Computability Theory: Constructive Applications of the Lefthanded Local Lemma and Characterizations of some Classes of Cohesive Powers

Daniel Samir Mourad, Ph.D.

University of Connecticut, 2023

ABSTRACT

The Lovász local lemma (LLL) is a technique from combinatorics for proving existential results. There are many different versions of the LLL. One of them, the lefthanded local lemma, is particularly well suited for applications to two player games. There are also constructive and computable versions of the LLL. The chief object of this thesis is to prove an effective version of the lefthanded local lemma and to apply it to effectivise constructions of non-repetitive sequences.

The second goal of this thesis is to categorize some classes of cohesive powers. We completely describe both the isomorphism types of cohesive powers of equivalence structures and injection structures, as well as clarify the relationship between these cohesive powers and their original structures. We also describe the finite condensation of cohesive powers of computable copies of the integers as a linear order by cohesive sets whose complement are computably enumerable.

Finally, we investigate the possibility of decomposing problems in the Weihrauch degrees into a product of first order part and second order part. We give a preliminary result in this direction.

Computability Theory: Constructive Applications of the Lefthanded Local Lemma and Characterizations of some Classes of Cohesive Powers

Daniel Samir Mourad

B.S., University of Maryland College Park, 2015

M.Ed., University of Maryland College Park, 2018

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2023

APPROVAL PAGE

Doctor of Philosophy Dissertation

Computability Theory: Constructive Applications of the Lefthanded Local Lemma and Characterizations of some Classes of Cohesive Powers

Presented by

Daniel Samir Mourad, B.S., M.Ed.

Major Advisor _____
David Reed Solomon

Associate Advisor _____
Damir D. Dzhafarov

Associate Advisor _____
Iddo Ben-Ari

Associate Advisor _____
Thomas W. Roby

University of Connecticut

2023

ACKNOWLEDGMENTS

First of all, I thank Reed Solomon, my advisor. Without his guidance and support, it is safe to say that none of the work contained herein would have been possible. His guidance has been crucial for suggesting research directions and signposting the path forward. His perspective on writing have been incredibly important in developing my mathematical voice. Reed is also sympathetic in his advising and has always supported my academic interests. In short, Reed has been my ideal supervisor.

I also thank the faculty at UConn, who collectively have shared with me an immeasurable wealth of knowledge and experience. In particular, I thank the members of my thesis committee, Damir Dzhafarov, Iddo Ben-Ari, and Tom Roby, for their mentorship and their helpful comments on this thesis. Their guidance has been immensely beneficial and greatly enhanced my development as a scholar. I thank Dzhafarov especially for his steady encouragement and mentorship. His teaching and insights into both hard and soft aspects of the field have been invaluable to me. I thank Ben-Ari especially for being an amazing role model and for spearheading an exciting REU, empowering our students to conduct mathematical research. I thank Roby especially for his stimulating lectures and aspirational teaching style, as well as for the vital contribution of pointing out how to pronounce "Lovász"! In addition, I thank Masha Gordina, Dave McArdle, Fabiana Cardetti, Katherine Hall, and Myron Minn-Thu-Aye for being caring people and for generally having a positive impact on the community in the UConn math department.

I thank the mathematicians I have discussed this work with and who have given me the opportunity to present this work, including but not limited to Aidan Backus, Heidi Benham, Anton Bernshteyn, Vasco Brattka, Wesley Calvert, Douglas Cen-

zer, Andrew DeLapo, Johanna Franklin, Elliot Glazer, Valentina Harizanov, Denis Hirschfeldt, Noah Hughes, Corrie Ingall, Waseet Kazmi, Karen Lange, Tyler Markkanen, Joe Miller, David Nichols, Brian Pinsky, Isabella Scott, Phillip Scowcroft, Gihanee Senadheera, Forte Shinko, Teerawat Thewmorakot, Robin Tucker-Drob, and Java Villano. I thank Ileana Vasu for her exceptional perspectives and mentorship on student centered instruction and inquiry based learning. I thank Monique Roy and Rachel D'Antonio for all of their amazing work in keeping the math department running. I also thank mentors who inspired my interest in mathematical research, including Wojtek Czaja and Chris Laskowski. I also thank William Rose, whose creative way of teaching logic sparked my passion for the subject early on.

