1. If the root of T has only one child, then the statement is a direct consequence of the induction hypothesis, as the whole tree has height H iff the unique subtree of the root has height $H - 1$.

Now suppose that the root of T has z children $u_1, u_2, ...u_z$. For all $i \in \{1, ...z\}$, let T_i be the subtree of T rooted at u_i, T'_i the subtree of T' rooted at u_i and T'' be the subtree of T'' rooted at u_i (T_i' and T_i'' are empty trees if u_i is deleted). The injectiveness hypothesis on σ entails that no other node in T_i is labelled with $Z_{\sigma(u_i)}$, so whatever happens in the non-root nodes of T_i is independent of whether u_i itself is being deleted or not.

The event $\{h(T') \leq H\}$ can be defined as the conjunction of z similar events, one on each subtree: for all i, we define E_i as

$$E_i = \{Z_{\sigma(u_i)} = 1 \vee (Z_{\sigma(u_i)} = 0 \wedge h(T_i') \leq H - 1)\}$$

and we have that

$$\{h(t') \leq H\} = \bigwedge_{i=1}^{z} E_i.$$

Every E_i is determined by $\{Z_1, ..., Z_r\}$ and are monotonically increasing in those events (because if one of the Z_k goes from 0 to 1 then no E_i can go from 1 to 0). A conjunction of any number of these events will have the same property. Therefore, we can repeatedly apply the FKG inequality:

$$\begin{aligned}
\Pr[h(T') \leq H] &= \Pr[E_1 \wedge E_2 \wedge E_3 \wedge ... \wedge E_z] \\
&\geq \Pr[E_1] \cdot \Pr[E_2 \wedge E_3 \wedge ... \wedge E_z] \\
&\geq \Pr[E_1] \cdot \Pr[E_2] \cdot \Pr[E_3 \wedge ... \wedge E_z] \\
&\geq \Pr[E_1] \cdot \Pr[E_2] \cdot \Pr[E_3] \cdot ... \cdot \Pr[E_z]
\end{aligned}$$

Since T_i is independent from $Z_{\sigma(u_i)}$, we have that, for each E_i,

$$\begin{aligned}
\Pr[E_i] &= \Pr[Z_{\sigma(u_i)} = 1] + \Pr[Z_{\sigma(u_i)} = 0] \cdot \Pr[h(T_i') \leq H - 1] \\
&= p + (1-p) \Pr[h(T'_{\sigma(u_i)}) \leq H - 1]
\end{aligned}$$

and, by induction hypothesis,

$$\Pr[E_i] \geq p + (1-p) \Pr[h(T''i) \leq h - 1].$$

Combining all of these yields

$$\begin{aligned}
\Pr[h(T') \leq H] &\geq \prod_{i=1}^{z} (p + (1-p) \Pr[h(T_i'') \leq H - 1]) \\
&= \Pr[h(T'') \leq H]
\end{aligned}$$

□

3.2.6 Integrating over the rank of the root

We are now ready to prove Theorem 3.8, which we recall here:

Theorem 3.8. For any $\varepsilon > 0$, there exists $D_\varepsilon \in \mathbb{N}$ depending only on ε such that the following holds. Let T be a finite tree of degree at most $(d-1)(k-1)$ and $\sigma : V(T) \to \{1, ...r\}$ a labelling of the nodes of T such that on each path from the root to a leaf of T, σ is injective. Let $X_1, X_2, ..., X_r$ be real random variables distributed uniformly from $[0, 1]$ and mutually independently. Consider the experiment of drawing $X_1, .X_2, ..., X_r$ according to their distribution and then deleting all nodes u from T (along with the corresponding subtrees) for which $X_{\sigma(u)} \leq X_{\sigma(\text{root})}$.

Then the probability that the resultant tree T' contains more than D_ε nodes is

$$\Pr_{X_1,...,X_r}[|V(T')| > D_\varepsilon] \leq S_{(d,k)} + \varepsilon$$

where

$$S_{(d,k)} = \int_0^1 \frac{t^{\frac{1}{(d-1)(k-1)}} - t}{1-t} \,\mathrm{d}t$$

We fix ε, and we will, at the end of this proof, fix D_ε so that the statement holds.

Without loss of generality, suppose $\sigma(\text{root}) = X_r$. This value is independent of all the other values used because the root is part of all paths and σ is injective on each path, so X_r doesn't occur a second time as a label.

Now we condition on $X_r = \gamma$ for some fixed value $\gamma \in [0,1]$ and we consider, for $1 \le i \le r-1$, the binary random variables

$$Z_i = \begin{cases} 1 & \text{if } X_i < \gamma \\ 0 & \text{otherwise} \end{cases}.$$

The variables $\{Z_1, Z_2, ...Z_{r-1}\}$ are mutually independent binary random variables, each of which takes value 1 with probability exactly γ. This is exactly the situation of Lemma 3.12, and so we can use this result to conclude that

$$\Pr[h(T') \le H \mid X_r = \gamma] \ge R_{(d,k)}(\gamma) - \varepsilon_H(\gamma).$$

To convert this conditional into an unconditional probability, we would like to invoke the law of total probability, which, since Z_r is uniformly distributed, reads

$$\Pr[h[T'] \le H] = \int_0^1 \Pr[h(T') \le H \mid X_r = \gamma] \,\mathrm{d}\gamma,$$

and then apply our previous inequality involving $R_{(d,k)}(\gamma)$. This is significantly harder than in the 3-SAT case, though, because we do not have a closed expression for $R_{(d,k)}(\gamma)$. It is not even obvious that an integral on $R_{(d,k)}(\gamma)$ is properly defined. We will proceed as follows:

- we will introduce a function $q \mapsto S_{(d,k)}(q)$ that will allow us to characterize $R_{(d,k)}(\gamma)$
- we will also show that we can use the Riemann sums approximations given in Lemma 2.14 and proven in Appendix A.4,
- and we will relate the integrals of $R_{(d,k)}$ and $S_{(d,k)}$ in a way that allows us to conclude the proof of Theorem 3.8.

Definition 3.13. For $p \in [0,1)$, we define

$$S_{(d,k)}(q) = \frac{q^{\frac{1}{(d-1)(k-1)}} - q}{1-q}$$

and we let

$$S_{(d,k)}(1) = \frac{(d-1)(k-1)-1}{(d-1)(k-1)}.$$

We can now characterize $R_{(d,k)}$.

Lemma 3.14. For $p \in \left[\frac{(d-1)(k-1)-1}{(d-1)(k-1)}, 1\right]$, we have $R_{(d,k)}(p) = 1$.

For $p \in \left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$, $R_{(d,k)}(p)$ is the inverse of $S_{(d,k)}(q)$.

Proof. Recall that $R_{(d,k)}(p)$ is the smallest $q \geq 0$ that satisfies the equation

$$q = (p + (1-p) \cdot q)^{(d-1)(k-1)}.$$

Since $q = 1$ satisfies this equation, then $R_{d,k}(p) \leq 1$.
Let

$$f_{(d,k)}(p,q) = (p + (1-p)q)^{(d-1)(k-1)}.$$

$f_{(d,k)}$ is a growing function of p; for $p \in \left[\frac{(d-1)(k-1)-1}{(d-1)(k-1)}, 1\right]$ and $q < 1$, $f_{(d,k)}$ is minimized at $p = \frac{(d-1)(k-1)-1}{(d-1)(k-1)}$. Then we have:

$$\begin{array}{rcl} f_{(d,k)}(p,q) & = & (p + (1-p)q)^{(d-1)(k-1)} \\ & \geq & \left(\dfrac{(d-1)(k-1)-1}{(d-1)(k-1)} + 1 - \dfrac{1}{(d-1)(k-1)}q\right)^{(d-1)(k-1)} \\ & = & \left(1 - \dfrac{1}{(d-1)(k-1)}(1-q)\right)^{(d-1)(k-1)} \end{array}$$

Here we apply the following inequality, proven in Appendix B.1.2:

$$(1+p)^n > 1 + np,$$

which allows us to conclude that

$$f_{(d,k)}(p,q) > q.$$

Hence, for $p \in \left[\frac{(d-1)(k-1)-1}{(d-1)(k-1)}, 1\right]$, $R_{(d,k)}(p) = 1$ (because there is no root for $q < 1$, so the first root is 1), which proves the first part of the lemma.

For the second statement of the lemma, for $q \in [0,1)$, the following are equivalent:

$$\begin{array}{rcl} (p + (1-p)q)^{(d-1)(k-1)} & = & q \\ p + (1-p)q & = & q^{\frac{1}{(d-1)(k-1)}} \\ p(1-q) & = & q^{\frac{1}{(d-1)(k-1)}} - q \\ p & = & \dfrac{q^{\frac{1}{(d-1)-k-1)}} - q}{(1-q)} = S_{(d,k)}(q). \end{array}$$

Therefore, if $R_{(d,k)}(p) < 1$, then $S_{(d,k)}(q)$ is the inverse of $R_{(d,k)}(p)$. We now want to prove that for $p \in \left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right)$, $R_{(d,k)}(p) < 1$, which will conclude the proof of the second statement of the lemma.

First observe that, for $p = 0$, the smallest $q \geq 0$ satisfying the equation

$$q = (p + (1-p))q)^{(d-1)(k-1)} = q^{(d-1)(k-1)}$$

is $q = 0$; hence $R_{(d,k)}(0) = 0 < 1$, so we only have to prove the statement for $0 < p < \frac{(d-1)(k-1)-1}{(d-1)(k-1)}$.

By definition of $R_{d,k}(p)$, we can write

$$R_{(d,k)}(p) = (p + (1-p)R_{(d,k)}(p))^{(d-1)(k-1)}.$$

Let $\delta = 1 - R_{(d,k)}(p)$; this yields:

$$\begin{aligned} (p + (1-p)(1-\delta))^{(d-1)(k-1)} + \delta &= 1 \\ (1 - (1-p)\delta)^{(d-1)(k-1)} + \delta - 1 &= 0. \end{aligned}$$

Let $g(p,\delta) = (1-(1-p)\delta)^{(d-1)(k-1)} + \delta - 1$. For a fixed $0 \le p \le \frac{(d-1)(k-1)-1}{(d-1)(k-1)}$, g is a continuous function of δ. Moreover, we have that, still for a fixed p, $g'(p,\delta) = (d-1)(k-1)(p-1)(1-(1-p)\delta)^{(d-1)(k-1)-1} + 1$, which yields $g'(p,0) = (d-1)(k-1)(p-1)+1$. So for $d > 1$ or $k > 1$, $g'(p,0) < 0$. We also have that $g(p,0) = 0$, and that $g(p,1) > 0$ for $p > 0$. So for all $0 < p < \frac{(d-1)(k-1)-1}{(d-1)(k-1)}$ there exists a $\delta^* > 0$ such that $g(\delta^*, p) = 0$. So $1 - \delta^* < 1$ is a solution of

$$q = (p + (1-p))q)^{(d-1)(k-1)} = q^{(d-1)(k-1)}$$

which proves that $R_{(d,k)}(p) < 1$, and hence that $S_{(d,k)}(q)$ is the inverse of $R_{(d,k)}(p)$ on the interval $\left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right)$. □

Now we want, as we did in the 3-SAT case, to apply Lemma 2.14, which we recall here and which is proven in Appendix A.4:

Lemma 2.14. Let $\phi : [0,1] \to [0,1]$ be a continuous and monotonically non-decreasing function. Then for any $N \ge 1$,

$$\frac{1}{N}\sum_{i=0}^{N-1} \varphi\left(\frac{1}{N}\right) \le \int_0^1 \varphi(x)\,\mathrm{d}x \le \frac{1}{N}\sum_{i=0}^{N-1} \varphi\left(\frac{1}{N}\right) + \frac{1}{N}$$

As in the 3-SAT case, we have that $\Pr[h(T') \le H \mid X_r = \gamma]$ is continuous and non decreasing as a function of γ, so

$$\begin{aligned} \int_0^1 \Pr[h(T') \le H \mid X_r = \gamma]\,\mathrm{d}\gamma &\ge \sum_{i=0}^{N-1} \frac{1}{N} \Pr\left[h(T') \le H \mid X_r = \frac{i}{N}\right] && (3.1) \\ &\ge \sum_{i=0}^{N-1} \frac{1}{N} R_{(d,k)}\left(\frac{i}{N}\right) - \sum_{i=0}^{N-1} \frac{1}{N}\varepsilon_H\left(\frac{i}{n}\right). && (3.2) \end{aligned}$$

We will use the following fact to prove that $R_{d,k}$ is continuous and non decreasing:

$$\forall n \in \mathbb{N}, \forall a, b \in \mathbb{R}, a^n - b^n = (a-b)\sum_{i=0}^{n-1} a^i b^{n-1-i}.$$

The proof of this statement is given in Appendix B.1.3.

Using $a = 1$ and $b = q^{\frac{1}{(d-1)(k-1)}}$, we have, for $q \in [0,1)$:

$$\begin{aligned} S_{(d,k)}(q) &= \frac{q^{\frac{1}{(d-1)(k-1)}} - q}{1-q} \\ &= 1 - \frac{1 - q^{\frac{1}{(d-1)(k-1)}}}{1-q} \\ &= 1 - \frac{1 - q^{\frac{1}{(d-1)(k-1)}}}{\left(1 - q^{\frac{1}{(d-1)(k-1)}}\right)\sum_{i=0}^{(d-1)(k-1)-1} q^{\frac{i}{(d-1)(k-1)-1}}} \\ &= 1 - \frac{1}{\sum_{i=0}^{(d-1)(k-1)} q^{\frac{i}{(d-1)(k-1)-1}}}. \end{aligned}$$

From this it follows that $q \mapsto S_{(d,k)}(q)$ is a continuous, strictly increasing function which maps $[0,1]$ to $\left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$.

But then, we have also established that $R_{(d,k)}(p)$ is the inverse of $S_{(d,k)}(q)$ for $p \in \left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$, and 1 in $\left[\frac{(d-1)(k-1)-1}{(d-1)(k-1)}, 1\right]$. Consequently, on the interval $\left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$, $R_{(d,k)}$ is continuous and strictly increasing, mapping $\left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$ to $[0,1]$. Hence, $R_{(d,k)}$ is a continuous, non-decreasing function from $[0,1]$ to $[0,1]$. This means that we can apply the other side of the Riemann inequality in equation (3.2):

$$\begin{aligned}\int_0^1 \Pr[h(T') \leq H \mid X_r = \gamma] &\geq \sum_{i=0}^{N-1} \frac{1}{N} R_{(d,k)}\left(\frac{i}{N}\right) - \sum_{i=0}^{N-1} \frac{1}{N} \varepsilon_H \left(\frac{i}{n}\right) \\ &\geq \int_0^1 R_{(d,k)}(x)\,\mathrm{d}x - \frac{1}{N} - \sum_{i=0}^{N-1} \frac{1}{N} \varepsilon_H \left(\frac{i}{N}\right).\end{aligned}$$

By defining

$$\psi(N,H) = \frac{1}{N} + \sum_{i=0}^{N-1} \frac{1}{N} \varepsilon_H \left(\frac{i}{n}\right),$$

we get that, for an arbitrary n,

$$\Pr[h(T') \leq H] \geq \int_0^1 R_{(d,k)}(x)\,\mathrm{d}x - \psi(N,d).$$

Let us define

$$R_{(d,k)} = \int_0^1 R_{(d,k)}(x)\,\mathrm{d}x$$

and

$$S_{(d,k)} = \int_0^1 S_{(d,k)}(x)\,\mathrm{d}x.$$

We still need to relate $R_{(d,k)}$ to $S_{(d,k)}$ to be able to prove Theorem 3.8.

We need to prove the following lemma:

Lemma 3.15.

$$R_{(d,k)} = 1 - S_{(d,k)}.$$

Proof. We have:

$$1 - S_{(d,k)} = 1 - \int_0^1 S_{(d,k)}(t)\,\mathrm{d}t.$$

We do integration by parts and we get:

$$\begin{aligned}1 - S_{(d,k)} &= 1 - 1 \cdot S_{(d,k)}(1) + 0 \cdot S_{(d,k)} + \int_0^1 S'_{(d,k)}(t) t\,\mathrm{d}t \\ &= \frac{1}{(d-1)(k-1)} + \int_0^1 R_{(d,k)}(S_{(d,k)}(t)) S'_{(d,k)}(t)\,\mathrm{d}t\end{aligned}$$

We now use the following substitution rule, which is a classical result:

Lemma 3.16. Let $f : I \to \mathbb{R}$ be a continuous function and $\phi : [a, b] \to \mathbb{R}$ a continuous differentiable function where $\phi([a, b]) \subset I$. Then we have

$$\int_a^b f(\phi(t))\phi'(t)\mathrm{d}t = \int_{\phi(a)}^{\phi(b)} f(x)\,\mathrm{d}x.$$

By using the substitution $r = S_{(d,k)}(t)$ and since $\int_{\frac{(d-1)(k-1)-1}{(d-1)(k-1)}}^{1} R_{(d,k)}(r)\,\mathrm{d}r = \frac{1}{(d-1)(k-1)}$, this is

$$\int_{\frac{(d-1)(k-1)-1}{(d-1)(k-1)}}^{1} R_{(d,k)}(r)\,\mathrm{d}r + \int_0^{\frac{(d-1)(k-1)-1}{(d-1)(k-1)}} R_{(d,k)}(r)\,\mathrm{d}r = R_{(d,k)}.$$

□

The error term $\psi(N, H)$ depends only on N and H and not on the choice of T'. Therefore, given $\varepsilon > 0$ as in Theorem 3.8, we can simply pick $N = N(\varepsilon)$ which is large enough such that $\frac{1}{N} < \varepsilon/2$, and then pick $H = H(\varepsilon, N)$ large enough such that $\varepsilon_H(x) < \varepsilon/2$ for all the N points x where the function is evaluated, yielding that $\psi(N, d) \leq \varepsilon$ and thus all possible tree have a probability of having at most height H of at least $R_{(d,k)} - \varepsilon$. Since the trees are $(d-1)(k-1)$-ary trees, setting $D_\varepsilon > ((d-1)(k-1))^{H(\varepsilon, N(\varepsilon))+1}$ yields that

$$\Pr[|V(T') \leq D_\varepsilon] \geq R_{(d,k)} - \varepsilon = 1 - S_{(d,k)} - \varepsilon$$

and consequently that

$$\Pr[|V(T') > D_\varepsilon] \leq S_{(d,k)} + \varepsilon$$

as desired.

We use this theorem to finish the proof of Theorem 3.3. We introduce the notation

$$S_{(d,k)}^{(D)} = \sup_T \left(\Pr_{X_1,\ldots,X_r}[|V(T')| > D] \right)$$

where the supremum is over all choices of finite trees with labels T (as in Theorem 3.8) and T' is the random tree arising from t by conducting the experiment described in Theorem 3.8. In this language, the theorem states that

$$\lim_{D\to\infty} S_{(d,k)}^{(D)} \leq S_{(d,k)}.$$

The limit exists because $S_{(d,k)}^{(D)}$ is monotonic (it decreases as D increases) and bounded. Recall that

$$\begin{aligned}
\mathop{\mathbb{E}}_{\pi}[|\mathrm{Guessed}(F, \alpha_0, \alpha^*, \pi, D)|] &= n(\alpha_0) - \sum_{x\in\mathcal{U}(\alpha_0)} \Pr_\pi[x \in \mathrm{Forced}(F, \alpha_0, \alpha^*, \pi, D)] \\
&\leq n(\alpha_0) - \sum_{x\in\mathcal{U}(\alpha_0)} \Pr_\pi\left[\mathrm{Reachable}(T_x, \pi(x), \pi| \leq \frac{D}{d-1}\right]
\end{aligned}$$

By definition of $S_{(d,k)}^{(D)}$, this gives

$$\mathop{\mathbb{E}}_{\pi}[|\mathrm{Guessed}(F, \alpha_0, \alpha^*, \pi, D)|] \leq S_{(d,k)}^{(D/(d-1))} \cdot n$$

where, by Theorem 3.3, $S_{(d,k)}^{(D/(d-1))} \to \infty$. This can be achieved by selecting D to be some function that grows slowly in n, for instance $D = \log\log n$. This still allows us to examine all $G \subseteq F$ such that $|G| \leq D$ and check for D-implications in subexponential time, and thus to obtain Theorem 3.3.