I thank friends from home, including but not limited to Adam, Aiden, Ameen, Arizona, Ashe, Ben, Bui, Channing, Daniel, Egan, Franky, Henok, Hillel, Mikhail, Nguyen, and William for being trustworthy, lifelong companions. I also thank the friends I met in Connecticut, including Adriana, Aniruddha, Antigoni, Anurag, Devika, Erin, Evelyn, Gianmarco, John, Jonathan, Matt, Mike, Noor, Paul, Surya, Treena, Tulika, Utsav, Vinayak and many others for all of the encouragement and fun times through the past five years.

I would be living a different life without the love and encouragement of my family. I thank my parents, Teresa and Joe, for the opportunities and guidance they have given me in life. I also thank my father for his indispensable mathematical guidance. I thank my brother, Benjamin; my grandparents, Marlene and the late Samir; I also thank my aunts and uncles: Nadia, Pierre "The Great and the Magnificent", Adaleine, Noha, Mona, Paul, Barry, Mark; and my cousins: Andrew, Matthew, Elizabeth, Nicholas, Megan, Michael, and William, who have all supported me throughout the years. Finally, I thank my partner, Stuti, for her endless supply of strength and love.

Contents

Chapter 1

Lefthanded Computable Lovász Local Lemma

1.1 Introduction

The Lovász local lemma (LLL) (Theorem 1.2.3) is a technique from combinatorics, specifically from the probabilistic method. The LLL generalizes the following fact. Note that, in this document, we write $\bar{A}$ for the complement (rather than the closure) of a set A.

Proposition 1.1.1. *Let $\mathcal{A}$ be a finite set of mutually independent events on some probability space Ω. If $\Pr(\bar{A}) > 0$ for each $A \in \mathcal{A}$, then there is positive probability that the $A \in \mathcal{A}$ are simultaneously false, i.e.*

$$\Pr\left(\bigcap_{A\in\mathcal{A}} \bar{A}\right) > 0.$$

In particular, the set of outcomes $\bigcap_{A\in\mathcal{A}} \bar{A}$ *making all of the* $A \in \mathcal{A}$ *false is non-empty.*

Proof. Since the events in $\mathcal{A}$ are mutually independent, we have that

$$\Pr\left(\bigcap_{A\in\mathcal{A}} \bar{A}\right) = \prod_{A\in\mathcal{A}} \Pr(\bar{A}).$$

The condition that $\Pr(\bar{A}) > 0$ for each $A \in \mathcal{A}$ give us

$$\prod_{A\in\mathcal{A}} \Pr(\bar{A}) > 0.$$

Only empty sets have probability 0, so we have that $\bigcap_{A\in\mathcal{A}} \bar{A}$ is nonempty. □

The LLL allows the possibility of dependence between events in $\mathcal{A}$ in exchange for stricter requirements on their probabilities. The original formulation of the LLL by Erdős and Lovász [16] has in common with Proposition 1.1.1 that both apply only when $\mathcal{A}$ is finite. However, when used in combinatorics, the non-emptiness of $\bigcap_{A\in\mathcal{A}} \bar{A}$ is the main point of interest. In this context, both Proposition 1.1.1 and the LLL can be made to generalize to infinite sets $\mathcal{A}$ of events if Ω is compact and each $A \in \mathcal{A}$ is open (so that $\bar{A}$ is closed).

However, in many cases of interest, this use of compactness is non-constructive. In fact, the original proof of the LLL is non-constructive for finite objects as well. In cases where Ω is finite, Erdős and Lovász's proof of the LLL gives no hint as to how to find an element of $\bigcap_{A\in\mathcal{A}} \bar{A}$ in a way faster than a brute force search. This problem was of great interest in combinatorics and computer science [4, 2, 23, 9, 28]. In a seminal result, Moser and Tardos [24] give a simple and efficient constructive version of the LLL (Theorem 1.3.1). The algorithm of Moser and Tardos has been used as a foundation for many improvements, [18, 17, 25, 21, 1, 20], including a

computable version by Rumyantsev and Shen [27]. This effective version has been used for multiple applications in computability theory and reverse math [19, 22, 8].