3.3 Multiple satisfying assignments

In the previous section, we've proven Theorem 3.3, which deals with the case where F has a unique satisfying assignment. In general, though, we cannot rely on such a strong assumption, and we would like to prove Theorem 3.1, which we re-state here:

Theorem 3.1. For any satisfiable (d,k)-ClSP formula F on n variables V, PPSZ-WEAK$(F, V, \alpha_0, \log\log n)$ returns some satisfying assignment with probability $\Omega(d^{-S_{(d,k)}n-o(n)})$.

where

$$S_{(d,k)} = \int_0^1 \frac{t^{\frac{1}{(d-1)(k-1)}} - t}{1-t}\,\mathrm{d}t$$

The proof of this theorem follows again the structure established in Chapter 2: we will define a cost function, and relate it to the probability of success of PPSZ-WEAK.

As in the 3-SAT case, the analysis of the unique case does not go through in the general case because there is no guarantee that we can build critical clause trees for the formula. If a given variable has the same value in all the satisfying assignments, we call this variable "frozen". In that case, the argument goes through, and we can state the following lemma:

Lemma 3.17. Let F be a (d,k)-ClSP formula and x be a frozen variable. Let furthermore α be any satisfying assignment. Then $\Pr[x \in \text{Guessed}(F, \alpha_0, \alpha, \pi, D)] \leq S^{(D)}_{(d,k)}$.

3.3.1 Definition of a cost function

We proceed as in the 3-SAT case. We want to define a cost function that is bounded by $S^{(D)}_{(d,k)}$, and which never increases when doing a step of the PPSZ-WEAK algorithm. The intuition behind the cost definition is given in Section 2.3.3 for the 3-SAT case and can be easily translated to the ClSP case.

We partition the variables into three categories:

$$\text{vbl}(F) = \text{V}_{\text{fo}}(F) \mathbin{\dot\cup} \text{V}_{\text{fr}}(F) \mathbin{\dot\cup} \text{V}_{\text{nf}}(F)$$

where

- $\text{V}_{\text{nf}}(F)$ are the variables that are not frozen, i.e. the one that can be assigned at least two values while keeping the formula satisfiable. We define the shorthand $\text{V}_{\text{nf}}(\alpha_0) = \text{V}_{\text{nf}}(F^{[\alpha_0]})$.
- $\text{V}_{\text{fo}}(F)$ are the variables that are currently forced: they are frozen, and they follow from F via D-implication, that is there exists c such that for every $c' \neq c, c \in [d]$, $F \vDash_D (x \neq c')$. We define the shorthand $\text{V}_{\text{fo}}(\alpha_0) = \text{V}_{\text{fo}}(F^{[\alpha_0]})$.
- $\text{V}_{\text{fr}}(F)$ are the variables that are frozen, but not forced. We define the shorthand $\text{V}_{\text{fr}}(\alpha_0) = \text{V}_{\text{fr}}(F^{[\alpha_0]})$.

Now we can define the cost function as follows.

Definition 3.18. Let α_0 be a partial and α be a total assignment, and let x be any variable. We define the *cost of x when completing α_0 to α*, in writing $\text{cost}(\alpha_0, \alpha, x)$ as follows:

- If $x \neq \mathcal{U}(\alpha_0)$, then $\text{cost}(\alpha_0, \alpha, x) = 0$.
- If α_0 and α are incompatible, i.e. $\exists y \in \text{vbl}(\alpha_0), \alpha_0(y) \neq \alpha(y)$, then $\text{cost}(\alpha_0, \alpha, x) = 0$.
- If α does not satisfy F, then $\text{cost}(\alpha_0, \alpha, x) = 0$.

- Else:
  - If $x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)$, then $\mathrm{cost}(\alpha_0, \alpha, x) = 0$.
  - If $x \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$, then
$$\mathrm{cost}(\alpha_0, \alpha, x) = \Pr_{\pi}[x \in \mathrm{Guessed}(F, \alpha_0, \alpha, \pi, D)].$$
  - If $x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$, then $\mathrm{cost}(\alpha_0, \alpha, x) = S^{(D)}_{(d,k)}$.

Recall the definition of *likelihood* that we gave in the 3-SAT case:

Definition 2.17. Let $F^{[\alpha_0]}$ be satisfiable and let $\mathcal{S}_{\alpha_0}$ be the set of value assignments $l = \{x \mapsto b\}$ such that $x \in \mathcal{U}(\alpha_0)$ and $F^{[\alpha_0[l]]}$ is satisfiable.

We define the random process $\mathrm{AssignSL}(F, \alpha_0)$ that produces an assignment on $\mathrm{vbl}(F)$ as follows. Start with the assignment α_0, and repeat the following step until $\mathrm{vbl}(F^{[\alpha_0]}) = \emptyset$: Choose a value assignment $l \in \mathcal{S}_{\alpha_0}$ uniformly at random and add l to α_0. At the end, output α_0.

Let α be a total assignment on $\mathrm{vbl}(F)$. Then the *likelihood of completing* α_0 *to* α, in writing $\mathrm{lkhd}(\alpha_0, \alpha)$ is defined as the probability that $\mathrm{AssignSL}(F, \alpha_0)$ returns α. For completeness, if $F^{[\alpha_0]}$ is not satisfiable, we define $\mathrm{lkhd}(\alpha_0, \alpha) = 0$.

Then we define the cost of x when completing α_0 to any satisfying assignment as:

$$\mathrm{cost}(\alpha_0, x) = \sum_{\alpha \in \mathrm{sat}_V(F)} \mathrm{lkhd}(\alpha_0, \alpha) \cdot \mathrm{cost}(\alpha_0, \alpha, x) = \sum_{\alpha \in \mathrm{sat}_V(F)} \mathrm{wcost}(\alpha_0, \alpha, x)$$

and the total cost of completing α_0 to any satisfying assignment as

$$\mathrm{cost}(\alpha_0) = \sum_{x \in V} \sum_{\alpha \in \mathrm{sat}_V(F)} \mathrm{lkhd}(\alpha_0, \alpha) \cdot \mathrm{cost}(\alpha_0, \alpha, x)$$

We will prove the following lemma:

Lemma 3.19. Let α_0 be such that $F^{[\alpha_0]}$ is satisfiable. Then the overall probability of PPSZ-WEAK to output some satisfying assignment when starting in state α_0 is at least $d^{-\mathrm{cost}(\alpha_0)}$.

Observe moreover the following:

Observation 3.20. For any α_0, α, x we have $\mathrm{cost}(\alpha_0, \alpha, x) \leq S^{(D)}_{(d,k)}$. Furthermore, $\mathrm{cost}(\alpha_0) \leq S^{(D)}_{(d,k)} n(\alpha_0)$.

This observation follows directly from the definition of the cost and from Lemma 3.17. Proving Lemma 3.19 allows us to conclude directly that Theorem 3.1 holds as well.

3.3.2 Proving Lemma 3.19

Setup of the proof of Lemma 3.19

We first define $p(\alpha_0)$ as the probability that PPSZ-WEAK outputs some satisfying assignment when starting from state α_0.

We define the set $\mathcal{S}_{\alpha_0}$ as follows:

$$\mathcal{S}_{\alpha_0} := \{(x, c) \in \mathcal{U}(\alpha_0) \times [d] \mid F^{\alpha_0 \cup \{x \mapsto c\}]} \text{ is satisfiable}\}$$

and the set $\mathcal{S}_{\alpha_0}(x)$, for every $x \in \mathcal{U}_{\alpha_0}$, as

$$\mathcal{S}_{\alpha_0}(x) := \{(x,c) \in \{x\} \times [d] \mid F^{\alpha_0 \cup \{x \mapsto c\}} \text{ is satisfiable}\}.$$

The set $\mathcal{S}_{\alpha_0}$ represents all the choices that the PPSZ-WEAK run can do and still have a satisfying formula in the next step. The set $\mathcal{S}_{\alpha_0}$ contains at least $2|\mathrm{V}_{\mathrm{nf}}(\alpha_0)|+|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|+|\mathrm{V}_{\mathrm{fr}}(\alpha_0)|$ elements.

The proof of Lemma 3.19 is a proof by induction. We suppose that the claim holds for all α_0 that fix a larger number of variables. If α_0 is total, then the statement holds trivially, because the cost of α_0 is 0, and $p(\alpha_0) = 1$.

Let x and c be random variables: $x \in \mathcal{U}(\alpha_0)$, and c is the forced value by D-implication if $x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)$, and u.a.r. in $[d]$ otherwise.

We have:

$$p(\alpha_0) = \mathop{\mathbb{E}}_{x \in_{\text{u.a.r.}} \mathcal{U}(\alpha_0), c}[p(\alpha_0 \cup \{x \mapsto c\})].$$

(x, c) is in $\mathcal{S}_{\alpha_0}$ if:

- $x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$ and $(x,c) \in \mathcal{S}_{\alpha_0}(x)$;
- $x \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$, and c has the correct value: x is not forced, so c is chosen at random in $[d]$, but $\mathcal{S}_{\alpha_0}$ contains only one of these values because x is frozen;
- $x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)$, because x is forced, so c is not chosen at random but gets assigned its proper value, and $\mathcal{S}_{\alpha_0}$ contains (x, c).

Hence, the probability that $(x,c) \in \mathcal{S}_{\alpha_0}$ is

$$\begin{aligned}
&\Pr[(x,c) \in \mathcal{S}_{\alpha_0}] \\
&\quad = \Pr[x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0) \wedge (x,c) \in \mathcal{S}_{\alpha_0}(x)] + \Pr[x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)] + \frac{1}{d}\Pr[x \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)] \\
&\quad = \frac{\sum_{x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)} |\mathcal{S}_{\alpha_0}(x)|}{dn(\alpha_0)} + \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{n(\alpha_0)} + \frac{|\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{dn(\alpha_0)} \\
&\quad = \frac{\sum_{x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)} |\mathcal{S}_{\alpha_0}(x)| + d|\mathrm{V}_{\mathrm{fo}}(\alpha_0)| + |\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{dn(\alpha_0)} \\
&\quad = \frac{|\mathcal{S}_{\alpha_0}| + (d-1)|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{dn(\alpha_0)}
\end{aligned}$$

Hence, our working expression becomes:

$$p(\alpha_0) = \frac{|\mathcal{S}_{\alpha_0}| + (d-1)|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{dn(\alpha_0)} \mathop{\mathbb{E}}_{x \in_{\text{u.a.r.}} \mathcal{U}(\alpha_0), c}[p(\alpha_0 \cup \{x \mapsto c\}) \mid (x,c) \in \mathcal{S}_{\alpha_0}]. \tag{3.3}$$

Relating to uniform choice over $\mathcal{S}_{\alpha_0}$

As we did in the 3-SAT case, we relate the probability distribution over "x, then c" to the distribution uniformly at random over $\mathcal{S}_{\alpha_0}$.

Consider the event "a given (x, c) is chosen next in the process"; we look at the probabilities of said event within both distributions.

	$x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)$		$x \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$		$x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$	
	$(x,c) \notin \mathcal{S}_{\alpha_0}$	$(x,c) \in \mathcal{S}_{\alpha_0}$	$(x,c) \notin \mathcal{S}_{\alpha_0}$	$(x,c) \in \mathcal{S}_{\alpha_0}$	$(x,c) \notin \mathcal{S}_{\alpha_0}$	$(x,c) \in \mathcal{S}_{\alpha_0}$
x, then c	0	$\frac{1}{n(\alpha_0)}$	$\frac{1}{n(\alpha_0)} \cdot \frac{1}{d}$	$\frac{1}{n(\alpha_0)} \cdot \frac{1}{d}$	$\frac{1}{n(\alpha_0)} \cdot \frac{1}{d}$	$\frac{1}{n(\alpha_0)} \cdot \frac{1}{d}$
u.a.r in $\mathcal{S}_{\alpha_0}$	0	$\frac{1}{\lvert\mathcal{S}_{\alpha_0}\rvert}$	0	$\frac{1}{\lvert\mathcal{S}_{\alpha_0}\rvert}$	0	$\frac{1}{\lvert\mathcal{S}_{\alpha_0}\rvert}$

We do not care what happens when $(x,c) \notin \mathcal{S}_{\alpha_0}$, because then the corresponding term in the expectation will be 0 anyway. The weight of variables in $\mathrm{V}_{\mathrm{fo}}(\alpha_0)$ is d times the weight of the other variables; we define

$$w(x,c) := \begin{cases} d & \text{if } x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0) \\ 1 & \text{otherwise} \end{cases}$$

If we change the expectation of equation (3.3) to $(X,C) \in \mathcal{S}_{\alpha_0}$ u.a.r., each $(X,C) \in \mathcal{S}_{\alpha_0}$ now has weight $w(X,C)$. We still have to normalize the weight: the probability distribution must sum to one. Let W be the normalization factor:

$$\begin{aligned} \sum_{(x',c') \in \mathcal{S}_{\alpha_0}} \Pr[(x',c') \mapsto (X,C)] &= W \cdot \sum_{(X,C) \in \mathcal{S}_{\alpha_0}} w(X,C) \cdot \frac{1}{|\mathcal{S}_{\alpha_0}|} \\ &= W \cdot \frac{|\mathcal{S}_{\alpha_0}| + (d-1)|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} \end{aligned}$$

So the normalization factor is $W = \frac{|\mathcal{S}_{\alpha_0}|}{|\mathcal{S}_{\alpha_0}|+(d-1)|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}$, which in turns allows us to write:

$$\begin{aligned} p(\alpha_0) &= \frac{|\mathcal{S}_{\alpha_0}| + (d-1)|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{dn(\alpha_0)} \\ &\quad \cdot \mathop{\mathbb{E}}_{(X,C) \in_{\text{u.a.r.}} \mathcal{S}_{\alpha_0}} \left[\frac{|\mathcal{S}_{\alpha_0}|}{|\mathcal{S}_{\alpha_0}| + (d-1)|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|} w(X,C) p(\alpha_0 \cup \{X \mapsto C\}) \right] \\ &= \frac{|\mathcal{S}_{\alpha_0}|}{dn(\alpha_0)} \mathop{\mathbb{E}}_{(X,C) \in_{\text{u.a.r.}} \mathcal{S}_{\alpha_0}} [w(X,C) p(\alpha_0 \cup \{X \mapsto C\})] \end{aligned}$$

3.3.3 Using the induction hypothesis

We have now expressed $p(\alpha_0)$ in terms of an expectation that depends on the uniform choice over the possible value assignments of $F^{[\alpha_0]}$. From now one, we will omit in this proof the $(X,C) \in_{\text{u.a.r.}} \mathcal{S}_{\alpha_0}$ for ease of notation and reading.

We first use Jensen's inequality, so that we get expressions with which we can work. This yields:

$$\begin{aligned} p(\alpha_0) &\geq d^{\log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{dn(\alpha_0)}\right)} d^{\mathbb{E}[\log_d(w(X,C)p(\alpha_0 \cup \{X \mapsto C\})]} \\ &= d^{\log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{dn(\alpha_0)}\right)} d^{\mathbb{E}[\log_d(w(X,C)] - \mathbb{E}[-\log_d(p(\alpha_0 \cup \{X \mapsto C\}))]} \end{aligned}$$

by linearity of expectation.

Since $\log_d(1) = 0$ and $\log_d(d) = 1$, we have exactly that

$$\mathbb{E}[\log(w(X, C))] = \Pr[w(X, C) = d] = \Pr[x \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)] = \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|}.$$

Moreover, we can now apply the induction hypothesis, which states that

$$p(\alpha_0 \cup \{X \mapsto C\}) \geq d^{-\mathrm{cost}(\alpha_0 \cup \{X \mapsto C\})}$$

to state that

$$\mathbb{E}[-\log_d(p(\alpha_0 \cup \{X \mapsto C\}))] \leq \mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{X \mapsto C\})].$$

Putting everything together, we obtain

$$p(\alpha_0) \geq d^{\log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{dn(\alpha_0)}\right) + \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} - \mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{X \mapsto C\}]}. \tag{3.4}$$

Now the only thing that we need to evaluate is $\mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{X \mapsto C\})]$. To do this, we will first need to establish a few facts about cost and likelihood.

Lemma 3.21. Let α_0 and α be fixed and compatible. For any fixed variable $x \in \mathcal{U}(\alpha_0)$, if we set x according to α, then

(i) the likelihood of α can only increase, i.e.

$$\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \geq \mathrm{lkhd}(\alpha_0, \alpha),$$

with equality if x is frozen in $F^{[\alpha_0]}$.

(ii) the cost of a fixed variable $y \in V$ w.r.t. α can only decrease, i.e.

$$\mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y) \leq \mathrm{cost}(\alpha_0, \alpha, y).$$

When choosing $x \in \mathcal{U}(\alpha_0)$ uniformly at random and setting it according to α, then

(iii) the likelihood of α increases on average as

$$\mathbb{E}[\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)] = \left(\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\right) \mathrm{lkhd}(\alpha_0, \alpha).$$

(iv) the cost of a fixed, frozen, non-forced variable $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$ decreases on expectation as

$$\mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)] \leq \mathrm{cost}(\alpha_0, \alpha, y) - \frac{1}{n(\alpha_0)}.$$

Proof. (i) We prove the claim by induction over the size of α_0. The claim holds trivially if α_0 is a complete assignment. Otherwise, we have

$$\begin{aligned}
\mathrm{lkhd}(\alpha_0, \alpha) &= \mathop{\mathbb{E}}_{(x', c') \in \mathcal{S}_{\alpha_0}} \mathrm{lkhd}(\alpha_0 \cup \{x' \mapsto c'\}, \alpha) \\
&= \sum_{(x', c') \in \mathcal{S}_{\alpha_0}} \frac{1}{|\mathcal{S}_{\alpha_0}|} \mathrm{lkhd}(\alpha_0 \cup \{x' \mapsto c'\}, \alpha) \\
&= \sum_{x' \in \mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{S}_{\alpha_0}|} \mathrm{lkhd}(\alpha_0 \cup \{x' \mapsto \alpha(x')\}, \alpha) \\
&= \frac{1}{|\mathcal{S}_{\alpha_0}|} \left(\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) + \sum_{x' \in \mathcal{U}(\alpha_0) \setminus \{x\}} \mathrm{lkhd}(\alpha_0 \cup \{x' \mapsto \alpha(x')\}, \alpha) \right).
\end{aligned}$$

We apply the induction hypothesis and we get

$$\begin{aligned}
\text{lkhd}(\alpha_0, \alpha) \quad &\leq \quad \frac{1}{|\mathcal{S}_{\alpha_0}|}\Bigg(\text{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \\
&\quad + \sum_{x' \in \mathcal{U}(\alpha_0)\setminus\{x\}} \text{lkhd}(\alpha_0 \cup \{x' \mapsto \alpha(x')\} \cup \{x \mapsto \alpha(x)\}, \alpha)\Bigg) \\
&= \quad \frac{1}{|\mathcal{S}_{\alpha_0}|}\big(\text{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \\
&\quad + |\mathcal{S}_{\alpha_0 \cup \{x \mapsto \alpha(x)\}}|\text{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)\big)\,.
\end{aligned}$$

Now observe that, since $\mathcal{S}_{\alpha_0 \cup \{x \mapsto c\}} \subsetneq \mathcal{S}_{\alpha_0}$, then $|\mathcal{S}_{\alpha_0 \cup \{x \mapsto c\}}| \leq |\mathcal{S}_{\alpha_0}| - 1$, and this allows us to conclude that

$$\text{lkhd}(\alpha_0, \alpha) \leq \text{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}).$$

If x is forced, then by induction hypothesis we have the equality in the previous reasoning, and we have $|\mathcal{S}_{\alpha_0 \cup \{x \mapsto c\}}| = |\mathcal{S}_{\alpha_0}| - 1$, which proves the equality.

(ii) We consider the three cases of Definition 3.18. Note that if $x = y$, the statement holds trivially.

If $y \in \text{V}_{\text{nf}}(\alpha_0)$, then $\text{cost}(\alpha_0, \alpha, y) = S^{(D)}_{(d,k)}$. Since the cost of a variable is always less than $S^{(D)}_{(d,k)}$, the statement holds.

If $y \in \text{V}_{\text{fr}}(\alpha_0)$ or $y \in \text{V}_{\text{fo}}(\alpha_0)$, then $\text{cost}(\alpha_0, \alpha, y)$ is the probability that y is not forced in the remainder of the PPSZ-WEAK run. If we now fix another variable x to $\alpha(x)$, then this probability cannot increase, because adding a value assignment cannot "un-force" another variable. So $\text{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y) \leq \text{cost}(\alpha_0, \alpha, y)$.

(iii) We have

$$\begin{aligned}
\text{lkhd}(\alpha_0, \alpha) \quad &= \quad \sum_{(x,c) \in \mathcal{S}_{\alpha_0}} \frac{1}{|\mathcal{S}_{\alpha_0}|}\text{lkhd}(\alpha_0 \cup \{x \mapsto c\}, \alpha) \\
&= \quad \sum_{x \in \mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{S}_{\alpha_0}|}\text{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \\
&= \quad \frac{n(\alpha_0)}{|\mathcal{S}_{\alpha_0}|} \mathop{\mathbb{E}}_{x \in_{\text{u.a.r}} \mathcal{U}(\alpha_0)}[\text{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)],
\end{aligned}$$

which proves the statement.

(iv) The statement tells us that the probability that y is guessed is reduced by $1/n(\alpha_0)$ after one step. This is because with probability $1/n(\alpha_0)$, y comes next in π and is guessed now (since it is not forced now). This $1/n(\alpha_0)$ is counted in $\text{cost}(\alpha_0, \alpha, y)$ but not in $\mathbb{E}[\text{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)]$.

□

Evaluating $\mathbb{E}[\textbf{cost}(\alpha_0 \cup \{X \mapsto C\})]$

In this section, we will prove the following lemma:

Lemma 3.22. If $(X, C) \in \mathcal{S}_{\alpha_0}$ is selected uniformly at random, then

$$\mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{X \mapsto C\})] \leq \mathrm{cost}(\alpha_0) - \frac{|\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} - \frac{S^{(D)}_{(d,k)}}{|\mathcal{S}_{\alpha_0}|} \sum_{x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)} |\mathcal{S}_{\alpha_0}(x)|.$$

To prove this lemma, we will need the following auxilliary lemma:

Lemma 3.23. Let α be a fixed satisfying assignment and $\alpha_0 \subsetneq \alpha$. Let $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$ be a fixed frozen variable. If we select $x \in \mathcal{U}(\alpha_0)$ uniformly at random and assign it according to α, then

$$\mathbb{E}[\mathrm{wcost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)] \leq \mathrm{wcost}(\alpha_0, \alpha, y) \cdot \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} - \frac{\mathrm{lkhd}(\alpha_0, \alpha)}{n(\alpha_0)}$$

Proof. We use, as in the 3-SAT case, Lemma 2.24 (proven in Appendix B.2.1), which we recall here:

Lemma 2.24. Let $A, B \in \mathbb{R}$ be random variables and $a, b, \bar{a}, \bar{b} \in \mathbb{R}$ fixed numbers such that $A \geq a$ and $B \leq b$ always, and $\mathbb{E}[A] = \bar{a}$ and $\mathbb{E}[B] = \bar{b}$. Then

$$\mathbb{E}[A \cdot B] \leq a\bar{b} + b\bar{a} - ab.$$

We have that $\mathrm{wcost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y) = \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \cdot \mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)$, so we define $A = \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)$ and $B = \mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)$. Using Lemma 3.21, we define $a = \mathrm{lkhd}(\alpha_0, \alpha)$, $\bar{a} = \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} \mathrm{lkhd}(\alpha_0, \alpha)$, $b = \mathrm{cost}(\alpha_0, \alpha, y)$ and $\bar{b} \leq \mathrm{cost}(\alpha_0, \alpha, y) - \frac{1}{n(\alpha_0)}$. Applying our correlation inequality, we deduce that, for $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$, when selecting $x \in \mathcal{U}(\alpha_0)$ u.a.r. and assigning it according to α, we have:

$$\begin{aligned}
& \mathbb{E}[\mathrm{wcost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)] \\
\leq\ & \mathrm{lkhd}(\alpha_0, \alpha) \left(\mathrm{cost}(\alpha_0, \alpha, y) - \frac{1}{n(\alpha_0)} \right) + \mathrm{cost}(\alpha_0, \alpha, y) \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} \mathrm{lkhd}(\alpha_0, \alpha) \\
& - \mathrm{lkhd}(\alpha_0) \mathrm{cost}(\alpha_0, \alpha, y) \\
=\ & \mathrm{wcost}(\alpha_0, \alpha, y) \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} - \frac{\mathrm{lkhd}(\alpha_0, \alpha)}{n(\alpha_0)}.
\end{aligned}$$

□

We can now prove Lemma 3.22.

Proof of Lemma 3.22. We analyze the cost decrease contribution of the different types of variables separately. Each variable forced in state α_0 contributes zero cost both before and after the step.

For a non-frozen variable $y \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$, note that $\mathcal{S}_{\alpha_0}$ contains $|\mathcal{S}_{\alpha_0}(y)|$ pairs containing y, in contrats to the other types of variables of which $\mathcal{S}_{\alpha_0}$ features only one pair each. This means that with probability $|\mathcal{S}_{\alpha_0}(y)|/|\mathcal{S}_{\alpha_0}|$, a pair featuring y is selected. In that case, the cost contribution of y drops from $S^{(D)}_{(d,k)}$ in all assignments. No matter what happens to these costs otherwise (they certainly cannot increase by definition), the non-frozen variables hence contribute the last term of the claimed inequality.

Now consider the frozen variables. If we fix some satisfying and α_0-compatible assignment $\alpha \in [d]^V$, and condition the experiment on the event that $\alpha(X) = C$, then X becomes uniformly at random among $\mathcal{U}(\alpha_0)$ because $\mathcal{S}_{\alpha_0}$ contains exactly one pair (x', c') per variable $x' \in \mathcal{U}(\alpha_0)$

such that $\alpha(x') = c'$. Now we can apply directly Lemma 3.23 to find that, conditioning on $\alpha(X) = C$, the cost of each frozen variable drops on average as

$$\mathbb{E}[\mathrm{wcost}(\alpha_0 \cup \{X \mapsto C\}, \alpha, y) \mid \alpha(X) = C] \leq \mathrm{wcost}(\alpha_0, \alpha, y) \cdot \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} - \frac{\mathrm{lkhd}(\alpha_0, \alpha)}{n(\alpha_0)}.$$

The condition itself is satisfied with probability $n(\alpha_0)/|\mathcal{S}_{\alpha_0}|$. If it does not apply, then the cost contribution of α drops to zero altogether. Therefore, the unconditional change can be obtained by multiplying the right-hand side by $n(\alpha_0)/|\mathcal{S}_{\alpha_0}|$, which then yields

$$\mathbb{E}[\mathrm{wcost}(\alpha_0 \cup \{X \mapsto C\}, \alpha, y)] \leq \mathrm{wcost}(\alpha_0, \alpha, y) - \frac{\mathrm{lkhd}(\alpha_0, \alpha)}{|\mathcal{S}_{\alpha_0}|}.$$

If we sum over all assignments $\alpha \in [d]^V$, the claim follows. $\square$

Putting everything together

We are back to the main track of the proof of Lemma 3.19, and we can now introduce the result from Lemma 3.22 into equation (3.4). We obtain:

$$p(\alpha_0) \geq d^{\log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{dn(\alpha_0)}\right) + \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} - \mathrm{cost}(\alpha_0) + \frac{|\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} - \frac{S^{(D)}_{d,k}}{|\mathcal{S}_{\alpha_0}|}\sum_{x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)} |\mathcal{S}_{\alpha_0}(x)|}.$$

Since we want to show that $p(\alpha_0) \geq d^{-\mathrm{cost}(\alpha_0)}$, we need to show:

$$\log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{dn(\alpha_0)}\right) + \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} + \frac{|\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} - \frac{S^{(D)}_{(d,k)}}{|\mathcal{S}_{\alpha_0}|} \sum_{x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)} |\mathcal{S}_{\alpha_0}(x)| \geq 0.$$

To refer easily to this quantity, we define

$$L := \log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{dn(\alpha_0)}\right) + \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} + \frac{|\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} - \frac{S^{(D)}_{(d,k)}}{|\mathcal{S}_{\alpha_0}|} \sum_{x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)} |\mathcal{S}_{\alpha_0}(x)|$$

We write

$$\begin{aligned}
\log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{dn(\alpha_0)}\right) &= -1 + \log\left(\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\right) \\
&= -1 + \log_d\left(\frac{|\mathrm{V}_{\mathrm{fr}}(\alpha_0)| + |\mathrm{V}_{\mathrm{fo}}(\alpha_0)| + \sum_{x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)} |\mathcal{S}_{\alpha_0}(x)|}{n(\alpha_0)}\right) \\
&= -1 + \log_d\left(1 + \frac{\sum_{x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)| - 1)}{n(\alpha_0)}\right)
\end{aligned}$$

We now use an inequality about logarithms, namely that for $x \geq 0$,

$$\log_d(1 + x) \geq \log_d(e)\frac{x}{1 + x}.$$

This inequality is proven in Appendix B.1.1. This yields:

$$\begin{aligned}
\log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{dn(\alpha_0)}\right) &\geq -1+\log_d(e)\frac{\frac{\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)|-1)}{n(\alpha_0)}}{1+\frac{\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)|-1)}{n(\alpha_0)}} \\
&= -1+\log_d(e)\frac{\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)|-1)}{n(\alpha_0)+\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)|-1)} \\
&= -1+\log_d(e)\frac{\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)|-1)}{|\mathcal{S}_{\alpha_0}|}
\end{aligned}$$

Moreover, we have that

$$\begin{aligned}
&-1+\frac{|\mathrm{V_{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|}+\frac{|\mathrm{V_{fr}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|}+\frac{S^{(D)}_{(d,k)}}{|\mathcal{S}_{\alpha_0}|}\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)| \\
=\ & -\frac{|\mathcal{S}_{\alpha_0}|}{|\mathcal{S}_{\alpha_0}|}+\frac{|\mathcal{S}_{\alpha_0}|-\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)|}{|\mathcal{S}_{\alpha_0}|}+\frac{S^{(D)}_{(d,k)}}{|\mathcal{S}_{\alpha_0}|}\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)| \\
=\ & -\frac{1}{|\mathcal{S}_{\alpha_0}|}\left(\left(1-S^{(D)}_{(d,k)}\right)\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)|\right)
\end{aligned}$$

Therefore, we get

$$\begin{aligned}
L &\geq \log_d(e)\frac{\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)|-1)}{|\mathcal{S}_{\alpha_0}|}-\frac{1}{|\mathcal{S}_{\alpha_0}|}\left(\left(1-S^{(D)}_{(d,k)}\right)\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)|\right) \\
&= \frac{1}{|\mathcal{S}_{\alpha_0}|}\left(\log_d(e)\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)-1)-\left(1-S^{(D)}_{(d,k)}\right)\sum_{x\in \mathrm{V_{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)|\right) \\
&= \frac{1}{|\mathcal{S}_{\alpha_0}|}\sum_{x\in \mathrm{V_{nf}}(\alpha 0)}\left(\log_d(e)(|\mathcal{S}_{\alpha_0}(x)|-1)-(1-S^{(D)}_{(d,k)})|\mathcal{S}_{\alpha_0}(x)|\right)
\end{aligned}$$

Proving that every term of the sum is positive proves that $L \geq 0$. For $x \in \mathrm{V_{nf}}(\alpha_0)$, we know that $|\mathcal{S}_{\alpha_0}(x)| \geq 2$. So the following holds:

$$\begin{aligned}
&\log_d(e)(|\mathcal{S}_{\alpha_0}(x)|-1)-\left(1-S^{(D)}_{(d,k)}\right)|\mathcal{S}_{\alpha_0}(x)| \geq 0 \\
\Leftrightarrow\ &\log_d(e)-\left(1-S^{(D)}_{(d,k)}\right)\frac{|\mathcal{S}_{\alpha_0}(x)|}{|\mathcal{S}_{\alpha_0}(x)|-1} \geq 0
\end{aligned}$$

where $\frac{|\mathcal{S}_{\alpha_0}(x)|}{|\mathcal{S}_{\alpha_0}(x)|-1} \leq 2$. It follows that $L \geq 0$ if $\log_d(e) - 2\left(1-S^{(D)}_{(d,k)}\right) \geq 0$.

For $d = 2$ and $k = 3$, we already observed, at the end of Section 2.3, that $\log e + 2S^{(D)}_{(2,3)} > 2$, which proves the statement for $d = 2$ and $k = 3$. Moreover, from the definition of $S_{(d,k)}$, it can be easily seen that $S_{(d,k)}$ increases for larger k, so $S_{(d,k)} \geq S_{(d,3)}$.

Suppose now that $d > 2$. We prove the following lemma:

Lemma 3.24.

$$S_{(d,3)} \geq 1-\left(\frac{\pi^2}{12}\right)\frac{1}{d-1}$$

Proof. For ease of notation, we define $m = 2(d-1)$. Recall that $S_{(d,3)} = \int_0^1 S_{d,3}(t)\,\mathrm{d}t$. By definition of $S_{(d,k)}(t)$, we have:

$$\begin{aligned}
S_{(d,3)}(t) &= \frac{t^{\frac{1}{(d-1)(3-1)}} - t}{1-t} = \frac{t^{1/m} - t}{1-t} \\
&= 1 - \frac{1 - t^{1/m}}{1-t} \\
&= 1 - (1 - t^{1/m}) \sum_{i=0}^{\infty} t^i \\
&= 1 - \sum_{i=0}^{\infty} (t^i - t^{i+1/m})
\end{aligned}$$

Thus,

$$S_{(d,3)} = 1 - \int_0^1 \sum_{i=0}^{\infty} (t^i - t^{i+1/m})\,\mathrm{d}t.$$

We want to apply the monotone convergence theorem to exchange the integral and the sum. For this we define $f_k(t) = \sum_{i=0}^{k} (t^i - t^{i+1/m})$. Then the sequence $f_1, f_2...$ is pointwise non-decreasing because $t^i - t^{i+1/m} \geq 0$ for $t \in [0,1]$ and the pointwise limit of the sequence (f_n) is, for every t,

$$\lim_{k\to\infty} f_k(t) = 1 - S_{(d,3)}(t).$$

Then

$$\begin{aligned}
\int_0^1 \sum_{i=0}^{\infty} (t^i - t^{i+1/m})\,\mathrm{d}t &= \int_0^1 \lim_{k\to\infty} f_k(t)\,\mathrm{d}t \\
&= \lim_{k\to\infty} \int_0^1 f_k(t)\,\mathrm{d}t \\
&= \lim_{k\to\infty} \int_0^1 \sum_{i=0}^{k} (t^i - t^{i+1/m})\,\mathrm{d}t \\
&= \lim_{k\to\infty} \sum_{i=0}^{k} \int_0^1 t^i - t^{i+1/m}\,\mathrm{d}t \\
&= \sum_{i=0}^{\infty} \int_0^1 t^i - t^{i+1/m}\,\mathrm{d}t
\end{aligned}$$

as desired.

We then continue the computation:

$$
\begin{aligned}
S_{(d,3)} = 1 - \sum_{i=0}^{\infty} \int_0^1 (t^i - t^{i+1/m})\,\mathrm{d}t &= 1 - \sum_{i=0}^{\infty} \left(\frac{1}{i+1} - \frac{1}{i+1+\frac{1}{m}} \right) \\
&= 1 - \sum_{i=1}^{\infty} \left(\frac{1}{i} - \frac{1}{i+\frac{1}{m}} \right) \\
&= 1 - \sum_{i=1}^{\infty} \frac{\frac{1}{m}}{i\left(i+\frac{1}{m}\right)} \\
&= 1 - \frac{1}{m} \sum_{i=1}^{\infty} \frac{1}{i\left(i+\frac{1}{m}\right)} \\
&\geq 1 - \frac{1}{m} \sum_{i=1}^{\infty} \frac{1}{i^2} \\
&= 1 - \frac{1}{m}\frac{\pi^2}{6} = 1 - \left(\frac{\pi^2}{12}\right)\frac{1}{d-1}
\end{aligned}
$$

□

Back to the main computation, we have shown that $L \geq 0$ if $\log_d(e) - 2(1 - S_{(d,3)}) \geq 0$, which yields that $L \geq 0$ if the following condition holds:

$$
\begin{aligned}
& \log_d(e) - \left(\frac{\pi^2}{6}\right)\frac{1}{d-1} \geq 0 \\
\Leftrightarrow \quad & \frac{1}{\ln d} \geq \left(\frac{\pi^2}{6}\right)\frac{1}{d-1} \\
\Leftrightarrow \quad & \left(\frac{\pi^2}{6}\right)\ln d \leq d-1 \\
\Leftrightarrow \quad & d - 1 - \left(\frac{\pi^2}{6}\right)\ln d \geq 0
\end{aligned}
$$

For $d = 3$, we have that $2 - \left(\frac{\pi^2}{6}\right)\ln 3 \geq 0$. Furthermore, $d - 1 - \left(\frac{\pi^2}{6}\right)\ln d$ is monotonically increasing as the derivative is $1 - \left(\frac{\pi^2}{6}\right)\frac{1}{d} > 0$ for $d \geq 3$.

This concludes the proof of Lemma 3.19, which in turn concludes the proof of Theorem 3.1.

3.4 Summary

To prove Theorem 3.1, we went through the following steps:

1. We first proved Theorem 3.3, which makes the analysis of the algorithm easier by considering a single satisfying assignment.
2. We used the bound obtained in this analysis to define a cost function for the formula. By definition, this cost is bounded by the value that we need to establish the same bound on the runtime of PPSZ-WEAK as in the unique case.

3. We proved that the probability that PPSZ-WEAK returns any satisfying assignment is at least $2^{-\mathrm{cost}(F)}$, where $\mathrm{cost}(F)$ is the cost of returning a satisfying assignment when starting with an empty assignment. For this, we had to prove that, for any α_0 such that $F^{[\alpha_0]}$ is satisfiable,

$$L = \log_d \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} + \frac{|\mathrm{V}_{\mathrm{fo}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} + \frac{|\mathrm{V}_{\mathrm{fr}}(\alpha_0)|}{|\mathcal{S}_{\alpha_0}|} - \frac{S_{(d,k)}^{(D)}}{|\mathcal{S}_{\alpha_0}|} \sum_{x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)} |\mathcal{S}_{\alpha_0}(x)|$$

is non-negative, which required us to bound the different quantities of this expression and to apply some numerical values to the result.

Chapter 4

A strong version of PPSZ for (d, k)-ClSP

We've observed in Chapter 3 that there was an obvious improvement to the weak PPSZ algorithm that is presented there: instead of picking the value of the non-forced variable uniformly at random from the complete domain $[d]$, we can pick it from the set of the non-forbidden variables.

This chapter deals with the analysis of this variant of the algorithm. The results presented in this chapter have been developed by Szedlák [7] for the unique case and Millius [3] for the general case.