In this chapter, we modify the Moser–Tardos algorithm to be more broadly applicable, both in the finite case and in computability theory. We use the modified algorithm to effectivise previously non-constructive theorems of Beck [5] (Theorem 1.2.1), and Alon, Spencer, and Erdős [3] (Theorem 1.2.2) on the existence of certain types of non-repetitive sequences. We also effectivise Pegden's [26] application of a generalization of the LLL called the *lefthanded Lovász local lemma* (LLLL) to game versions of Theorems 1.2.1 and 1.2.2. The LLLL allows for stronger interdependence between the events in exchange for the existence of a partial order on the events with certain properties. Theorem 1.7.13 makes a similar improvement to the results of Moser and Tardos; and Rumyantsev and Shen, allowing stronger interdependence in exhcange for the existence of a linear order (a computable linear order in the computable version) with certain properties.

1.2 Non-repetitive Sequences and the Local Lemma

The Lovász local lemma (LLL) can be used to prove two closely related results about the existence of non-repetitive sequences. One of these results is by Beck [5] and the other is an exercise of Alon, Spencer, and Erdős [3]. They are stated below.

Theorem 1.2.1 (Beck [5])**.** *Given arbitrary small $\varepsilon > 0$, there is some N_ε and an infinite $\{0,1\}$-valued sequence $a_1, a_2, a_3, \ldots$ such that any two identical intervals $a_k, \ldots, a_{k+n-1}$ and $a_\ell, \ldots, a_{\ell+n-1}$ of length $n > N_\varepsilon$ have distance $\ell - k$ greater than $(2-\varepsilon)^n$.*

Theorem 1.2.2 (Alon, Spencer, and Erdős [3]). *Given arbitrary small $\varepsilon > 0$, there is some N_ε and an infinite $\{0,1\}$-valued sequence $a_1, a_2, a_3, \ldots$ such that any two adjacent intervals of length $n > N_\varepsilon$ differ in at least $(\frac{1}{2} - \varepsilon)n$ many places. That is, for each k and $n > N_\varepsilon$, $a_{k+i} \neq a_{k+n+i}$ for at least $(\frac{1}{2} - \varepsilon)n$ many i with $0 \leq i < n$.*

For $i_1 < i_2$, we say that intervals

$$[i_1, j_1) := a_{i_1}, a_{i_1+1}, \ldots, a_{i_j-1}$$

and $[i_2, j_2)$ are adjacent if $i_2 = j_1$. Their distance is $i_2 - i_1$. We now state the version of the Lovász local lemma that is used in the proof of Theorems 1.2.1 and 1.2.2. The statement of the local lemma uses the language of graph theory. In the following, we will work with both directed and undirected graphs. For a directed graph G and vertex $v \in G$, let $\Gamma(v) = \{s \in G : v \to s\}$ be the out-neighbor set of v. For undirected graph G with edge relation E and vertex $v \in G$, let $\Gamma(v) = \{s \in G : E(v,s)\}$ be the neighborhood (or neighbor set) of v. In both cases, let $\Gamma^+(v) = \Gamma(v) \cup \{v\}$. We will work with both finite and countable graphs.

Theorem 1.2.3 (Lovász Local Lemma, General Form). *Let $\mathcal{A}$ be a finite set of events in some probability space. Suppose there exists a directed graph G on $\mathcal{A}$ and a real-valued function $z : \mathcal{A} \to (0,1)$ such that, for each $A \in \mathcal{A}$,*

1. *A is mutually independent from $\mathcal{A} \setminus \Gamma^+(A)$ and*

2. $$\Pr(A) \leq z(A) \prod_{B \in \Gamma(A)} (1 - z(B))$$

Then,

$$\Pr(\bigcap_{A\in\mathcal{A}} \bar{A}) \geq \prod_{A\in\mathcal{A}} (1 - z(A)) > 0.$$

We call a graph with property (1) a **dependency graph** for $\mathcal{A}$.

We now give the proofs of finite versions of Theorems 1.2.1 and 1.2.2. They imply the full versions using a compactness argument which we will subsequently give.

Theorem 1.2.4 (Finite version of Theorem 1.2.1)**.** *For each $\varepsilon > 0$, there is some N_ε such that for every M there is a $\{0,1\}$-valued sequence of length M such that any two identical intervals of length $n > N_\varepsilon$ have distance greater than $f(n) = (2-\varepsilon)^n$.*

Proof. Fix $\varepsilon > 0$ and $N > 0$. We will show that if N is large enough, then $N_\varepsilon = N$ witnesses the theorem. Randomly select a $\{0,1\}$-valued sequence $a_0, a_1, a_2, \ldots, a_M$ by independently assigning each bit a_i either 0 or 1, each with probability $\frac{1}{2}$. For each k, l, n, let $A_{k,l,n}$ be the event that the intervals

$$[k, k+n) = a_k, a_{k+1}, \ldots, a_{k+n-1}$$

and

$$[l, l+n) = a_l, a_{l+1}, \ldots, a_{l+n-1}$$

are identical. Fix $M \in \mathbb{N}$. Then, $\mathcal{A}$ is the set $\{A_{k,l,n} : l - k < f(n)$ and $N_\varepsilon < n < M$ and $1 \leq k < l \leq M - n\}$. Define a dependency graph G on $\mathcal{A}$ by putting an edge between $A_{k,l,n}$ and $A_{k',l',n'}$ if and only if the intersection of $([k, k+n-1] \cup [l, l+n-1])$ and $([k', k'+n'-1] \cup [l', l'+n'-1])$ is nonempty. Since events of disjoint sets of the a_i are independent, G is a dependency graph for $\mathcal{A}$. Clearly, $\Pr(A_{k,l,n}) = 2^{-n}$. Let

$z(A_{k,l,n}) = \frac{1}{f(n)n^3}$. We need to show that, for each $A_{k_0,l_0,n_0} \in \mathcal{A}$,

$$\Pr(A_{k_0,l_0,n_0}) = 2^{-n_0} \leq z(A_{k_0,l_0,n_0}) \prod_{A_{k,l,n} \in \Gamma(A_{k_0,l_0,n_0})} (1 - z(A_{k,l,n})).$$

For each k_0, l_0, n_0 there are at most $2(n + n_0)$ many intervals of length n that have non-empty intersection with $[k_0, k_0 + n_0 - 1] \cup [l_0, l_0 + n_0 - 1]$. For any given interval I, we have that $I = [k, k + n)$ or $I = [l, l + n)$ for at most $2f(n)$ many pairs (k, l) such that $l - k \leq f(n)$. Together with the previous observation, we conclude that $\{(k, l) : A_{k,l,n} \in \Gamma(A_{k_0,l_0,n_0})\}$ is of size at most $4(n + n_0)f(n)$ for each n. This, along with the definition of $z(A_{k,l,n})$ and the fact that $0 < 1 - z(A_{k,l,n}) < 1$ for all k, l, n, gives us that

$$z(A_{k_0,l_0,n_0}) \prod_{A_{k,l,n} \in \Gamma(A_{k_0,l_0,n_0})} (1 - z(A_{k,l,n})) \geq \frac{1}{f(n_0)n_0^3} \prod_{n \geq N} \left(1 - \frac{1}{f(n)n^3}\right)^{4(n_0+n)(f(n))}.$$

Since $(1 - \frac{a}{x})^x \geq (1 - a)$ for $0 < a < 1$ and $x \geq 1$, the right hand side is greater than or equal to

$$\begin{aligned} &\frac{1}{f(n_0)n_0^3} \prod_{n \geq N} \left(1 - \frac{1}{n^3}\right)^{4n_0} \left(1 - \frac{1}{n^2}\right)^4 \\ =&\frac{1}{f(n_0)n_0^3} \left(\prod_{n \geq N} \left(1 - \frac{1}{n^3}\right)\right)^{4n_0} \left(\prod_{n \geq N} \left(1 - \frac{1}{n^2}\right)\right)^4. \end{aligned} \tag{1.2.5}$$

For $0 < a_n < 1$, $\prod_{n \geq N}(1 - a_n) \geq 1 - \sum_{n \geq N} a_n$. So, Line 1.2.5 is greater than or equal

to

$$\frac{1}{f(n_0)n_0^3}\left(1-\sum_{n\geq N}\frac{1}{n^3}\right)^{4n_0}\left(1-\sum_{n\geq N}\frac{1}{n^2}\right)^4. \tag{1.2.6}$$

For $N \geq 2$, the two sums in Line 1.2.6 are bounded above by $\frac{1}{N-1}$, so Line 1.2.6 is greater than or equal to

$$\frac{(2-\varepsilon)^{-n_0}}{n_0^3}\left(1-\frac{1}{N-1}\right)^{4n_0+4}.$$

If N is large enough, this is greater than or equal to 2^{-n_0}, fulfilling the conditions of the Local Lemma. □