4.1 The algorithm

In this chapter, we will consider the following algorithm for (d, k)-ClSP formulas.

```
PPSZ-WEAK($F, V, \alpha_0, D$)
    $\pi \leftarrow$ a permutation of $\mathcal{U}(\alpha_0)$ chosen u.a.r.
    return PPSZ-STRONG($F, V, \alpha_0, D, \pi$)

PPSZ-STRONG($F, V, \alpha_0, D, \pi$)
    $\alpha_{\text{prog}} \leftarrow \alpha_0$
    for $i \leftarrow 1$ to $n(\alpha_0)$
        do
            $x \leftarrow x_{\pi_i}$
            $\mathcal{G}(x, \pi) \leftarrow \{c \in [d] \mid F^{[\alpha_{\text{prog}}]} \not\models_D (x \neq c)\}$
            if $\mathcal{G}(x, \pi) = \emptyset$
                then return 'failure'
            $c \leftarrow_{\text{u.a.r.}} \mathcal{G}(x, \pi)$
            $\alpha_{\text{prog}} \leftarrow \alpha_{\text{prog}} \cup \{x \mapsto c\}$
    if $\alpha_{\text{prog}}$ satisfies $F$
        then return $\alpha_{\text{prog}}$
        else return 'failure'
```

We will follow the same structure for this algorithm as we did in the previous chapter: we will first prove a theorem concerning the unique case. Ideally, we would like to be able to prove that the bound that we prove for the unique case is also attained for the general case; although we believe that it is indeed the case, it is not yet fully proven.

The theorem for the unique case is stated as follows:

Theorem 4.1. For any (d, k)-ClSP formula F on n variables V which has a unique satisfying assignment, PPSZ-STRONG$(F, V, \alpha_0, \log\log n)$ returns this assignment with probability $\Omega(d^{-\sum_{x\in\mathcal{U}(\alpha_0)}\mathbb{E}[\log_d(|\mathcal{A}(x,\pi)|]n})$, where

$$\mathbb{E}[\log_d(|\mathcal{A}(x,\pi)|)] \leq \sum_{j=0}^{d-1}\binom{d-1}{j}\log_d(1+j)$$
$$\cdot \int_0^1 s^{d-1-j}(1-s)^j \cdot \frac{(1-s^{d-1})(\frac{1}{k-1}s^{\frac{1}{k-1}-1}-(d-1)s^{d-2})+(s^{\frac{1}{k-1}}-s^{d-1})(d-1)s^{(d-2)}}{(1-s^{d-1})^2}\,\mathrm{d}s.$$

4.2 Unique satisfying assignment

In this section, we consider that the formula F has a unique satisfying assignment α^*. Without loss of generality, we suppose that this satisfying assignment is the all-d assignment, i.e. $\alpha^* = (d, d, ..., d)$. As in the previous chapters, we will use α_0 to denote a partial assignment, $\mathcal{U}(\alpha_0)$ to denote the variables that are not fixed by α_0 and $n(\alpha_0) = |\mathcal{U}(\alpha_0)|$. These definitions will be mostly used in the multiple satisfying assignment case. In this section, we can assume that $\alpha_0 = \emptyset$, $\mathcal{U}(\alpha_0) = V$ and $n(\alpha_0) = n$.

4.2.1 The sets $\mathcal{A}(x, \pi, D)$

We will use the sets $\mathcal{A}(x, \pi, D)$ to analyze the success probability of PPSZ-STRONG.

Definition 4.2. For every variable x, every total assignment α, every partial assignment α_0 compatible with α, every integer D, and every permutation π, we define the sets $\mathcal{A}(x, \alpha_0, \alpha, \pi, D)$ as follows:

$$\mathcal{A}(x, \alpha_0, \alpha, \pi, D) = \{c \in [d] \mid F^{[\alpha_0\cup\beta]} \nvDash_D (x \neq c)\}$$

where β is the partial assignment corresponding to the restriction of α to the variables that come before x in π.

In the unique satisfying assignment case, we are only interested in $\alpha = \alpha^*$, the satisfying assignment. We also consider that $\alpha_0 = \emptyset$. In what follows, we will denote $\mathcal{A}(x, \emptyset, \alpha^*, \pi, D)$ by $\mathcal{A}(x, \pi, D)$.

Observe that PPSZ-STRONG(F, V, α_0, D, π) returns the satisfying assignment α^* if and only if it picks the right value $\alpha^*(x)$ when processing x. It does so with probability $\frac{1}{|\mathcal{A}(x,\pi,D)|}$, always. For a fixed permutation π, we obtain

$$\Pr[\text{PPSZ-STRONG}(F, V, \alpha_0, D, \pi) \text{ returns } \alpha^*] = \prod_{x\in\mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{A}(x, \pi, D)|}$$

and, consequently, for π chosen uniformly at random,

$$\Pr[\text{PPSZ-STRONG}(F, V, \alpha_0, D) \text{ returns } \alpha^*] = \mathop{\mathbb{E}}_{\pi}\left[\prod_{x\in\mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{A}(x, \pi, D)|}\right].$$

We apply Jensen's inequality (see Appendix A.1) with the convex function $x \mapsto d^{-x}$ and we obtain

$$\Pr[\text{PPSZ-STRONG}(F, V, \alpha_0, D) \text{ returns } \alpha^*] \geq d^{-\sum_{x \in \mathcal{U}(\alpha_0)} \mathbb{E}_\pi[\log_d |\mathcal{A}(x,\pi,D)|]}.$$

The rest of this section will be devoted to the evaluation of $\mathbb{E}_\pi[\log_d |\mathcal{A}(x, \pi, D)|]$.

4.2.2 Building critical clause trees

The core reasoning here is the same than in the 3-SAT case and than in the PPSZ-WEAK case. However, in the current case, we are not only interested in knowing whether or not a variable is forced or not, but we want to know the "degree of freedom" of non-forced variables. In the PPSZ-WEAK case, this would correspond to the number of directions in which the tree extends "too much" to conclude that the variable is forced.

The way we analyse this is to build not one tree per variable x, but $(d-1)$ trees for each variable. The root of each of these trees has children in only one direction, corresponding to a given value assignment for x. The rest of the tree is built as in the PPSZ-WEAK analysis. This will allow us to evaluate the number of trees for which the analogue of the set Reachable$(T_x, \pi(x), \pi)$ is too large, and hence bound the size of the set $\mathcal{A}(x, \pi, D)$ for each x.

We consider a formula F that has a unique assignment α^*, and let α^* be, without loss of generality, the all-d assignment.

We construct a collection of trees $\{T_x^c\}_{x \in \mathcal{U}(\alpha_0), c \in [d-1]}$; for a given variable x, the set of trees $\{T_x^c\}_{c \in [d-1]}$ is called the *set of critical clause trees of x*.

Recall the following definition:

Definition 3.5. We call T a *rooted tree with children into j directions* if the following holds. T is a tree with a designated root, root(T). The children of a vertex v are partitioned into j groups which we denote Children$_1(v)$, Children$_2(v)$,..., Children$_j(v)$. Each child belongs to exactly one group, i.e. Children$_i(v)$ and Children$_k(v)$ are disjoint sets whenever $i \neq k$.

We build, for every $x \in \mathcal{U}(\alpha_0)$ and for every $c \in [d-1]$, a rooted tree into $(d-1)$ directions T_x^c, where every node $u \in V(T)$ is labelled both with a variable $x \in V$, which we denote by var-label(u), and a set of clauses $\mathcal{C} \in F^{[\alpha_0]}$, denoted by clause-label(u). Here is how T_x^c is built for a fixed $x \in \mathcal{U}(\alpha_0)$ and a fixed $c \in [d-1]$:

1. Start with T_x^c consisting of a single root. This root has variable label x, and and empty clause label.

2. Let C be a constraint that is not satisfied by $\alpha^*[x \mapsto c]$. Let C be the clause label of the root. For each literal $(y \neq d) \in C$, add a child to the root, which is var-labeled with the variable the literal is over. We define these nodes to be in Children$_c(x)$.

3. As long as there is a leaf $u \in V(T)$ that has an empty clause label, do the following:

    (a) Define $W :=$ {var-label(v) | $v \in V(T)$ is an ancestor of u in T}, where *ancestor* includes u itself and the root.

    (b) Let the path from the root to be such that var-label$(y_0) = x$, $y_1 \in$ Children$_{\ell_1}(x)$, $y_2 \in$ Children$_{\ell_2}(y_1), ..., y_m \in$ Children$_{\ell_m}(y_{m-1})$, $u \in$ Children$_{\ell_{m+1}}(y_m)$. Then $\ell_1, ...,$ ℓ_{m+1} are well-defined, and uniquely defined. We define the partial assignment μ_0 as follows:

$$\mu_0 : \text{vbl}(F) \to \{0,1\}, \begin{cases} \mu_0(\text{var-label}(y_i)) = \ell_{i+1} & \forall i \in \{0, ..., m\} \\ \mu_0(z) = \alpha^*(z) = d & \forall z \notin \{\text{var-label}(y_0), ..., \\ & \text{var-label}(y_m), \text{var-label}(u)\} \end{cases}$$

(c) For $j = 1$ to $d-1$, we define $\mu_j = \mu_0[\text{var-label}(v) \mapsto j]$. For each j, let C_j be a constraint that is not satisfied by μ_j. Since μ_j is not compatible with α^* and α^* is the unique satisfying assignment, such a clause exists. Add C_j to clause-label(u).

(d) For each literal $(y \neq d)$ in C_j, add a node to Children$_j(v)$, which is var-labeled with the variable the literal is over.

We denote the resulting tree by T_x^c. Note that T_x^c is not unique for a given x and a given c. We still consider the collection $\{T_x^c\}_{x \in \mathcal{U}(\alpha_0), c \in [d-1]}$ to be fixed from now on.

The root has at most $(k-1)$ children; any other node has at most $(d-1)(k-1)$ children: for each non-root node, we add, for each $j = 1$ to $d-1$, a group of at most $(k-1)$ children: it cannot happen that a clause has k literals $(y \neq d)$ because the all-d is a satisfying assignment.

Suppose that v is an ancestor of u and var-label$(v) = y$. Since $\mu_0(y) \neq d$, it cannot happen that clause-label(u) contains a clause that has literal $(y \neq d)$ (otherwise this clause would be satisfied). Therefore:

Observation 4.3. In T_x^c, no node has the same var-label as one of its proper ancestors.

This also implies that the height of the tree cannot exceed n, and thus the process terminates, making T_x^c well-defined.

4.2.3 Critical trees and $\mathcal{A}(x, \pi, D)$

As in section Section 2.2.4, we now consider π as a placement; the values $\pi(x)$, called place of x, are chosen independently at random from $[0, 1]$ for each $x \in \mathcal{U}(\alpha_0)$.

Let $\gamma \in [0, 1]$ and T_x^c be the critical clause tree for some variable T_x^c. We can use the same definition as in Section 2.2.4 for *reachable nodes*: a node $u \in T_x^c$ is reachable at time γ w.r.t. π if there exists a path $v_0, v_1, ..., v_m$ such that v_0 is th root of the tree, $v_m = u$ and $\pi(v_i) \geq \gamma$ for all $i \leq i \leq m$. Let us denote Reachable(T_x^c, γ, π) the set of all nodes in T_x reachable at time γ w.r.t. π. Observe that this set is independent of the place of x.

We prove the following lemma, which is essentially the same argument as in the PPSZ-WEAK case:

Lemma 4.4. *If we have* $|\text{Reachable}(T_x^c, \pi(x), \pi)| \leq D$, *then it holds has well that* $F \vDash_{D(d-1)} (x \neq c)$.

Proof. Let α' be the restriction of $\alpha^* = (d, d, ..., d)$ to the variables $y \in (\alpha_0)$ with $\pi(y) < \pi(x)$. Let $G = \text{clause-label}(\text{Reachable}(T_x^c, \pi(x), \pi))$, i.e. the subformula of F consisting of the union of all the clause labels sets of reachable nodes in T_x^c. Since by hypothesis $|\text{Reachable}(T_x^c, \pi(x), \pi)| \leq D$, each clause label contains at most $(d-1)$ clauses, then clearly $|G| \leq (d-1)D$.

Suppose that G does not imply $(x \neq c)$. Then we can fix an assignment $\nu : V \to [d]$ which is compatible with $\alpha_0 \cup \alpha'$, which has $\nu(x) = c$ and which satisfies G. Choose a maximal path in T_x^c, starting at the root, x, and containing only nodes v such that $\nu(\text{var-label}(v)) \neq d$. Since $\nu(x) = c$, this path is non-empty. Let u be its endpoint. Since ν is compatible with α^* on all the variables before x, it must be that var-label(u) is either x itself or after x, and hence $\pi(\text{var-label}(u)) \geq \pi(x)$ and so u is reachable, by definition. For all children z of u, we have that $\nu(\text{var-label}(z)) = d$ (because the path is maximal); all ancestors y of u are such that $\nu(\text{var-label}(y)) \neq d$. By definition of T_x^c, all the clauses of the clause-label of u are unsatisfied by ν and, since clause-label$(u) \subseteq G$, this is a contradiction. □

It follows immediately from Lemma 4.4 that, over the uniform choice of π, we have

$$\Pr_\pi[c \notin \mathcal{A}(x, \pi, D)] \geq \Pr_\pi\left[|\text{Reachable}(T_x^c, \pi(x), \pi)| \leq \frac{D}{d-1}\right].$$

For $1 \leq c \leq d-1$, we define

$$Y_{(c)} = \begin{cases} 1 & \text{if } |\text{Reachable}(T_x^c, \pi(x), \pi)| > \frac{D}{d-1} \\ 0 & \text{otherwise} \end{cases}$$

For $c = d$, we define $Y_{(d)} = 1$. Finally, we define $Y = \sum_{c=1}^{d} Y_{(c)}$. We get the following lemma:

Lemma 4.5.

$$|\mathcal{A}(x, \pi, D)| \leq Y$$

Proof. For $c = 1, ..., d-1$, we know that $Y_{(c)}$ is 0 if and only if $|\text{Reachable}(T_x^c, \pi(x), \pi)| \leq \frac{D}{d-1}$, in which case $c \notin \mathcal{A}(x, \pi, D)$. Hence, if $c \in \mathcal{A}(x, \pi)$, then $Y_{(c)} = 1$. Moreover, for $c = d$, we always have $d \in \mathcal{A}(x, \pi, D)$. $\square$

This reduces the problem to a probabilistic calculation on trees: when sorting the nodes of a set of trees according to a random permutation (caveat: some nodes have the same label and are prescribed to get assigned the same place) and deleting all nodes whose place is after the root, what is the expected number of trees that have more than $\frac{D}{d-1}$ nodes reachable?

The next sections aim at proving Theorem 4.1 and we will proceed in the following six high-level steps:

1. We will first consider the random deletion in one infinite tree where all the nodes are deleted independently with probability p.

2. We will then use this bound to get a bound for the case of random deletion in several trees where all the nodes are deleted, again independently, with the same probability p.

3. We argue that if the trees are not infinite but not finite, the bound still holds.

4. We argue that the bound also holds if we introduce dependencies between the labelling of different trees, while keeping the labelling of individual trees independent.

5. We argue that it still holds if we introduce dependencies between the nodes of a single tree.

6. We finally consider the case where p is not fixed anymore but the place of the root is also random.

4.2.4 Random deletion in one infinite tree

In this section, we consider a set of $(d-1)$ infinite rooted full trees T_∞^c of root degree $(k-1)$ and of non-root degree $(d-1)(k-1)$. Each non-root node of the trees T_∞^c is deleted (along with its subtree) independently of all other nodes with probability p; this yields the trees $T^{c\prime}$.

For a given $T^{c\prime}$ to be finite, each of the root's children must be either deleted (which happens with probability p), or the root of a finite tree of degree $(d-1)(k-1)$, that we got by doing random deletions in an infinite tree of degree $(d-1)(k-1)$, whose root we do not delete.

We showed in Section 3.2.4 that the probability for that event is larger than $R_{(d,k)}(p)$, where $R_{(d,k)}(p)$ is the smallest q satisfying the equation

$$q = (p + (1-p) \cdot q)^{(d-1)(k-1)}.$$

Lemma 4.6. Let T_∞ be an infinite rooted full tree of root degree $(k-1)$ and of non-root degree $(d-1)(k-1)$. Consider the following random experiment: each non-root from T_∞ is deleted (along with its subtree) independently from all other nodes with probability p. Then the probability that the resultant tree T' is of finite size is:

$$\Pr[T' \text{ is finite}] \geq \tilde{R}_{(d,k)}(p)$$

where

$$\tilde{R}_{(d,k)}(p) = (p + (1-p)R_{(d,k)}(p))^{(k-1)}.$$

Proof. Let y_i be the i^{th} child of the root of T_∞ and let Q_i be the subtree rooted at y_i. Note that what happens to y_i is independent of what happens in Q_i. Then we have:

$$\begin{aligned}
\Pr[T' \text{ is finite}] &= \Pr\left[\bigwedge_{i=1}^{k-1} y_i \text{ is deleted } \vee (y_i \text{ is not deleted } \wedge Q_i \text{ is finite})\right] \\
&\geq \prod_{i=1}^{k-1} p + (1-p)R_{(d,k)}(p) \\
&= (p + (1-p)R_{(d,k)}(p))^{(k-1)} \\
&= \tilde{R}_{(d,k)}(p)
\end{aligned}$$

□

4.2.5 From one tree to $(d-1)$ trees

So far, we have considered the deletion of nodes of a single tree. However, contrarily to the 3-SAT case and to the weak case, we are not interested, here, simply in whether the tree has a bounded height, but we want to know how many of these trees have a bounded height. Back to the original problem, we are interested in the quantity $\mathbb{E}_\pi[\log(|\mathcal{A}(x,\pi,D)|)]$ where, as Lemma 4.5 implies, $|\mathcal{A}(x,\pi,D)| \leq 1 + \sum_{c=1}^{d-1} Y_{(c)}$.

In the following lemma, we suppose for the moment that the nodes are all deleted independently of each other. We will then prove, in the following sections, that having non-independent deletions can only help.

Lemma 4.7. Let $\{T_\infty^c\}_{(c\in[d-1])}$ be a set of infinite rooted full trees of root degree $(k-1)$ and of non-root degree $(d-1)(k-1)$. Consider the following random experiment: each non-root from every T_∞^c is deleted (along with its subtree) independently from all other nodes with probability p. We denote the resulting trees $\{T^{c\prime}\}$. Then, if we define $\mathcal{B}$ as the set of infinite trees in $\{T^{c\prime}\}$ after this random experiment, we have

$$\mathbb{E}[\log_d(1+|\mathcal{B}|)] \leq \sum_{j=1}^{d-1} \log_d(1+j)\tilde{R}_{(d,k)}(p)^{d-1-j}(1-\tilde{R}_{(d,k)}(p))^j.$$

Proof. We will use the following lemma, which we call the general lemma on expectation, and which we prove in Appendix B.4. We need this lemma here because we have only a bound on the probability that an arbitrary tree is finite: we know that it is greater than $\tilde{R}_{(d,k)}(p)$. Since we are interested in the size of the set of the infinite trees, we will have to consider events with probabilities less than $1-\tilde{R}_{(d,k)}(p)$ as well, and these bounds do not allow us to conclude directly, hence the use of the lemma.

Lemma 4.8. Let A be a finite set and let $f : 2^A \to \mathbb{R}$ be a function that satisfies $f(B') \leq f(B)$, $\forall B' \subseteq B \subseteq A$.

Let $q' : A \to [0, 1]$ and $q : A \to [0, 1]$ be two functions that satisfy $q'(a) \leq q(a), \forall a \in A$.

At first, let $C_{q'}$ and C_q be the empty set. Now consider the two following random experiments:

1. $\forall a \in A$, choose $a \in C_{q'}$ with probability $q'(a)$.

2. $\forall a \in A$, choose $a \in C_q$ with probability $q(a)$.

Note that the events are independent.

Then we have $\mathbb{E}[f(C_{q'})] \leq \mathbb{E}[f(C_q)]$.

Now we consider A as the set of all possible infinite trees $T^{c'}$ when proceeding to the experiment of Lemma 4.6. We have proven in Lemma 4.6 that an arbitrary tree was finite with probability greater than $\tilde{R}_{(d,k)}(p)$, and so an arbitrary tree is infinite with probability less than $1 - \tilde{R}_{(d,k)}(p)$.