Theorem 1.2.7 (Finite version of Theorem 1.2.2)**.** *For each arbitrary small $\varepsilon > 0$ there is some N_ε such that for each M there is a $\{0,1\}$-valued sequence $a_1, a_2, a_3, ..., a_M$ of length M such that any two adjacent intervals of length $n > N_\varepsilon$ differ in at least $(\frac{1}{2}-\varepsilon)n$ many places. That is, for each k and $n > N_\varepsilon$, $a_{k+i} \neq a_{k+n+i}$ for at least $(\frac{1}{2}-\varepsilon)n$ many i with $0 \leq i < n$.*

Proof. Fix $\varepsilon > 0$. As before, randomly select an infinite $\{0,1\}$-valued sequence $a_0, a_1, a_2, \ldots$ by independently assigning each bit a_i either 0 or 1, each with probability $\frac{1}{2}$. For each $k, n \in \mathbb{N}$, let $A_{k,n}$ be the event that blocks $a_k, a_{k+1}, ..., a_{k+n-1}$ and $a_{k+n}, a_{k+n+1}, ..., a_{k+2n-1}$ share at least $(\frac{1}{2}+\varepsilon)n$ many entries. The probability that $[k, k+n)$ shares *exactly* r entries with $[k+n, k+2n)$ is $2^{-n}\binom{n}{r}$, so

$$\Pr(A_{k,n}) = 2^{-n}\sum_{r=\lceil(\frac{1}{2}+\varepsilon)n\rceil}^{n}\binom{n}{r}.$$

Since the greatest of the $\binom{n}{r}$ is $\binom{n}{\lceil(\frac{1}{2}+\varepsilon)n\rceil}$,

$$\Pr(A_{i,b}) \leq n2^{-n}\binom{n}{\lceil(\frac{1}{2}+\varepsilon)n\rceil}.$$

It is known that there is an $\alpha < 1$ such that $\binom{N}{\lceil(\frac{1}{2}+\varepsilon)N\rceil} < (\alpha 2)^N$ for large enough N, so there is N_ε such that for $n > N_\varepsilon$,

$$\Pr(A_{i,n}) < n2^{-n}(\alpha 2)^n = n\alpha^n,$$

which has limit 0 as $n \to \infty$.

Let $z_{i,n} = \frac{b^n}{n}$ for some b such that $\alpha^{1/3} < b < 1$. Fix A_{i_0,n_0}. We have that $\Gamma(A_{i_0,n_0}) = \{A_{i,n} : i_0 - n + 1 \leq i \leq i_0 + 2n_0 - 1\}$, so it suffices for the Lovász local lemma to check that

$$\frac{b^n}{n}\prod_{n=N_\varepsilon}^{\infty}\prod_{i=i_0-n+1}^{i_0+2n_0-1}\left(1-\frac{b^n}{n}\right) \geq n_0\alpha^{n_0}.$$

for large enough N_ε. The left-hand side is equal to

$$\begin{aligned}
\frac{b^{n_0}}{n_0}\prod_{n=N_\varepsilon}^{\infty}\left(1-\frac{b^n}{n}\right)^{2n_0+2n-2} &\geq \frac{b^{n_0}}{n_0}\left(\prod_{n=N_\varepsilon}^{\infty}\left(1-\frac{b^n}{n}\right)^{2n_0}\right)\left(\prod_{n=N_\varepsilon}^{\infty}\left(1-\frac{b^n}{n}\right)^{2n}\right)\\
&\geq \frac{b^{n_0}}{n_0}\left(\prod_{n=N_\varepsilon}^{\infty}(1-b^n)^{2n_0}\right)\left(\prod_{n=N_\varepsilon}^{\infty}(1-b^n)^{2}\right)\\
&\geq \frac{b^{n_0}}{n_0}\prod_{n=N_\varepsilon}^{\infty}(1-b^n)^{2n_0+2}\\
&\geq \frac{b^{n_0}}{n_0}\left(1-\sum_{n=N_\varepsilon}^{\infty}b^n\right)^{2n_0+2}\\
&= \frac{b^{n_0}}{n_0}\left(1-\frac{b^{N_\varepsilon}}{1-b}\right)^{2n_0+2}\\
&\geq \frac{b^{n_0}}{n_0}(b^{2n_0+2})\\
&= \frac{b^{3n_0+2}}{n_0}\\
&\geq n_0\alpha^{n_0},
\end{aligned}$$

where the last three lines are for n_0 large enough (adjust N_ε accordingly). □

The full proofs of Theorems 1.2.1 and 1.2.2 are obtained by the following compactness argument. Fix $\varepsilon > 0$. Say that sequences and finite strings with the properties promised by Theorems 1.2.1 and 1.2.2 for a given $\varepsilon > 0$ are "good". By the local lemma argument from the finite versions of the theorems, good strings exist of each length M. Because any pair of intervals demonstrating that a given string σ is not good are also present in any extension of σ, we see that initial segments of good strings and sequences are also good. Therefore, the good strings form a binary tree T under the initial segment partial order. By weak König's lemma, T has a path X. To see that X is good, suppose that it is not. Then there is a pair of blocks of X

witnessing that X is not good. These blocks are finite, so there is N such that $X|_N$ contains them. Then, $X|_N$ is not good, contradicting that $X|_N \in T$.

It is natural to ask for the computational complexity of such sequences. For strings σ, it is uniformly computable to determine whether σ is good. Since the good σ form a tree T, the paths through this tree form a Π^0_1 class. As we saw above, the paths through T are all good. Thus, by the respective basis theorems, there is a good sequence of each of low, hyperimmune-free, and c.e.-degree.

However, this compactness argument does not show that there is a computable good sequence. Indeed, many Π^0_1 classes do not contain a computable element. Instead, this question is answered by a computable version of an extension of the local lemma. Rumyantsev and Shen [27] first gave a computable version of Theorem 1.2.3. Their proof, which is reproduced in Sections 1.5 and 1.6 closely follows the constructive version of the local lemma given by Moser and Tardos [24], which is reproduced in Section 1.6. For reasons we describe in Section 1.7, the move from finite strings to infinite sequences in the application of Theorem 1.2.3 introduce complications in applying the computable LLL of Rumyantsev and Shen [27]. Instead, we introduce an extension of the computable LLL inspired by the lefthanded local lemma introduced by Pegden [26].

1.3 The Constructive Local Lemma

The Moser–Tardos algorithm, also known as the resample algorithm, applies to a special case of the local lemma known as the *variable case*. Let $\mathcal{X} = \{x_1, x_2, ..., x_t\}$ be a finite set of independent random variables with finite ranges and rational-valued

probability distributions. Let $\mathcal{A}$ be a finite set of events with each $A \in \mathcal{A}$ determined by a finite set $\mathrm{VBL}(A) \subset \mathcal{X}$ of random variables from $\mathcal{X}$. We call $\mathcal{X}$ a *variable context* for $\mathcal{A}$. For example, each $x_i \in \mathcal{X}$ might be a fair coin toss. The variable context induces the following natural dependency graph for applying the Local Lemma. Let G be the undirected graph defined on $\mathcal{A}$ where, for each $A, B \in \mathcal{A}$, there is an edge from A to B if and only if $\mathrm{VBL}(A) \cap \mathrm{VBL}(B) \neq \emptyset$ and $A \neq B$. Let $\Gamma(A) = \{B \in \mathcal{B} :$ there is an edge between A and $B\}$. Let $\Gamma^+(A) = \Gamma(A) \cup \{A\}$. If $S \subset \mathcal{A} \setminus \Gamma^+(A)$, then $\mathrm{VBL}(A) \cap \bigcup_{B \in S} \mathrm{VBL}(B) \neq \emptyset$, so A and $\mathcal{A} \setminus \Gamma^+(A)$ are mutually independent, satisfying Condition 1 of the local lemma (Theorem 1.2.3).

Moser and Tardos [24] showed that the resample algorithm efficiently constructs witnesses to applications of the LLL in a variable context. The resample algorithm proceeds as follows. Fix an enumeration $A_1, A_2, A_3, ..., A_r$ of $\mathcal{A}$.

- Stage 0: Begin with a random sample of the values for each $x \in \mathcal{X}$. In the coin analogy, this would mean flipping each coin once.

- Stage $n + 1$: Check if the events in $\mathcal{A}$ are all false at the end of stage n. If they are all false, terminate and return the current valuation of $\mathcal{X}$. If any are true, let k be least such that A_k is true at the end of stage n. Then, select new independent random samples for all $x \in \mathrm{VBL}(A_k)$. In the coin analogy, we would toss a new coin for each $x \in \mathrm{VBL}(A)$ and set the value of x to the result of the coin toss.