To apply Lemma 4.8, we define q' as the probability that an arbitrary tree is infinite, and $q = 1 - \tilde{R}_{(d,k)}(p)$ – so $C_{q'} = \mathcal{B}$, i.e the set of infinite trees $T^{c'}$ when proceeding to the experiment from Lemma 4.6. We also define $f = \log_d(1 + |B|)$ for all $B \subseteq A$. Observe that $f(B') \leq f(B)$ for all $B' \subseteq B \subseteq A$.

From Lemma 4.8, we then have

$$\mathbb{E}[\log_d(1 + |\mathcal{B}|)] = \mathbb{E}[f(C_{q'})] \leq \mathbb{E}[f(C_q)] = \mathbb{E}[\log_d(1 + |C_q|)].$$

But then we have:

$$\begin{aligned}
\mathbb{E}[\log_d(1 + |C_q|)] &= \sum_{C_q \subseteq A} \Pr[\forall a \in C_q, a \text{ is picked } \wedge \forall a \notin C_q, a \text{ is not picked}] \log_d(1 + |C_q|) \\
&= \sum_{j=1}^{d-1} \log_d(1 + j) \binom{d-1}{j} \Pr[a \text{ is picked}]^j \Pr[a \text{ is not picked}]^{d-1-j} \\
&= \sum_{i=1}^{d-1} \log_d(1 + j) \binom{d-1}{j} (1 - \tilde{R}_{(d,k)}(p))^j \tilde{R}_{(d,k)}(p)^{d-1-j},
\end{aligned}$$

which concludes the proof. □

4.2.6 From infinite to finite trees

We prove the following lemma:

Lemma 4.9. Let T^c_∞ be a set of infinite rooted full trees of root degree $(k - 1)$ and of non-root degree $(d - 1)(k - 1)$. Consider the following random experiment: each non-root from every T^c_∞ is deleted (along with its subtree) independently from all other nodes with probability p. The resulting trees are denoted as $T^{c'}$. Then, if we define $\mathcal{B}_H$ as the set of trees in $\{T^c_\infty\}$ that have a height greater than H after this experiment and $\mathcal{B}$ as the set of infinite trees after this experiment, we have

$$\lim_{H \to \infty} \mathbb{E}[\log_d(1 + |\mathcal{B}_H|)] = \mathbb{E}[\log_d(1 + |\mathcal{B}|)].$$

Proof. We have

$$\mathbb{E}[\log_d(1 + |\mathcal{B}_H|)] = \sum_{j=0}^{d-1} \Pr[|\mathcal{B}_H| = j] \log_d(1 + j),$$

so

$$\lim_{H\to\infty} \mathbb{E}[\log_d(1+|\mathcal{B}_H|)] = \sum_{j=0}^{d-1} \log_d(1+j) \lim_{H\to\infty} \Pr[|\mathcal{B}_H| = j].$$

Moreover,

$$\Pr[|\mathcal{B}_H| = j] = \Pr[|\mathcal{B}_H \leq j|] - \Pr[|\mathcal{B}_H| \leq j-1]$$

and, since the events $|\mathcal{B}_H| \leq j$ and $\Pr[|\mathcal{B}_H| \leq j-1]$ are monotonically increasing in H, we can apply the monotone convergence theorem (see Appendix A.2). We have that $\lim_{H\to\infty} |\mathcal{B}_H| = |\mathcal{B}|$, so

$$\lim_{H\to\infty} \Pr[|\mathcal{B}_H| \leq j] = \Pr[|\mathcal{B}| \leq j]$$

by the monotone convergence theorem, and similarly

$$\lim_{H\to\infty} \Pr[|\mathcal{B}_H| \leq j-1] = \Pr[|\mathcal{B}| \leq j-1].$$

Putting everything together, we have

$$\begin{aligned}
\mathbb{E}[\log_d(1+|\mathcal{B}_H|)] &= \sum_{j=0}^{d-1} \log_d(1+j)(\Pr[|\mathcal{B}| \leq j] - \Pr[|\mathcal{B}| \leq j-1]) \\
&= \sum_{j=0}^{d-1} \log_d(1+j) \Pr[|\mathcal{B}| = j] \\
&= \mathbb{E}[\log_d((1+|\mathcal{B}|)]
\end{aligned}$$

as desired. □

We extend Lemma 4.7 to get the following lemma:

Lemma 4.10. There exists a sequence $\varepsilon_1(p), \varepsilon_2(p), ..., \in \mathbb{R}_0^+$ of numbers depending only on p, having $\varepsilon_H(p) \to 0$ for $H \to \infty$ such that the following holds. Let T_∞^c be a set of infinite rooted full trees of root degree $(k-1)$ and of non-root degree $(d-1)(k-1)$. Consider the following random experiment: each non-root from every T_∞^c is deleted (along with its subtree) independently from all other nodes with probability p. The resulting trees are denoted as $T^{c\prime}$. Then let $\mathcal{B}_H$ be the set of trees $T^{c\prime}$ that have a height larger than H and $\mathcal{B}$ be the set of trees $T^{c\prime}$ that have an infinite height: we have

$$\mathbb{E}[\log_d(1+|\mathcal{B}_H|)] \leq \mathbb{E}[\log_d(1+|\mathcal{B}|)] + \varepsilon_H(p).$$

Proof. Define, for all $H \geq 1$,

$$\varepsilon_H(p) := \max\{\mathbb{E}[\log_d(1+|\mathcal{B}_H|)] - \mathbb{E}[\log_d(1+|\mathcal{B}|)], 0\}$$

Then we find that

$$\mathbb{E}[\log_d(1+|\mathcal{B}_H|)] \leq \mathbb{E}[\log_d(1+|\mathcal{B}|)] + \varepsilon_H(p)$$

and, from Lemma 4.9,

$$\lim_{H\to\infty} \varepsilon_H(p) = 0.$$

□

Let us now consider any family of $d-1$ finite (not necessary full) trees of root degree $(k-1)$ and of non-root degree $(k-1)(d-1)$, $\{T^c\}$. Consider the following random experiment: each non-root node from every tree of $\{T^c\}$ is deleted (along with its subtree independently from all other nodes with probability p.

We can couple this experiment to a random experiment conducted on the family $\{T^c_\infty\}$: the family $\{T^c\}$ is embeddable into $\{T^c_\infty\}$ with every root from a tree from $\{T^c\}$ coinciding to a root from a tree from $\{T^c_\infty\}$. If we delete every node from the trees from $\{T^c_\infty\}$ with probability p, the same experiment is taking place on $\{T^c\}$. Let $\{T^{c\prime\prime}\}$ be the trees resulting from the deletions in $\{T^c_\infty\}$ and $\{T^{c\prime}\}$ be the trees resulting from the deletions in $\{T^c\}$. Since the trees $T^{c\prime}$ are subtrees of the trees $T^{c\prime\prime}$, we have $h(T^{c\prime}) \leq h(T^{c\prime\prime})$. If $\mathcal{B}''_H$ is the set of trees from $\{T^{c\prime\prime}\}$ that have a height greater than H and $\mathcal{B}'_H$ the set of trees from $\{T^{c\prime}\}$ that have a height greater than H, we have

$$\mathbb{E}[\log(1 + |\mathcal{B}'_H|)] \leq \mathbb{E}[\log(1 + |\mathcal{B}''_H|)].$$

So any upper bound that we can establish on full, infinite trees, can also be applied to non-full, finite trees.

4.2.7 From independent to dependent labels

As we did in the 3-SAT case and in the weak PPSZ case, we now want to remove the assumption that the labels of the trees are independent. We will prove the following lemma:

Lemma 4.11. Let $Z_1, Z_2, ..., Z_r \in \{0, 1\}$ be mutually independent binary random variables, each of which takes value 1 with probability p. Let T^c be a set of $(d-1)$ finite (and not necessarily full) trees of root degree $k-1$ and of non-root degree $(d-1)(k-1)$ with a labelling $\sigma : V(T^c)\backslash\{\text{root}\} \to \{1, ..., r\}$ of the non-roots of the trees T^c with indices that have the property that, on each path from a root to a leaf, σ is injective. Consider the experiment of drawing Z_1, ..., Z_r according to their distribution and then deleting all nodes u from all trees T^c (along with their subtrees) for which $Z_{\sigma(u)} = 1$. Call the resulting trees U^c, and call $\mathcal{J}_H$ the set of trees in $\{U^c\}$ that have a height larger than H.

Juxtapose the experiment where in T^c, the labelling (which we call σ') has the additional property that no two labels of two different trees are the same. Such a labelling can, for instance, be achieved by replacing any label in σ that happens more than once by a set of labels that allow to cover every corresponding nodes exactly once. Call the trees resulting from this experiment $T^{c\prime}$, and call $\mathcal{K}_H$ the set of trees in $\{T^{c\prime}\}$ that have a height larger than H.

Finally, juxtapose the experiment where in T^c, every non-root node is deleted independently from all other nodes with probability p. Call the random trees resulting from this experiment $T^{c\prime\prime}$, and call $\mathcal{L}_H$ the set of trees in $\{T^{c\prime\prime}\}$ that have a height larger than H.

Then we have, for any H:

$$\mathbb{E}[\log(1 + |\mathcal{J}_H|)] \leq \mathbb{E}[\log(1 + |\mathcal{K}_H|)] \leq \mathbb{E}[\log(1 + |\mathcal{L}_H|)].$$

Proof. We prove both inequalities one after the other. For the first inequality, we will use the following lemma, which we prove in Appendix B.3:

Lemma 4.12. Let X, T, T', U, V be finite vectors of real random variables, all independent, taking only finitely many values. Suppose T and T' are over $\mathbb{R}^n$ and have the same distribution. Let $f, g : \mathbb{R}^n \to \mathbb{R}$ be monotonically increasing functions, and let $h : \mathbb{R} \to \mathbb{R}$ be a concave function. Then

$$\mathbb{E}[h(f(X, T, U) + g(X, T, V))] \leq \mathbb{E}[h(f(X, T, U) + g(X, T', V))].$$

In this lemma, we can think of the left hand side and the right hand side as two different, but related, random experiments. The functions f and g are fed with random input. The vector X represents shared random input (between both functions), U and V are individual random input (for each function), and T and T' are random input that is shared in the first experiment between f and g, but made independent in the second experiment.

Let us define the following variables:

$$X_c = \begin{cases} 1 & \text{if } h(U^c) > H \\ 0 & \text{otherwise} \end{cases}$$

and

$$Y_c = \begin{cases} 1 & \text{if } h(T^{c\prime}) > H \\ 0 & \text{otherwise} \end{cases}$$

Observe that for all $c \in [d-1]$, $X_c = 1$ if and only if in U^c there exists a node u at depth $H+1$ such that all nodes from the root to u are labeled 1. Hence:

$$\begin{aligned} \forall c \in [d-1], X_c &= \bigvee_{\substack{p \text{ path in } U^c \text{ from root} \\ \text{to node at depth } H+1}} \bigwedge_{u \text{ on } p} Z_{\sigma(u)} \\ &= \bigvee_{\substack{p \text{ path in } T^{c\prime} \text{ from root} \\ \text{to node at depth } H+1}} \bigwedge_{u \text{ on } p} Z_{\sigma(u)} \end{aligned}$$

where the equality inequality comes from the fact that U^c and T_{prime} only differ from their labelling, while their structure stays the same. Similarly,

$$\forall c \in [d-1], Y_c = \bigvee_{\substack{p \text{ path in } T^{c\prime} \text{ from root} \\ \text{to node at depth } H+1}} \bigwedge_{u \text{ on } p} Z_{\sigma'(u)}$$

This implies that the functions X_c and Y_c are monotonically increasing functions depending on the Z_i. Moreover, we have, by definition of $\mathcal{K}_H$ and $\mathcal{J}_H$, that $|\mathcal{K}_H| = \sum_{i=1}^{d-1} Y_c$ and $|\mathcal{J}_H| = \sum_{i=1}^{d-1} X_c$. Now we can apply Lemma 4.12 repeatedly. We start with

$$\begin{aligned} \mathbb{E}[\log_d(1 + |\mathcal{J}_H|)] &= \mathbb{E}\left[\log_d\left(1 + \sum_{c=1}^{d-1} Y_c\right)\right] \\ &= \mathbb{E}\left[\log_d\left(1 + Y_1 + \sum_{c=2}^{d-1} Y_c\right)\right] \\ &= \mathbb{E}\left[\log_d\left(1 + \bigvee_{p \text{ path in } T^{1\prime}} \bigwedge_{u \text{ on } p} Z_{\sigma'(u)} + \sum_{c=2}^{d-1} \bigvee_{p \text{ path in } T^{c\prime}} \bigwedge_{u \text{ on } p} Z_{\sigma'(u)}\right)\right] \end{aligned}$$

Applying the notations of Lemma 4.12, in the first step, X is empty; T is the vector of the Z_i that appear both in Y_1 and in some other Y_c; T' is an independent vector of values of the same size as T; U is the vector of the Z_i that only appear in Y_1, V is the vector of the Z_i that do not appear in Y_1. f and g are functions of the Z_i such that $f(X, T, V) = \sum_{c=2}^{d-1} Y_c$ and $g(X, T, U) = Y_1$. We have observed that the Y_c were monotonically increasing functions of the Z_i; f and g are consequently monotonically increasing functions as well. The function h is $h : t \mapsto \log(1+t)$, which is concave. h is not a function $\mathbb{R} \mapsto \mathbb{R}$, but we can define a concave function defined on all $\mathbb{R}$ that agrees with h on $[1, \infty)$.

By Lemma 4.12, we can then write

$$\begin{aligned}
&\mathbb{E}\left[\log_d\left(1+\bigvee_{p\text{ path in }T^{1\prime}}\bigwedge_{u\text{ on }p}Z_{\sigma'(u)}+\sum_{c=2}^{d-1}\bigvee_{p\text{ path in }T^{c\prime}}\bigwedge_{u\text{ on }p}Z_{\sigma'(u)}\right)\right]\\
\leq\ &\mathbb{E}\left[\log_d\left(1+\bigvee_{p\text{ path in }T^{1\prime}}\bigwedge_{u\text{ on }p}Z_{\sigma(u)}+\sum_{c=2}^{d-1}\bigvee_{p\text{ path in }T^{c\prime}}\bigwedge_{u\text{ on }p}Z_{\sigma'(u)}\right)\right]
\end{aligned}$$

We then apply Lemma 4.12 repeatedly. For an arbitrary step t, X is the vector of the Z_i that appear in X_1 to X_{t-1}; T is the vector of the Z_i that appear both in Y_t and in some other Y_c, $c > t$; T' is an independent vector of values of the same size as T; U is the vector of the Z_i that only appear in Y_t, V is the vector of the Z_i that do not appear in Y_t. f and g are functions of the Z_i such that $f(X, T, V) = \sum_{c=t+1}^{d-1} Y_c$ and $g(X, T, U) = Y_t + \sum_{c=1}^{t-1} X_c$. Then we write:

$$\begin{aligned}
&\mathbb{E}\left[\log_d\left(1+\sum_{c=1}^{t-1}X_c+\bigvee_{p\text{ path in }T^{t\prime}}\bigwedge_{u\text{ on }p}Z_{\sigma'(u)}+\sum_{c=t+1}^{d-1}\bigvee_{p\text{ path in }T^{c\prime}}\bigwedge_{u\text{ on }p}Z_{\sigma'(u)}\right)\right]\\
\leq\ &\mathbb{E}\left[\log_d\left(1+\sum_{c=1}^{t-1}X_c+\bigvee_{p\text{ path in }T^{t\prime}}\bigwedge_{u\text{ on }p}Z_{\sigma(u)}+\sum_{c=2}^{d-1}\bigvee_{p\text{ path in }T^{c\prime}}\bigwedge_{u\text{ on }p}Z_{\sigma'(u)}\right)\right]
\end{aligned}$$

At the end of this procedure, we get

$$\mathbb{E}[\log_d(1+|\mathcal{J}_H|)]=\mathbb{E}[\log_d(1+\sum_{c=1}^{d-1}Y_c)]\leq\mathbb{E}[\log(1+\sum_{c=1}^{d-1}X_c]=\mathbb{E}[\log_d(1+|\mathcal{K}_H|)]$$

which proves the first inequality.

The second inequality is a direct consequence of Lemma 4.8, or general lemma on expectation (see Appendix B.4).

We pick as A the set of all trees T^c, and as f the function $f : B \mapsto \mathbb{E}[\log_d(1 + |B|)]$. We pick the trees from T^c with the probabilities $\Pr[T^c \in \mathcal{K}_H]$ and $\Pr[T^c \in \mathcal{L}_H]$. If we prove that $\Pr[T^c \in \mathcal{K}_H] \leq \Pr[T^c \in \mathcal{L}_H]$, then by Lemma 4.12 $\mathbb{E}[\log_d(1 + |\mathcal{K}_H|)] \leq \mathbb{E}[\log_d(1 + |\mathcal{L}_H|)]$. So it remains to prove that $\Pr[T^c \in \mathcal{K}_H] \leq \Pr[T^c \in \mathcal{L}_H]$.

For this, we can simply apply Lemma 3.12 from the weak PPSZ case. We are in the case of a finite, non-full tree of degree $(d-1)(k-1)$ (in particular, our tree has root degree $(k-1)$), and the experiment is exactly the one from Lemma 3.12. Hence, the argument of the lemma applies and $\Pr[T^c \in \mathcal{K}_H] \leq \Pr[T^c \in \mathcal{L}_H]$, which concludes our proof. □

4.2.8 Integrating over the rank of the root

We have proven, in the previous sections, that, for a fixed p,

$$\mathbb{E}[\log_d(1+|\mathcal{J}_H|)\mid\pi(x)=p]\leq\sum_{j=1}^{d-1}\left(\binom{d-1}{j}\log_d(1+j)\tilde{R}_{(d,k)}(p)^{d-1-j}(1-\tilde{R}_{(d,k)}(p))^j\right)+\varepsilon_H(p)$$

where $\mathcal{J}_H$ is defined as in Lemma 4.11

$$\tilde{R}_{(d,k)}(p)=(p+(1-p)R_{(d,k)}(p))^{k-1}$$

and where, in turn, $R_{(d,k)}(p)$ has been defined in the previous chapter as the smallest $q \geq 0$ satisfying the equation

$$q = (p + (1-p) \cdot q)^{(d-1)(k-1)}.$$

To get rid of the conditioning on p, we want, as usual, to integrate our expression on all possible values of p, which would yield

$$\mathbb{E}[\log_d(1+|\mathcal{J}_H|)] \leq \int_0^1 \left(\sum_{j=1}^{d-1} \left(\binom{d-1}{j} \log_d(1+j) \tilde{R}_{(d,k)}(p)^{d-1-j} (1-\tilde{R}_{(d,k)}(p))^j \right) + \varepsilon_H(p) \right) \mathrm{d}p$$

Moreover, we are interested in the behavior of this expression as H tends to the infinity. For this, we want to apply the dominated convergence theorem. The dominated convergence theorem is a classical result of integration theory and we state it here:

Theorem 4.13 (Dominated convergence theorem [1])**.** Let $\{f_n\}$ be a sequence of real-valued measurable functions on a measure space (S, μ). Suppose that the sequence converges pointwise to a function f and is dominated by some integrable function g in the sense that $|f_n(x)| \leq g(x)$ for all numbers n in the index set of the sequence and all points $x \in S$. Then f is integrable and

$$\lim_{n\to\infty} \int_S f_n \, \mathrm{d}\mu = \int_S f \, \mathrm{d}\mu.$$

We have that, for all p,

$$\begin{aligned} &\lim_{H\to\infty} \sum_{j=1}^{d-1} \left(\binom{d-1}{j} \log_d(1+j) \tilde{R}_{(d,k)}(p)^{d-1-j} (1-\tilde{R}_{(d,k)}(p))^j \right) + \varepsilon_H(p) \\ = \; & \sum_{j=1}^{d-1} \left(\binom{d-1}{j} \log_d(1+j) \tilde{R}_{(d,k)}(p)^{d-1-j} (1-\tilde{R}_{(d,k)}(p))^j \right). \end{aligned}$$

Moreover, $\sum_{j=1}^{d-1} \left(\log_d(1+j) \tilde{R}_{(d,k)}(p)^{d-1-j} (1-\tilde{R}_{(d,k)}(p))^j \right)$ is bounded by a constant, since $R_{(d,k)}$ is, and so $\tilde{R}_{(d,k)}$ is as well, and $\varepsilon_H(p)$ is bounded by 1 (because, in $\log_d(1+|\mathcal{B}|)$, $\mathcal{B}$ has size at most $d-1$). So the whole expression is bounded by a constant. Consequently, we can apply the dominated convergence theorem, and we are now working with the expression

$$\begin{aligned} &\int_0^1 \sum_{j=1}^{d-1} \left(\binom{d-1}{j} \log_d(1+j) \tilde{R}_{(d,k)}(p)^{d-1-j} (1-\tilde{R}_{(d,k)}(p))^j \right) \mathrm{d}p \\ = \; & \sum_{j=1}^{d-1} \binom{d-1}{j} \log_d(1+j) \int_0^1 \tilde{R}_{(d,k)}(p)^{d-1-j} (1-\tilde{R}_{(d,k)}(p))^j \, \mathrm{d}p \end{aligned}$$

As before, $\tilde{R}_{(d,k)}$ is not given explicitely. We will prove the following lemma:

Lemma 4.14. For $p \in \left[\frac{(d-1)(k-1)-1}{(d-1)(k-1)}, 1\right]$, we have $\tilde{R}_{(d,k)}(p) = 1$.

For $p \in \left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$, $\tilde{R}_{(d,k)}$ is a strictly monotone growing function of p.

Proof. We have shown in Lemma 3.14 that, for $p \in \left[\frac{(d-1)(k-1)-1}{(d-1)(k-1)}, 1\right]$, we have $R_{(d,k)}(p) = 1$. By definition of $\tilde{R}_{(d,k)}$, we have that $\tilde{R}_{(d,k)}(p) = R_{(d,k)}(p)^{\frac{1}{d-1}}$ and therefore the first statement holds. Lemma 3.14 also implies that $R_{(d,k)}$ is a strictly monotone growing function for $p \in \left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$ to $[0,1]$, and hence so is $\tilde{R}_{(d,k)}$. $\square$

On the interval $\left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$, we can find an inverse function $\tilde{R}^{-1}_{(d,k)}$ mapping $[0,1]$ to $\left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$.

Lemma 4.15.

$$R^{-1}_{(d,k)}(s) = \frac{s^{\frac{1}{k-1}} - s^{d-1}}{1 - s^{d-1}}.$$

Proof. From Lemma 3.14, we know that the inverse function of $R_{(d,k)}$ on the interval $\left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$ is $R^{-1}_{(d,k)}(s) = \frac{s^{\frac{1}{(d-1)(k-1)}} - s}{1-s}$. Since $\tilde{R}_{(d,k)}(p) = R_{(d,k)}(p)^{1/d-1}$ we have

$$\tilde{R}^{-1}_{(d,k)}(s) = R^{-1}_{(d,k)}(s^{d-1}) = \frac{s^{\frac{1}{k-1}} - s^{d-1}}{1 - s^{d-1}}.$$

□

We define $h_j : [0,1] \to [0,1]$ where $h_j(p) = t^{d-1-j}(1-t)^j$. Hence, our integral can be written as

$$\int_0^1 \tilde{R}_{(d,k)}(p)^{d-1-j}(1 - \tilde{R}_{(d,k)})^j \,\mathrm{d}p = \int_0^1 h_j(\tilde{R}_{(d,k)}(p)) \,\mathrm{d}p.$$

We will finally use the following lemma:

Lemma 4.16.

$$\int_0^1 h_j(\tilde{R}_{(d,k)}(p)) \,\mathrm{d}p = \int_0^1 h_j(s)(\tilde{R}^{-1})'(s) \,\mathrm{d}s.$$

Proof. We use again the following substitution of Lemma 3.16, which we recall here:

Lemma 3.16. Let $f : I \to \mathbb{R}$ be a continuous function and $\phi : [a,b] \to \mathbb{R}$ a continuous differentiable function where $\phi([a,b]) \subset I$. Then we have

$$\int_a^b f(\phi(t))\phi'(t)\mathrm{d}t = \int_{\phi(a)}^{\phi(b)} f(x) \,\mathrm{d}x.$$

Now, we have that $\tilde{R}^{-1}$ is only defined if $\tilde{R}$ is restricted to $\left[0, \frac{(d-1)(k-1)-1}{(d-1)(k-1)}\right]$, we first have to split up our integral. For $j > 0$, we have

$$\begin{aligned}
\int_0^1 h_j(\tilde{R}_{(d,k)}(p)) \,\mathrm{d}p &= \int_0^{\frac{(d-1)(k-1)-1}{(d-1)(k-1)}} h_j(\tilde{R}_{(d,k)}(p)) \,\mathrm{d}p + \int_{\frac{(d-1)(k-1)-1}{(d-1)(k-1)}}^1 h_j(\tilde{R}_{(d,k)}(p)) \,\mathrm{d}p \\
&= \int_0^{\frac{(d-1)(k-1)-1}{(d-1)(k-1)}} h_j(\tilde{R}_{(d,k)}(p)) \,\mathrm{d}p,
\end{aligned}$$

where the last equation follows since $\tilde{R}_{(d,k)}(p) = 1$ on $\left[\frac{(d-1)(k-1)-1}{(d-1)(k-1)}, 1\right]$ and $h_j(1) = 0$. Now we use our substitution rule and replace f and ϕ by $h \circ \tilde{R}$ and $\tilde{R}^{-1}$ respectively. The preconditions hold for $p < 1$ and therefore we have

$$\begin{aligned}
\int_0^1 h_j(\tilde{R}(p)) \,\mathrm{d}p &= \lim_{\varepsilon \to 0} \int_{\tilde{R}^{-1}(0)}^{\tilde{R}^{-1}(1)-\varepsilon} h(\tilde{R}(\tilde{R}^{-1}(s)))(\tilde{R}^{-1})'(s) \,\mathrm{d}s \\
&= \lim_{\varepsilon \to 0} \int_0^{1-\varepsilon} h_j(s)(\tilde{R}^{-1})'(s) \,\mathrm{d}s \\
&= \int_0^1 h_j(s)(\tilde{R}^{-1})'(s) \,\mathrm{d}s
\end{aligned}$$

□

To wrap up, we have that

$$\int_0^1 \tilde{R}_{(d,k)}(p)^{d-1-j}(1-\tilde{R}_{(d,k)}(p))^j \,\mathrm{d}p = \int_0^1 s^{d-1-j}(1-s)^j(\tilde{R}^{-1})'(s)\,\mathrm{d}s$$

and

$$\tilde{R}^{-1}_{(d,k)}(s) = \frac{s^{\frac{1}{k-1}} - s^{d-1}}{1-s^{d-1}}$$

which yields

$$\begin{aligned} & \int_0^1 \tilde{R}_{(d,k)}(p)^{d-1-j}(1-\tilde{R}_{(d,k)}(p))^j \,\mathrm{d}p \\ = & \lim_{\varepsilon\to 0}\int_0^{1-\varepsilon} s^{d-1-j}(1-s)^j \\ & \cdot\frac{(1-s^{d-1})(\frac{1}{k-1}s^{\frac{1}{k-1}-1}-(d-1)s^{d-2}) + (s^{\frac{1}{k-1}}-s^{d-1})(d-1)s^{(d-2)}}{(1-s^{d-1})^2}\,\mathrm{d}s. \end{aligned}$$

And this concludes the proof of Theorem 4.1.

4.2.9 Summary

To prove Theorem 4.1, we went through the following steps:

1. We related the probability of success of the strong PPSZ algorithm to the expected size of the set of "non-forbidden" elements at each step (the sets $\mathcal{A}(x,\pi,D)$).
2. We related the size of this set to the expected number of trees that exceed a given size after we apply some random cuts with a given probability to a set of trees
3. We considered the case where these trees are full, infinite, and where the cuts are done independently, and we proved that this was the worst possible case for the size of the set of "large" trees.
4. We integrated the result to get the final result for any variable, independently of its place.

4.3 Multiple satisfying assignments

In the previous section, we have proven Theorem 4.1, which deals with the case where F has a unique assignment. In general, though, we cannot rely on such a strong assumption, and we would like to prove the following:

Conjecture 4.17. For any (d,k)-ClSP formula F on n variables V, PPSZ-STRONG$(F, V, \alpha_0, \log\log n)$ returns some assignment with probability $\Omega(d^{-S_{(d,k)}n-o(n)})$, where

$$\begin{aligned} S_{(d,k)} = & \sum_{j=0}^{d-1}\binom{d-1}{j}\log_d(1+j) \\ \cdot\int_0^1 & s^{d-1-j}(1-s)^j\cdot\frac{(1-s^{d-1})(\frac{1}{k-1}s^{\frac{1}{k-1}-1}-(d-1)s^{d-2}) + (s^{\frac{1}{k-1}}-s^{d-1})(d-1)s^{(d-2)}}{(1-s^{d-1})^2}\,\mathrm{d}s. \end{aligned}$$

This conjecture is not fully proven yet, but we give some elements towards the proof that follow the structure established in Chapter 2: we will define a cost function, and relate it to the probability of success of PPSZ-STRONG. The main difference is that our success probability does not depend on the probability that a variable is guessed (as in the 3-SAT case or in the PPSZ-WEAK case), but on the quantity $\mathbb{E}[\log_d(|\mathcal{A}(x, \alpha_0, \alpha, \pi, D)|]$, where we defined $\mathcal{A}(x, \alpha_0, \alpha, \pi, D)$ to be the set $\{c \in [d] \mid F^{[\alpha_0 \cup \beta]} \not\models_D (x \neq c)\}$. Consider a fixed satisfying assignment α. Observe that, for a fixed placement π, the probability that the algorithm returns α when starting in state α_0 is

$$\Pr[\text{PPSZ-STRONG}(F, V, \alpha_0, D, \pi) \text{ returns } \alpha] = \prod_{x \in \mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{A}(x, \alpha_0, \alpha, \pi, D)|}$$

which yields, for a placement π chosen uniformly at random:

$$\Pr[\text{PPSZ-STRONG}(F, V, \alpha_0, D, \pi) \text{ returns } \alpha] = \mathbb{E}\left[\prod_{x \in \mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{A}(x, \alpha_0, \alpha, \pi, D)|}\right]$$

Applying Jensen's inequality with the function $x \mapsto d^{-x}$ as the convex function yields

$$\mathbb{E}\left[\prod_{x \in \mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{A}(x, \alpha_0, \alpha, \pi, D)|}\right] \geq d^{-\sum_{x \in \mathcal{U}(\alpha_0)} \mathbb{E}[\log_d |\mathcal{A}(x, \alpha_0, \alpha, \pi, D)|]}.$$

In the unique case, we have shown that $\mathbb{E}[\log_d |\mathcal{A}(x, \alpha_0, \alpha, \pi, D)|] \leq S_{(d,k)}$, which proved the theorem. But as in the 3-SAT case and in the PPSZ-WEAK case, the analysis of the unique case does not go through in the general case because there is no guarantee that we can build critical clause trees for the formula. If a given variable has the same value in all the satisfying assignments, we call this variable "frozen". In that case, the argument goes through, and we can state the following lemma:

Lemma 4.18. Let F be a (d, k)-ClSP formula and x be a frozen variable. Let furthermore α be any satisfying assignment, and α_0 be a partial assignment compatible with α. Then

$$\mathbb{E}[\log_d(|\mathcal{A}(x, \alpha_0, \alpha, \pi, D)|)] \leq S^{(D)}_{(d,k)}.$$

4.3.1 Definition of a cost function

We proceed again as in the 3-SAT case and as in the PPSZ-WEAK case. We want to define a cost function that is bounded by $S^{(D)}_{(d,k)}$. To lighten the notation, we will, from now on, consider that $D = \log \log n$ and we will define $S^{(D)}_{(d,k)} = S_{(d,k)}$. The intuition is, however, slightly different from the previous cases. In the previous cases, we were only interested in whether a variable was forced or not. Here, we want to know at which degree a variable is forced. Hence, instead of evaluating the probability (over the permutations and the previous choices of assignments) that a given variable has only one possible value (i.e. is forced), we want to evaluate the expected number of possible values, and we want our cost value to encompass that.

Let us recall the following definition:

Definition 4.2. For every variable x, every total assignment α, every partial assignment α_0 compatible with α, every integer D, and every permutation π, we define the sets $\mathcal{A}(x, \alpha_0, \alpha, \pi, D)$ as follows:

$$\mathcal{A}(x, \alpha_0, \alpha, \pi, D) = \{c \in [d] \mid F^{[\alpha_0 \cup \beta]} \not\models_D (x \neq c)\}$$

where β is the partial assignment corresponding to the restriction of α to the variables that come before x in π.

For a fixed total assignment α and a partial assignment α_0, we now define

$$\mathrm{bits}(x, \alpha_0, \alpha, D) = \mathop{\mathbb{E}}_{\pi} \log[|\mathcal{A}(x, \alpha_0, \alpha, \pi)|].$$

From this definition, we define the cost function directly:

$$\mathrm{cost}(\alpha_0, \alpha, x) = \begin{cases} 0 & \text{if } x \notin \mathcal{U}(\alpha_0) \\ 0 & \text{if } x \in \mathrm{V_{fo}}(\alpha_0) \\ S_{(d,k)} & \text{if } x \in \mathrm{V_{nf}}(\alpha_0) \\ \mathrm{bits}(x, \alpha_0, \alpha, D) & \text{if } x \in \mathrm{V_{fr}}(\alpha_0) \end{cases}$$

As in the previous cases, we also introduce the likelihood of a given assignment as follows:

Definition 2.17. Let $F^{[\alpha_0]}$ be satisfiable and let $\mathcal{S}_{\alpha_0}$ be the set of value assignments $l = \{x \mapsto b\}$ such that $x \in \mathcal{U}(\alpha_0)$ and $F^{[\alpha_0[l]]}$ is satisfiable.

We define the random process $\mathrm{AssignSL}(F, \alpha_0)$ that produces an assignment on $\mathrm{vbl}(F)$ as follows. Start with the assignment α_0, and repeat the following step until $\mathrm{vbl}(F^{[\alpha_0]}) = \emptyset$: Choose a value assignment $l \in \mathcal{S}_{\alpha_0}$ uniformly at random and add l to α_0. At the end, output α_0.

Let α be a total assignment on $\mathrm{vbl}(F)$. Then the *likelihood of completing* α_0 *to* α, in writing $\mathrm{lkhd}(\alpha_0, \alpha)$ is defined as the probability that $\mathrm{AssignSL}(F, \alpha_0)$ returns α. For completeness, if $F^{[\alpha_0]}$ is not satisfiable, we define $\mathrm{lkhd}(\alpha_0, \alpha) = 0$.

This definition allows us to define a cost function corresponding to the cost of a variable when completing a given partial assignment α_0 to any satisfying assignment:

$$\mathrm{cost}(\alpha_0, x) = \sum_{\alpha \in \mathrm{sat}_V(F)} \mathrm{lkhd}(\alpha_0, \alpha) \cdot \mathrm{cost}(\alpha_0, \alpha, x) = \sum_{\alpha \in \mathrm{sat}_V(F)} \mathrm{wcost}(\alpha_0, \alpha, x)$$

Finally, we define the cost of completing a given partial assignment to any satisfying assignment by summing over all variables:

$$\mathrm{cost}(\alpha_0) = \sum_{x \in V} \mathrm{cost}(\alpha_0, x).$$

We would like to prove the following conjecture:

Conjecture 4.19. Let α_0 be such that $F^{[\alpha_0]}$ is satisfiable. Then the overall probability of PPSZ-STRONG to output some satisfying assignment when starting in state α_0 is at least $d^{-\mathrm{cost}(\alpha_0)}$.

Observe the following:

Observation 4.20. For any α_0, α, x, we have $\mathrm{cost}(\alpha_0, \alpha, x) \leq S_{(d,k)}$. Furthermore, $\mathrm{cost}(\alpha_0) \leq n(\alpha_0) S_{(d,k)}$.

This observation follows directly from the definition of the cost and from Lemma 4.18. Proving Conjecture 4.19 would allow us to conclude directly that Conjecture 4.17 holds as well.

4.3.2 Towards the proof of Conjecture 4.19

Setup of the proof of Conjecture 4.19

We first define $p(\alpha_0)$ as the probability that PPSZ-STRONG outputs some satisfying assignment when starting from state α_0.

We define the set $\mathcal{S}_{\alpha_0}$ as follows:

$$\mathcal{S}_{\alpha_0} := \{(x, c) \in \mathcal{U}(\alpha_0) \times [d] \mid F^{[\alpha_0 \cup \{x \mapsto c\}]} \text{ is satisfiable}\}$$

and the set $\mathcal{S}_{\alpha_0}(x)$, for every $x \in \mathcal{U}(\alpha_0)$, as

$$\mathcal{S}_{\alpha_0}(x) := \{(x, c) \in \{x\} \times [d] \mid F^{[\alpha_0 \cup \{x \mapsto c\}]} \text{ is satisfiable}\}.$$

The set $\mathcal{S}_{\alpha_0}$ represents all the choices that the PPSZ-STRONG run can do and still have a satisfying formula in the next step. The set $\mathcal{S}_{\alpha_0}$ contains at least $2|\mathrm{V_{nf}}(\alpha_0)|+|\mathrm{V_{fo}}(\alpha_0)|+|\mathrm{V_{fr}}(\alpha_0)|$ elements.

We would like to be able to prove Conjecture 4.19 by induction, as we did in the 3-SAT case and in the PPSZ-WEAK case. We suppose that the claim holds for all α_0 that fix a larger number of variables. If α_0 is total, then the statement holds trivially, because the cost of α_0 is 0, and $p(\alpha_0) = 1$.

Let x and c be random variables: $x \in \mathcal{U}(\alpha_0)$ and c is the value chosen u.a.r. in the set of allowed values for x (i.e. such that $(x \neq c)$ is not $D-$implied by $F^{[\alpha_0]}$), which we denote from now on for ease of writing as $\mathcal{A}(x, \alpha_0)$. Then we have

$$p(\alpha_0) \quad = \quad \mathop{\mathbb{E}}_{x \in_{\text{u.a.r.}} \mathcal{U}(\alpha_0)} [p(\alpha_0 \cup \{x \mapsto c\})].$$

Relating to uniform choice over $\mathcal{S}_{\alpha_0}$

As we did in the 3-SAT case and in the PPSZ-WEAK case, we relate the probability distribution over "x, then c" to the distribution uniformly at random over $\mathcal{S}_{\alpha_0}$. However, in the previous cases, we could split the variables into frozen, forced and non frozen, and have constant probabilities for all these classes of variables. Here, whatever the class of the variable, we have, for each variable x, a probability that depends on $|\mathcal{A}(x, \alpha_0)|$ to pick a valid choice for c. Hence, we use another approach to relate the above probability to the uniform choice over assignments in $\mathcal{S}_{\alpha_0}$.