We restate the theorem with reference to efficiency limited to what is useful in the proof of Theorem 1.4.2.

Theorem 1.3.1 ([24]). *Suppose the set of events $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ with dependency graph G induced by variable context $\mathcal{X}$ satisfy the conditions of the Lo-*

cal Lemma. Let τ_n be the first stage of the resample algorithm at which each of $A_1, A_2, ..., A_n$ is false. Then,

$$\begin{aligned}\mathbb{E}(\tau_n) &< \sum_{A \in \mathcal{A}} \frac{z(A)}{1 - z(A)} \\ &< \infty\end{aligned}$$

for each n.

We reproduce the proof of Theorem 1.3.1 below because the proof of Theorem 1.7.13 uses many of the same techniques.

During the resample algorithm, we produce a *log* $E_1, E_2, E_3, ...$ of resampled events, where $E_i \in \mathcal{A}$ is the event resampled at stage i. When analyzing the log, we are interested in which entries of the log were relevant in the resampling of later entries of the log. We track this information using *Moser trees*, also known as *witness trees*. For a tree T with root r and vertex $v \in T$, we say that the *depth* of v is length of the shortest path between v and r.

Definition 1.3.2 (Moser tree)**.** Fix an initial segment $E_1, E_2, ..., E_n$ of the log generated by a run of the resample algorithm. The **Moser tree** T associated with $E_1, E_2, ..., E_n$ is a finite tree with labels in $\mathcal{A}$ constructed as follows.
We construct T in stages, beginning at stage n and ending at stage 1. At each stage i, we define T_i, setting $T = T_1$. For a vertex in $x \in T_i$, let $[x] \in \mathcal{A}$ denote the label of x.

- Stage n: let T_n be the tree with one vertex labeled E_n.
- Stage $k-1$: check whether there is some $x \in T_k$ such that $E_{k-1} \in \Gamma^+([x])$. If there is not, let $T_{k-1} = T_k$. If there is, pick such an $x \in T_k$ of maximal depth.

If there are multiple vertices of maximal depth, one may be chosen arbitrarily. Let T_{k-1} be T_k with a new vertex labeled E_{k-1} as a child of x.

If T is the Moser tree associated with some initial segment of the log generated by a run of the resample algorithm, then we say that this run of the resample algorithm **produces** T.

For $x \in T$, let $d(x)$ be the depth of x in T. Let $q(x)$ be the stage of the Moser–Tardos algorithm at which x was added to T. To be precise, $q(x)$ is the unique stage i such that $x \in T_i$ but $x \notin T_{i+1}$.

We give some properties of Moser trees:

Proposition 1.3.3. *Fix a run of the resample algorithm. Let $E_1, E_2, ..., E_n$ be an initial segment of the log. Then, the Moser tree T associated with $E_1, E_2, ..., E_n$ has the following properties.*

1. *For each $x, y \in T$, if $[x] \in \Gamma^+([y])$ and $x \neq y$, then $d(x) \neq d(y)$. Furthermore, under the same conditions, we also have that $d(x) > d(y)$ if and only if $q(x) < q(y)$.*

2. *T is not the Moser tree associated with $E_1, E_2, ..., E_m$ for any $m \neq n$.*

Proof. To see (1), suppose that $x \neq y$ and $[x] \in \Gamma^+([y])$. By construction, only one vertex can be added to the tree at each stage, so $q(x) \neq q(y)$. Because the neighborhood relation is symmetric, $[y] \in \Gamma^+([x])$. Thus, without loss of generality, we can assume that $q(x) < q(y)$. Then, when x is added to $T_{q(x)}$, y is already in $T_{q(x)+1}$. Since x is added as a child of a maximal depth node $z \in T_{q(x)+1}$ with $[x] \in \Gamma^+([z])$, we must have that $d(z) \geq d(y)$. So $d(x) \geq d(y) + 1$, and in particular $d(x) > d(y)$,

proving the first part of the statement. By the symmetrical argument, $q(x) > q(y)$ implies that $d(x) < d(y)$, proving the second part of the statement.