We have:

$$\begin{aligned} p(\alpha_0) &= \mathop{\mathbb{E}}_{x \in_{\text{u.a.r.}} \mathcal{U}(\alpha_0)} [p(\alpha_0 \cup \{x \mapsto c\})] \\ &= \frac{1}{n(\alpha_0)} \sum_{x \in \mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{A}(x, \alpha_0)|} \sum_{c \in \mathcal{A}(x, \alpha_0)} p(\alpha_0[x \mapsto c]) \\ &= \frac{1}{n(\alpha_0)} \sum_{x \in \mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{A}(x, \alpha_0)|} \sum_{l \in \mathcal{S}_{\alpha_0}(x)} p(\alpha_0[l]). \end{aligned}$$

This last equality comes from the fact that $\mathcal{S}_{\alpha_0}(x) \subseteq \mathcal{A}(x, \alpha_0)$ and, if $l \notin \mathcal{S}_{\alpha_0}(x)$, then $p(\alpha_0[l]) = 0$. We can now sum over all of $\mathcal{S}_{\alpha_0}$:

$$\begin{aligned} p(\alpha_0) &= \frac{1}{n(\alpha_0)} \sum_{l \in \mathcal{S}_{\alpha_0}} \frac{1}{|\mathcal{A}(\mathrm{vbl}(l), \alpha_0)|} p(\alpha_0[l]) \\ &= \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} \frac{1}{|\mathcal{S}_{\alpha_0}|} \sum_{l \in \mathcal{S}_{\alpha_0}} \frac{1}{|\mathcal{A}(\mathrm{vbl}(l), \alpha_0)|} p(\alpha_0[l]) \\ &= \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} \mathop{\mathbb{E}}_{l \in_{\text{u.a.r}} \mathcal{S}_{\alpha_0}} \left[\frac{1}{|\mathcal{A}(\mathrm{vbl}(l), \alpha_0)|} p(\alpha_0[l]) \right]. \end{aligned}$$

Using the induction hypothesis

We have now expressed $p(\alpha_0)$ in terms of an expectation that depends on the uniform choice over the possible literal in $\mathcal{S}_{\alpha_0}$. We first use Jensen's inequality:

$$p(\alpha_0) \geq d^{\log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\right)+\mathbb{E}[-\log_d(|\mathcal{A}(\mathrm{vbl}(l),\alpha_0)|p(\alpha_0[l]))]}$$

and then we apply the induction hypothesis and the linearity of expectation, which yields

$$p(\alpha_0) \geq d^{\log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\right)-\mathbb{E}[\log_d(\mathcal{A}(\mathrm{vbl}(l),\alpha_0)]-\mathbb{E}[\mathrm{cost}(\alpha_0[l])]}.$$

We are interested in the quantity

$$L := \log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\right) - \mathbb{E}[\log_d|\mathcal{A}(\mathrm{vbl}(l),\alpha_0)|] - \mathbb{E}[\mathrm{cost}(\alpha_0[l])] + \mathrm{cost}(\alpha_0) \tag{4.1}$$

and we want to prove that it is non-negative, which would then prove that Conjecture 4.19 holds. For this, we need to evaluate $\mathbb{E}[\mathrm{cost}(\alpha_0[l])]$, for which we need to establish a few facts about cost and likelihood, in a manner very similar to Lemma 3.21.

Proof. □

We can now prove the following facts about cost and likelihood.

Lemma 4.21. Let α_0 and α be fixed and compatible. For any fixed variable $x \in \mathcal{U}(\alpha_0)$, if we set x according to α, then

(i) the likelihood of α can only increase, i.e.

$$\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \geq \mathrm{lkhd}(\alpha_0, \alpha).$$

(ii) the cost of a fixed variable $y \in V$ w.r.t. α can only decrease, i.e.

$$\mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}) \leq \mathrm{cost}(\alpha_0, \alpha, y).$$

When choosing $x \in \mathcal{U}(\alpha_0)$ uniformly at random and setting it according to α, then

(iii) the likelihood of α increases on average as

$$\mathbb{E}[\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)] = \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\mathrm{lkhd}(\alpha_0, \alpha).$$

(iv) the cost of a fixed, frozen, non-forced variable $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$ decreases on expectation as

$$\mathbb{E}[\mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)] \leq \mathrm{cost}(\alpha_0, \alpha, y) - \frac{1}{n(\alpha_0)}\log_d|\mathcal{A}(y, \alpha_0)|.$$

Proof. We adapt the proof of Lemma 3.21 to the PPSZ-STRONG case; the arguments are identical for the likelihood ((i) and (iii)), and differ slightly for the cost ((ii) and (iv)) part of the lemma.

(i) We prove the claim by induction over the size of α_0. The claim holds trivially if α_0 is a complete assignment. Otherwise, we have

$$\begin{aligned}
\mathrm{lkhd}(\alpha_0, \alpha) &= \mathop{\mathbb{E}}_{(x',c') \in \mathcal{S}_{\alpha_0}} \mathrm{lkhd}(\alpha_0 \cup \{x' \mapsto c'\}, \alpha) \\
&= \sum_{(x',c') \in \mathcal{S}_{\alpha_0}} \frac{1}{|\mathcal{S}_{\alpha_0}|} \mathrm{lkhd}(\alpha_0 \cup \{x' \mapsto c'\}, \alpha) \\
&= \sum_{x' \in \mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{S}_{\alpha_0}|} \mathrm{lkhd}(\alpha_0 \cup \{x' \mapsto \alpha(x')\}, \alpha) \\
&= \frac{1}{|\mathcal{S}_{\alpha_0}|} \left(\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) + \sum_{x' \in \mathcal{U}(\alpha_0) \setminus \{x\}} \mathrm{lkhd}(\alpha_0 \cup \{x' \mapsto \alpha(x')\}, \alpha) \right).
\end{aligned}$$

We apply the induction hypothesis and we get

$$\begin{aligned}
\mathrm{lkhd}(\alpha_0, \alpha) &\le \frac{1}{|\mathcal{S}_{\alpha_0}|} \Bigg(\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \\
&\quad + \sum_{x' \in \mathcal{U}(\alpha_0) \setminus \{x\}} \mathrm{lkhd}(\alpha_0 \cup \{x' \mapsto \alpha(x')\} \cup \{x \mapsto \alpha(x)\}, \alpha) \Bigg) \\
&= \frac{1}{|\mathcal{S}_{\alpha_0}|} \big(\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \\
&\quad + |\mathcal{S}_{\alpha_0 \cup \{x \mapsto \alpha(x)\}}| \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \big).
\end{aligned}$$

Now observe that, since $\mathcal{S}_{\alpha_0 \cup \{x \mapsto c\}} \subsetneq \mathcal{S}_{\alpha_0}$, then $|\mathcal{S}_{\alpha_0 \cup \{x \mapsto c\}}| \le |\mathcal{S}_{\alpha_0}| - 1$, and this allows us to conclude that

$$\mathrm{lkhd}(\alpha_0, \alpha) \le \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}).$$

(ii) We consider the three cases: the variable y is non frozen, frozen or forced. Note that if $x = y$, the statement holds trivially.

If $y \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$, then $\mathrm{cost}(\alpha_0, \alpha, y) = S_{(d,k)}$. Since the cost of a variable is always less than $S_{(d,k)}$, the statement holds.

If $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$ or $y \in \mathrm{V}_{\mathrm{fo}}(\alpha_0)$, then $\mathrm{cost}(\alpha_0, \alpha, y)$ is the expected logarithm of the number of non-forbidden values for y in the remainder of the PPSZ-STRONG run. If we now fix another variable x to $\alpha(x)$, then this expectation cannot decrease, because adding a value assignment cannot allow a value that was forbidden. So $\mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y) \le \mathrm{cost}(\alpha_0, \alpha, y)$.

(iii) We have

$$\begin{aligned}
\mathrm{lkhd}(\alpha_0, \alpha) &= \sum_{(x,c) \in \mathcal{S}_{\alpha_0}} \frac{1}{|\mathcal{S}_{\alpha_0}|} \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto c\}, \alpha) \\
&= \sum_{x \in \mathcal{U}(\alpha_0)} \frac{1}{|\mathcal{S}_{\alpha_0}|} \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \\
&= \frac{n(\alpha_0)}{|\mathcal{S}_{\alpha_0}|} \mathop{\mathbb{E}}_{x \in_{\mathrm{u.a.r}} \mathcal{U}(\alpha_0)} [\mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)],
\end{aligned}$$

which proves the statement.

(iv) For a fixed, frozen, non-forced variable, we have that $\operatorname{cost}(\alpha_0, \alpha, y) = \operatorname{bits}(y, \alpha_0, \alpha, D)$. Let π be a random permutation on V and let z be the variable that comes next (after α_0 has been assigned) in π. By the law of total probability, we have that

$$\operatorname{bits}(y, \alpha_0, \alpha, D) = \mathop{\mathbb{E}}_{z}[\mathop{\mathbb{E}}_{\pi}[\log_d |\mathcal{A}(y, \alpha_0, \alpha, \pi, D)|] \mid z \text{ first in } \pi].$$

If $y = z$, then the expression in the expectation is $\log_d |\mathcal{A}(y, \alpha_0)|$. If $y \neq z$, then the expression in the expectation is $\operatorname{bits}(y, \alpha_0 \cup \{z \mapsto \alpha[z]\}, \alpha, D)$, and thus

$$\operatorname{bits}(y, \alpha_0, \alpha, D) = \frac{1}{n(\alpha_0)} \log_d |\mathcal{A}(y, \alpha_0)| + \frac{1}{n(\alpha_0)} \sum_{z \in \mathcal{U}(\alpha_0) \setminus \{y\}} \operatorname{bits}(y, \alpha_0 \cup \{z \mapsto \alpha[z]\}, \alpha, D).$$

But then, $\operatorname{bits}(y, \alpha_0 \cup \{y \mapsto \alpha(y)\}, \alpha, D) = 0$ so

$$\operatorname{bits}(y, \alpha_0, \alpha, D) = \frac{1}{n(\alpha_0)} \log_d |\mathcal{A}(y, \alpha_0)| + \frac{1}{n(\alpha_0)} \sum_{x \in \mathcal{U}(\alpha_0)} \operatorname{bits}(y, \alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, D),$$

which yields

$$\mathbb{E}[\operatorname{cost}(y, \alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)] = \operatorname{cost}(y, \alpha_0, \alpha) - \frac{1}{n(\alpha_0)} \log_d |\mathcal{A}(y, \alpha_0)|$$

as desired.

□

Now we can evaluate $\mathbb{E}[\operatorname{cost}(\alpha_0[l])]$.

Evaluating $\mathbb{E}[\mathbf{cost}(\alpha_0[l])]$

In this section, we will prove the following lemma:

Lemma 4.22. If $l \in \mathcal{S}_{\alpha_0}$ is selected uniformly at random, then

$$\mathbb{E}[\operatorname{cost}(\alpha_0[l])] \leq \operatorname{cost}(\alpha_0) - \frac{1}{|\mathcal{S}_{\alpha_0}|} \left(\sum_{x \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)} \log_d |\mathcal{A}(\alpha_0, x)| + S_{(d,k)} \sum_{x \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)} |\mathcal{S}_{\alpha_0}(x)| \right).$$

To prove this lemma, we will need the following auxilliary lemma:

Lemma 4.23. Let α be a fixed satisfying assignment and $\alpha_0 \subseteq \alpha$. Let $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$ be a fixed frozen variable. If we select $x \in \mathcal{U}(\alpha_0)$ uniformly at random and assign it according to α, then

$$\mathbb{E}[\operatorname{wcost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)] \leq \operatorname{wcost}(\alpha_0, \alpha, y) \cdot \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} - \frac{\operatorname{lkhd}(\alpha_0)}{n(\alpha_0)}$$

Proof. We use, as in the previous cases, Lemma 2.24 (proven in Appendix B.2.1, which we recall here:

Lemma 2.24. Let $A, B \in \mathbb{R}$ be random variables and $a, b, \bar{a}, \bar{b} \in \mathbb{R}$ fixed numbers such that $A \geq a$ and $B \leq b$ always, and $\mathbb{E}[A] = \bar{a}$ and $\mathbb{E}[B] = \bar{b}$. Then

$$\mathbb{E}[A \cdot B] \leq a\bar{b} + b\bar{a} - ab.$$

We have that $\mathrm{wcost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y) = \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha) \cdot \mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)$, so we define $A = \mathrm{lkhd}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha)$ and $B = \mathrm{cost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)$. Using Lemma 4.21, we define $a = \mathrm{lkhd}(\alpha_0, \alpha)$, $\bar{a} = \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\mathrm{lkhd}(\alpha_0, \alpha)$, $b = \mathrm{cost}(\alpha_0, \alpha, y)$ and $\bar{b} = \mathrm{cost}(\alpha_0, \alpha, y) - \frac{\log_d |\mathcal{A}(y,\alpha_0)|}{n(\alpha_0)}$. Applying our correlation inequality, we deduce that, for $y \in \mathrm{V}_{\mathrm{fr}}(\alpha_0)$, when selecting $x \in \mathcal{U}(\alpha_0)$ u.a.r. and assigning it according to α, we have

$$\begin{aligned}
& \mathbb{E}[\mathrm{wcost}(\alpha_0 \cup \{x \mapsto \alpha(x)\}, \alpha, y)] \\
\leq \;& \mathrm{lkhd}(\alpha_0, \alpha)\left(\mathrm{cost}(\alpha_0, \alpha, y) - \frac{\log_d |\mathcal{A}(y, \alpha_0)|}{n(\alpha_0)}\right) + \mathrm{cost}(\alpha_0, \alpha, y)\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\mathrm{lkhd}(\alpha 0, \alpha) \\
& -\mathrm{lkhd}(\alpha_0)\mathrm{cost}(\alpha_0, \alpha, y) \\
= \;& \mathrm{wcost}(\alpha_0, \alpha, y)\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} - \frac{\log_d |\mathcal{A}(y, \alpha_0)| \cdot \mathrm{lkhd}(\alpha_0, \alpha)}{n(\alpha_0)}
\end{aligned}$$

□

We can now prove Lemma 4.22.

Proof of Lemma 4.22. We analyze the cost decrease contribution of the different types of variables separately. Each variable that is forced in state α_0 contributes zero cost both before and after the step.

For a non-frozen variable $y \in \mathrm{V}_{\mathrm{nf}}(\alpha_0)$, note that $\mathcal{S}_{\alpha_0}$ contains $|\mathcal{S}_{\alpha_0}(y)|$ pairs containing y, so with probability $|\mathcal{S}_{\alpha_0}(y)|/|\mathcal{S}_{\alpha_0}|$, a pair featuring y is selected. In that case, the cost contribution of y drops from $S_{(d,k)}$ in all assignments. No matter what happens to these costs otherwise (they certainly cannot increase by definition), the non-frozen variables hence contribute to the last term of the claimed inequality.

Now consider the frozen variables. Let $l = (x, c)$. If we fix some satisfying and α_0-compatible assignment $\alpha \in [d]^V$, and condition the experiment on the event that $\alpha(x) = c$, then x becomes uniformly at random among $\mathcal{U}(\alpha_0)$ because $\mathcal{S}_{\alpha_0}$ contains exactly one pair (x', c') per variable $x' \in \mathcal{U}(\alpha_0)$ such that $\alpha(x') = c'$. Now we can apply directly Lemma 4.23 and find that, conditioning on $\alpha(X) = C$, the cost of each frozen variable drops on average as

$$\mathbb{E}[\mathrm{wcost}(\alpha[l], \alpha, y) \mid \alpha(x) = c] \leq \mathrm{wcost}(\alpha_0, \alpha, y) \cdot \frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)} - \frac{\log_d |\mathcal{A}(y, \alpha_0)|.\mathrm{lkhd}(\alpha_0, \alpha)}{n(\alpha_0)}.$$

The condition itself is satisfied with probability $n(\alpha_0)/|\mathcal{S}_{\alpha_0}|$. If it does not apply, then the cost contribution of α drops to zero altogether. Therefore, the unconditional change can be obtained by multiplying the right-hand side by $n(\alpha_0)/|\mathcal{S}_{\alpha_0}|$, which then yields

$$\mathbb{E}[\mathrm{wcost}(\alpha_0[l], \alpha, y)] \leq \mathrm{wcost}(\alpha_0, \alpha, y) - \frac{\log_d |\mathcal{A}(y, \alpha_0)|.\mathrm{lkhd}(\alpha_0, \alpha)}{|\mathcal{S}_{\alpha_0}|}.$$

If we sum over all assignments $\alpha \in [d]^V$ and all the variables, the claim follows. □

Putting everything together

We are back to the main track of the proof of Conjecture 4.19, and we can now introduce the result from Lemma 4.22 into equation (4.1). We obtain, cancelling $\text{cost}(\alpha_0)$:

$$\begin{aligned} L &= \log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\right) - \mathbb{E}[\log_d|\mathcal{A}(\text{vbl}(l),\alpha_0)|] - \mathbb{E}[\text{cost}(\alpha_0[l])] + \text{cost}(\alpha_0) \\ &\geq \log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\right) - \mathbb{E}[\log_d|\mathcal{A}(\text{vbl}(l),\alpha_0)|] \\ &\quad + \frac{1}{|\mathcal{S}_{\alpha_0}|}\left(\sum_{x\in \text{V}_{\text{fr}}(\alpha_0)} \log_d|\mathcal{A}(x,\alpha_0)| + S_{(d,k)}\sum_{x\in\text{V}_{\text{nf}}(\alpha 0)}|\mathcal{S}_{\alpha_0}(x)|\right). \end{aligned}$$

Furthermore, we have

$$\begin{aligned} \mathbb{E}[\log|\mathcal{A}(\text{vbl}(l),\alpha_0)|] &= \frac{1}{|\mathcal{S}_{\alpha_0}|}\left(\sum_{l\in\mathcal{S}_{\alpha_0}}\log_d|\mathcal{A}(\text{vbl}(l),\alpha_0)|\right) \\ &\leq \frac{1}{|\mathcal{S}_{\alpha_0}|}\left(\sum_{x\in\text{V}_{\text{fr}}(\alpha_0)}\log_d|\mathcal{A}(x,\alpha_0)| + \sum_{x\in\text{V}_{\text{nf}}(F)}|\mathcal{S}_{\alpha_0}(x)|\right), \end{aligned}$$

and therefore we can conclude that

$$L \geq \log_d\left(\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}\right) + \frac{1}{|\mathcal{S}_{\alpha_0}|}(S_{(d,k)}-1)\sum_{x\in\text{V}_{\text{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)|.$$

Now we know that

$$\mathcal{S}_{\alpha_0} = |\text{V}_{\text{fo}}(\alpha_0)| + |\text{V}_{\text{fr}}(\alpha_0)| + \sum_{x\in\text{V}_{\text{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)| = n(\alpha_0) + \sum_{x\in\text{V}_{\text{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)|-1)$$

so

$$L \geq \log_d\left(1+\frac{\sum_{x\in\text{V}_{\text{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)|-1)}{n(\alpha_0)}\right) - \frac{1}{|\mathcal{S}_{\alpha_0}|}\left((1-S_{(d,k)}\sum_{x\in\text{V}_{\text{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)|\right).$$

We now use the inequality $\log_d(1+x) \geq \log_d(e)\frac{x}{1+x}$ (proven in Appendix B.1.1) to get that

$$\begin{aligned} L &\geq \log_d(e)\frac{\frac{\sum_{x\in\text{V}_{\text{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)|-1)}{n(\alpha_0)}}{\frac{|\mathcal{S}_{\alpha_0}|}{n(\alpha_0)}} - \frac{1}{|\mathcal{S}_{\alpha_0}|}\left((1-S_{(d,k)})\sum_{x\in\text{V}_{\text{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)|\right) \\ &= \frac{1}{|\mathcal{S}_{\alpha_0}|}\left(\log_d(e)\sum_{x\in\text{V}_{\text{nf}}(\alpha_0)}(|\mathcal{S}_{\alpha_0}(x)|-1) - (1-S_{(d,k)})\sum_{x\in\text{V}_{\text{nf}}(\alpha_0)}|\mathcal{S}_{\alpha_0}(x)|\right) \\ &= \frac{1}{|\mathcal{S}_{\alpha_0}|}\sum_{x\in\text{V}_{\text{nf}}(\alpha_0)}\left(\log_d(e)(|\mathcal{S}_{\alpha_0}(x)|-1) - (1-S_{(d,k)})|\mathcal{S}_{\alpha_0}(x)|\right) \end{aligned}$$

Hence, if we show that, for $x \in \text{V}_{\text{nf}}(\alpha_0)$, we have

$$\log_d(e)(|\mathcal{S}_{\alpha_0}(x)|-1) - (1-S_{(d,k)})|\mathcal{S}_{\alpha_0}(x)| \geq 0,$$

this would prove that $L \geq 0$ and hence Conjecture 4.19. In order to show that the theorem holds for arbitrary d, since $\frac{|\mathcal{S}_{\alpha_0}(x)|}{|\mathcal{S}_{\alpha_0}(x)|-1} \leq 2$, this is given if

$$S_{(d,k)} \geq 1 - \frac{\log_d e}{2}.$$

From the definition of $S_{(d,k)}$, it can be seen that $S_{(d,k)}$ increases for larger k. Hence, $S_{(d,k)} \geq S_{(d,3)}$. We have the following numerical values: $S_{(3,3)} \approx 0.584497273$; $\log_3(e) \approx 0.910239$; $S_{(4,3)} \approx 0.654828537$; $\log_4(e) \approx 0.721347$. This allows us to conclude that this is the case for $d = 3$ and $d = 4$, which we summarize in the following theorem:

Theorem 4.24. For any $(d, 3)$-ClSP or $(d, 4)$-ClSP formula F on n variables V, PPSZ-STRONG$(F, V, \alpha_0, \log\log n)$ returns this assignment with probability $\Omega(d^{-S_{(d,k)}n-o(n)})$, where

$$S_{(d,k)} = \sum_{j=0}^{d-1} \binom{d-1}{j} \log_d(1+j)$$

$$\cdot \int_0^1 s^{d-1-j}(1-s)^j \cdot \frac{(1-s^{d-1})(\frac{1}{k-1}s^{\frac{1}{k-1}-1} - (d-1)s^{d-2}) + (s^{\frac{1}{k-1}} - s^{d-1})(d-1)s^{(d-2)}}{(1-s^{d-1})^2} \, \mathrm{d}s.$$

The proof of the general claim of Conjecture 4.19 is still open.

Appendix A

Glossary of classical notions

The results of this section are well-known result of analysis and probability theory. The formulation and proofs that we give here are the ones given in [8].

A.1 Jensen's inequality

Theorem A.1. Let I be a real interval. If $f : \mathbb{I} \to R$ is a convex function and X is a random variable that attains values in I only, then

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]),$$

provided that both expectations exist.

Proof. Let $\mu = \mathbb{E}[X]$. Let λ be such that $f(x) \geq f(\mu) + \lambda(x - \mu)$ for all $x \in I$. λ exists because f is convex. Due to linearity of expectation,

$$\mathbb{E}[f(X)] \geq \mathbb{E}[f(\mu) + \lambda(X - \mu)] = f(\mu) + \lambda(\mathbb{E}[X] - \mu) = f(\mu) = f(\mathbb{E}[X]),$$

and we are done. □

A.2 Monotone convergence theorem

Theorem A.2 (Monotone Convergence Theorem)**.** Let, in a discrete probability space, B_1, B_2, ... be an infinite sequence of events such that $B_i \supseteq B_{i+1}$ for all $i \geq 1$. Then we have

$$\lim_{n\to\infty} \Pr[B_n] = \Pr\left[\bigcap_{i=1}^{\infty} B_i\right].$$

Proof. We write

$$\bigcap_{i=1}^{\infty} B_i = B_1 \backslash \dot{\bigcup_{i=1}^{\infty}} (B_i \backslash B_{i+1}).$$

Since the union is disjoint and moreover contained in B_1, this implies

$$\Pr\left[\bigcap_{i=1}^{\infty} B_i\right] = \Pr[B_1] - \sum_{i=1}^{\infty} \Pr[B_i \backslash B_{i+1}].$$

By definition, the infinite sum on the right hand-side is the limit of sums truncated after the first n summands and since $\Pr[B_1]$ does not depend on this truncation we can write

$$\Pr\left[\bigcap_{i=1}^{\infty} B_i\right] = \lim_{n\to\infty}\left(\Pr[B_1] - \sum_{i=1}^{n}\Pr[B_i \backslash B_{i+1}]\right).$$

The expression inside the limit is $\Pr[B_{n+1}]$, establishing the claim. □

A.3 FKG inequality

Let $\mathcal{A} = \{A_1, A_2, ..., A_r\}$ be a collection of independent binary random variables. An event E is said to be *determined* by $\mathcal{A}$ if there exists a fixed list $S_E \subseteq 2^{\mathcal{A}}$ such that $E = \{\{A \in \mathcal{A} \mid A = 1\} \in S_E\}$, or, informally speaking, if knowing the values of $\mathcal{A}$ leads to knowing whether E occurs. Moreover, if S_E is upwards heritary, i.e. if

$$\forall \mathcal{A} \supseteq U \supseteq V : V \in S_E \Rightarrow U \in S_E,$$

then E is called *monotonically increasing in* $\mathcal{A}$.

Theorem 2.13. *Let $\mathcal{A} = \{A_1, A_2, ..., A_r\}$ be a collection of independent binary random variables and E_1 and E_2 events which are determined by $\mathcal{A}$ and monotonically increasing in $\mathcal{A}$. Then*

$$\Pr[E_1 \wedge E_2] \geq \Pr[E_1] \cdot \Pr[E_2].$$

Proof. We proceed by induction on r.

For the base case, let $r = 1$. Then, there are only two non-empty monotonically increasing events determined by A_1: either an event that occurs for both values of A_1 or an event that occurs only if $A_1 = 1$. If $E_1 = E_2$, the statement is trivial. Let now E_1 be the first of these cases and E_2 be the other. If p is the probability that $A_1 = 1$, then the probability that both events occur is p, the probability that E_1 occurs is 1 and the probability that E_2 occurs is p, establishing the claim.

For the induction step, let us assume that the statement holds for smaller values of r and let p be the probability that $A_1 = 1$. We can rewrite the right-hand side of our claim using the law of total probability as:

$$\begin{aligned}\Pr[E_1] \cdot \Pr[E_2] &= (p\Pr[E_1 \mid A_1 = 1] + (1-p)\Pr[E1 \mid A_1 = 0]) \\ &\quad \cdot (p\Pr[E_2 \mid A_1 = 1] + (1-p)\Pr[E2 \mid A_1 = 0])\end{aligned}$$

If we denote $e_{ij} = \Pr[E_i | A_1 = j]$, we get

$$\Pr[E_1] \cdot \Pr[E_2] = p^2 e_{11}e_{21} + p(1-p)(e_{11}e_{20} + e_{10}e_{21}) + (1-p)^2 e_{10}e_{20}.$$

The mutual independence of $\mathcal{A}$ together with the monotonicity of both E_1 and E_2 in A_1 implies that both events can only be more likely in the conditional space determined by $\{A_1 = 1\}$ than in the case $\{A_1 = 0\}$, thus we get $e_{11} \geq e_{10}$ and $e_{21} \geq e_{20}$. We can use this to estimate the mixed term in our expansion: consider $e_{11} \geq e_{10}$ to be weights and the mixed term to be a weighted sum of $e_{21} \geq e_{20}$. Currently, the larger weight accompanies the smaller summand. Thus, if we exchange the weights so that the larger summand gets the larger weight, the weighted sum can only increase. This yields

$$\begin{aligned}\Pr[E_1] \cdot \Pr[E_2] &\leq p^2 e_{11}e_{21} + p(1-p)(e_{10}e_{20} + e_{11}e_{21}) + (1-p)^2 e_{10}e20 \\ &= pe_{11}e_{21} + (1-p)e_{10}e_{20}.\end{aligned}$$

It is now time to invoke the induction hypothesis. We have assumed that the statement holds for smaller r. The events E_1 and E_2 are, in the conditional space of $\{A_1 = 1\}$, determined by $A_2, A_3, ...A_r$ and monotonically increasing in these variables. Therefore, in the conditional space of $\{A_1 = 1\}$, the FKG inequality holds for E_1 and E_2 by the induction hypothesis, yielding that

$$e_{11}e_{21} \leq \Pr[E_1 \wedge E_2 \mid A_1 = 1].$$

Analogously:

$$e_{10}e_{20} \leq \Pr[E_1 \wedge E_2 \mid A_1 = 0].$$

Consequently, we get

$$\Pr[E_1] \cdot \Pr[E_2] \leq p \Pr[E_1 \wedge E_2 \mid A_1 = 1] + (1-p) \Pr[E_1 \wedge E_2 \mid A_1 = 0] = \Pr[E_1 \wedge E_2]$$

as desired, by applying once more the law of total probability.

□

A.4 Riemann sums approximations

Lemma A.3. Let $\phi : [0,1] \to [0,1]$ be a continuous and monotonically non-decreasing function. Then, for any $N \geq 1$,

$$\frac{1}{N}\sum_{i=0}^{N-1} \phi\left(\frac{i}{N}\right) \leq \int_0^1 \phi(x)\,\mathrm{d}x \leq \frac{1}{N}\sum_{i=0}^{N-1} \phi\left(\frac{i}{N}\right) + \frac{1}{N}.$$

Proof. Since ϕ is monotonically increasing, we have for all $0 \leq x_0 < x < x_1 \leq 1$ that $\phi(x_0) \leq \phi(x) \leq \phi(x_1)$. In particular, if $x_0 = \frac{i}{N}$ and $x_1 = \frac{i+1}{N}$ for some $i \in \{0, ..., N-1\}$, then

$$\frac{1}{N}\phi(x_0) \leq \int_{x_0}^{x_1} \phi(x)\,\mathrm{d}x \leq \frac{1}{N}\phi(x_1) = \frac{1}{N}\phi(x_0) + \frac{1}{N}[\phi(x_1) - \phi(x_0)].$$

By summing over all i, we obtain

$$\frac{1}{N}\sum_{i=0}^{N-1} \phi\left(\frac{i}{N}\right) \leq \int_0^1 \phi(x)\,\mathrm{d}x \leq \frac{1}{N}\sum_{i=0}^{N-1} \phi\left(\frac{i}{N}\right) + \frac{1}{N}\sum_{i=0}^{N-1}\left[\phi\left(\frac{i+1}{N}\right) - \phi\left(\frac{i}{N}\right)\right].$$

The last term is a telescopic sum yielding $\phi(1) - \phi(0) \leq 1$, which completes the proof. □

Appendix B

Complementary proofs

B.1 General inequalities

B.1.1 An inequality about logarithms

Lemma B.1. For $x \geq 0$, we have

$$\log_d(1+x) \geq \log_d(e)\frac{x}{1+x}.$$

Proof. As $\log_d(x)$ is an antiderivative of $\log_d(e)1/x$ and $\log(1) = 0$, we have

$$\log_d(1+x) = \int_1^{1+x} \log_d(e)\frac{1}{t}\,\mathrm{d}t \geq \int_1^{1+x} \log_d(e)\frac{1}{1+x}\,\mathrm{d}t = \log_d(e)\frac{x}{1+x}.$$

□

B.1.2 An inequality about power functions

Proposition B.2. If $p \neq 0$ and $n \geq 2$, we have

$$(1+p)^n > 1 + np$$

Proof. We proceed by induction on n. For $n = 2$, we have

$$(1+p)^2 = 1 + 2p + p^2 > 1 + p.$$

Suppose the statement holds for $n - 1$. Then

$$\begin{aligned}(1+p)^n &= (1+p)(1+p)^{n-1}\\ &> (1+p)(1+(n-1)p)\\ &= 1 + np + (n-1)p^2 > 1 + np,\end{aligned}$$

as desired. □

B.1.3 A power identity

Proposition B.3.

$$\forall n \in \mathbb{N}, \forall a, b \in \mathbb{R}, a^n - b^n = (a-b)\sum_{i=0}^{n-1} a^i b^{n-1-i}.$$

Proof. We proceed by induction on n. For $n = 1$, we have $a-b = (a-b)\sum_{i=0}^{0} a^i b^n - 1 - i = a - b$, so the statement holds.

Suppose that the statement holds for $n-1$. Then

$$\begin{aligned}(a-b)\sum_{i=0}^{n-1} a^i b^{n-i-1} &= b(a-b)\sum_{i=0}^{n-2} a^i b^{n-i-2} + (a-b)a^{n-1} \\ &= b(a^{n-1} - b^{n-1}) + a^n - a^{n-1}b \\ &= a^n - b^n\end{aligned}$$

□

B.2 Correlation inequalities

B.2.1 Proof of Lemma 2.24

Lemma 2.24. Let $A, B \in \mathbb{R}$ be random variables and $a, b, \bar{a}, \bar{b} \in \mathbb{R}$ fixed numbers such that $A \geq a$ and $B \leq b$ always, and $\mathbb{E}[A] = \bar{a}$ and $\mathbb{E}[B] = \bar{b}$. Then

$$\mathbb{E}[A \cdot B] \leq a\bar{b} + b\bar{a} - ab.$$

Proof. We can write

$$\mathbb{E}[A \cdot B] = \mathbb{E}[(A-a) \cdot B] + a\,\mathbb{E}[B]$$

and then use $B \leq b$ and $A \geq a$ to obtain

$$\mathbb{E}[A \cdot B] \leq b\,\mathbb{E}[A-a] + aE[B] = b\bar{a} - ba + a\bar{b},$$

as claimed. □

B.3 Proof of Lemma 4.12

The following lemma is proven in [6].

Lemma 4.12. Let X, T, T', U, V be finite vectors of real random variables, all independent, taking only finitely many values. Suppose T and T' are over $\mathbb{R}^n$ and have the same distribution. Let $f, g : \mathbb{R}^n \to \mathbb{R}$ be monotonically increasing functions, and let $h : \mathbb{R} \to \mathbb{R}$ be a concave function. Then

$$\mathbb{E}[h(f(X,T,U) + g(X,T,V))] \leq \mathbb{E}[h(f(X,T,U) + g(X,T',V))].$$

Proof. Observe first that if the lemma holds for any particular choice of X, U and V and not only when taking expectation over these three vectors, the lemma follows. We prove this by substituing arbitrary concrete real vectors for X, U and V, and consequently we obtain functions f' and g' that depend only on Y and are still monotone. Hence, it suffices to prove that

$$\mathbb{E}[h(f(Y)+g(Y))] \leq \mathbb{E}[h(f(Y)+g(Y'))] \tag{B.1}$$

where Y and Y' are over $\mathbb{R}^n$ and $f, g : \mathbb{R}^n \to \mathbb{R}$ are monotonically increasing functions.

Second, we claim that it suffices to examine the case $n = 1$. If we can prove the lemma for $n = 1$, we can repeatedly replace every component Y_i of Y by Y_i', eventually obtaining the above inequality. Hence, let us assume $n = 1$, and Y, Y' independent identically distributed real random variables. By symmetry, it holds that $\mathbb{E}[h(f(Y)+g(Y))] = \mathbb{E}[h(f(Y')+g(Y'))]$, and $\mathbb{E}[h(f(Y)+g(Y'))] = \mathbb{E}[h(f(Y')+g(Y))]$. Multiplying equation (B.1) by 2 and using these identities, we obtain:

$$\begin{aligned}
& \mathbb{E}[h(f(Y)+g(Y))] \leq \mathbb{E}[h(f(Y)+g(Y'))] \\
\Leftrightarrow\quad & 2\,\mathbb{E}[h(f(Y)+g(Y))] \leq 2\,\mathbb{E}[h(f(Y)+g(Y'))] \\
\Leftrightarrow\quad & \mathbb{E}[h(f(Y)+g(Y))] + \mathbb{E}[h(f(Y')+g(Y'))] \leq \mathbb{E}[h(f(Y)+g(Y'))] + \mathbb{E}[h(f(Y')+g(Y))] \\
\Leftrightarrow\quad & \mathbb{E}[h(f(Y)+g(Y)) + h(f(Y')+g(Y'))] \leq \mathbb{E}[h(f(Y)+g(Y')) + h(f(Y')+g(Y))]
\end{aligned}$$

We now prove this last inequality, and for this we show that it holds not only in expectation, but for every particular choice of $Y, Y' \in \mathbb{R}$. Fix such a choice. Without loss of generality, we can assume that $Y \leq Y'$ and therefore $f(Y) \leq f(Y')$ and $g(Y) \leq g(Y')$, using motononicity. Second, we can assume without loss of generality that $f(Y) + g(Y') \leq f(Y') + g(Y)$. Writing $a := f(Y) + g(Y)$, $b := f(Y) + g(Y')$, $c := f(Y') + g(Y)$ and $d := f(Y') + g(Y')$, we see that $a \leq b \leq c \leq d$. The claimed inequality states that $h(a) + h(d) \leq h(b) + h(c)$. Divided by two, the statement becomes equivalent to saying that the midpoint of the longer line segment in Figure B.1 is below the one of the shorter. This follows immediately from h being a concave function. ☐

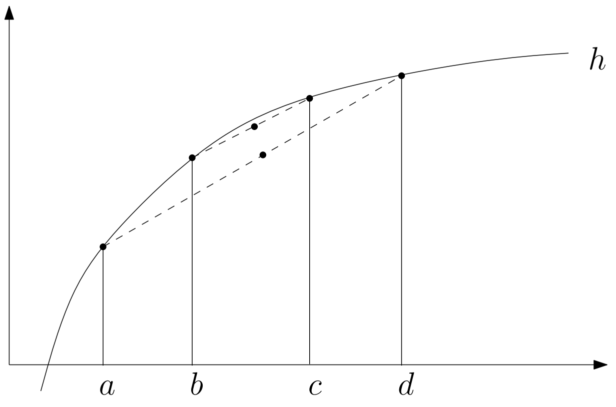


Figure B.1: Illustration of the final equality in the proof of Lemma 4.12.

B.4 Proof of Lemma 4.8

Lemma 4.8. Let A be a finite set and let $f : 2^A \to \mathbb{R}$ be a function that satisfies $f(B') \leq f(B)$, $\forall B' \subseteq B \subseteq A$.

Let $q' : A \to [0, 1]$ and $q : A \to [0, 1]$ be two functions that satisfy $q'(a) \leq q(a), \forall a \in A$.

At first, let $C_{q'}$ and C_q be the empty set. Now consider the two following random experiments:

1. $\forall a \in A$, choose $a \in C_{q'}$ with probability $q'(a)$.
2. $\forall a \in A$, choose $a \in C_q$ with probability $q(a)$.

Note that the events are independent.

Then we have $\mathbb{E}[f(C_{q'})] \leq \mathbb{E}[f(C_q)]$.

Proof. We define a set $C_{q''}$ such that $C_{q''} = C_{q'} \cup B$ where, for all $a \in A$, we choose $a \in B$ with probability $p(a) = \frac{q(a)-q'(a)}{1-q'(a)}$, independently of the other events.

Then the probability that a given x is in $C_{q''(a)}$ is

$$\begin{aligned}
\Pr[a \in C_{q''}] &= \Pr[a \in C_{q'} \vee a \in B] \\
&= \Pr[a \in C_{q'}] + \Pr[a \in B] - \Pr[a \in C_{q'} \wedge a \in B] \\
&= q'(a) + p(a) - p(a)q'(a) \\
&= p(a)(1 - q'(a)) + q'(a) \\
&= q(a(a))
\end{aligned}$$

Hence, $C_{q''}$ and C_q have the same probability distribution, so $\mathbb{E}[f(C_{q''})] = \mathbb{E}[f(C_q)]$. But then, $C_{q'} \subseteq C_{q''}$, so $\mathbb{E}[f(C_{q'})] \leq \mathbb{E}[f(C_{q''})] = \mathbb{E}[f(C_q)]$, which concludes the proof. $\square$

Bibliography

[1] Robert G Bartle. *The elements of integration and Lebesgue measure*, volume 92. Wiley-Interscience, 2011.

[2] Timon Hertli. 3-SAT faster and simpler – unique-SAT bounds for PPSZ hold in general. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 277–284. IEEE, 2011.

[3] Sebastian J. Millius. Towards a generalization of the PPSZ algorithm for large domains and multiple solutions. Master's thesis, Eidgenössische Technische Hochschule ETH Zürich, 2012.

[4] Ramamohan Paturi, Pavel Pudlák, Michael E Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *Journal of the ACM (JACM)*, 52(3):337–364, 2005.

[5] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 566–574. IEEE, 1997.

[6] Dominik Alban Scheder. *Algorithms and Extremal Properties of SAT and CSP*. PhD thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 19614, 2011.

[7] May Szedlák. The PPSZ algorithm for large domains. Bachelor thesis, Eidgenössische Technische Hochschule ETH Zürich, 2011.

[8] Emo Welzl. Boolean satisfiability – combinatorics and algorithms, 2013.