To see (2), let $r \in T$ be the root vertex. If T is the Moser tree associated with $E_1, E_2, ..., E_m$ then $E_m = E_n = [r]$. We also have that $[r] \in \Gamma^+([r])$, so the number of nodes labeled $[r]$ is equal to the number of instances of $[r]$ in the initial segment(s) that T is associated with. Thus, $E_1, E_2, \ldots, E_m$ has the same number of entries with value $[r]$ as $E_1, E_2, \ldots E_n$. This can only happen if $n = m$. □

Note that part (1) of Proposition 1.3.3 also implies that each event appears at each depth of T at most once.

Next we bound the probability that some initial segment of the log is associated with any given T. This is done via coupling with the following random process.

Definition 1.3.4 (T-check)**.** Fix a finite labeled tree T with labels from $\mathcal{A}$. Fix an ordering, $v_1, v_2, ..., v_k$, of the vertices of T with non-increasing depth. The **T-check** is the random process that proceeds as follows.

At stage i for $1 \leq i \leq k$, take a random, independent valuation of the variables in $\mathrm{VBL}([v_i])$ according to their distributions and check if this valuation makes $[v_i]$ true. The T-check passes if, for all j with $1 \leq j \leq k$, $[v_j]$ is found true at stage j.

Because the valuations at each stage are independent, the probability that the T-check passes is exactly $\Pi_{v \in T} \Pr([v])$.

For our purposes, the ordering of the vertices of equal depth does not matter. However, in the T-check used for the proof of Theorem 1.7.13, the ordering of the vertices will be completely determined by the labels of T.

Fixing the result of every resampling of the variables in $\mathcal{X}$ prior to running the algorithm facilitates the coupling argument by allowing us to record the number of

times each variable has been resampled at each stage of both the T-check and the resample algorithm. For the computable local lemma, we will need $\mathcal{X}$ to be countable, so we allow it in the following definition.

Definition 1.3.5 (Random Source)**.** Fix a finite (or countable) set of independent random variables $\mathcal{X} = x_1, x_2, x_3, \ldots$. A **random source** for $\mathcal{X}$ is a function $S(x, y)$ such that for all x_i and $j \in \omega$, $S(i, j) \in \text{range}(x_i)$. We typically denote $S(i, j)$ by x_i^j. We think of x_i^0 as the initially sampled value of x_i and, for $j > 0$, x_i^j as the value of x_i after the j'th resampling.

From now on, the statement about expectation in the local lemma (Theorem 1.3.1) will be interpreted in the probability space of all random sources with probability measure generated by taking each x_i^j as an independent random sample of x_i.

In a random source, the entry x_i^j represents the result of the j'th resampling of x_i. For each $x_i \in \mathcal{X}$ and stage n, we record $V_n(i)$ to be the number of times the resample algorithm has resampled x_i at the beginning of stage n. Fixing a random source for $\mathcal{X}$, the following deterministic restatement of the resample algorithm provides a more precise definition of $V_n(i)$.

Definition 1.3.6 (Deterministic Resample Algorithm)**.** The deterministic resample algorithm proceeds in stages, as follows.

- Stage 0: Let $V_0(i) = 0$ for all $x_i \in \mathcal{X}$.
- Stage $n+1$: Check if the events of $\mathcal{A}$ are all false under the valuation $x_i = x_i^{V_n(i)}$. If they are all false, terminate and return the valuation $x_i^{V_n(i)}$ for each x_i. If any are true, let k be least such that A_k is true under the valuation $x_i = x_i^{V_n(i)}$.

Then, define V_{n+1} by

$$V_{n+1}(i) = \begin{cases} V_n(i) & \text{if } x_i \notin \text{VBL}(A_k) \\ V_n(i) + 1 & \text{if } x_i \in \text{VBL}(A_k). \end{cases}$$

We also record $\widehat{V}_n(i)$ to be the number of times the T-check has resampled x_i at the beginning of stage n. As above, a more precise definition of $\widehat{V}_n(i)$ can be found by stating the T-check deterministically using a random source, as follows.

Definition 1.3.7 (Deterministic T-check)**.** Fix a rooted labeled tree T with labels from $\mathcal{A}$ and a random source $\{x_i^j\}$ for $\mathcal{X}$. The **T-check** is the random process that proceeds as follows.

- Fix an ordering, $v_1, v_2, ..., v_k$, of the vertices of T with non-increasing depth.
- Stage 0: Let $\widehat{V}_0(i) = 0$ for all $x_i \in \mathcal{X}$.
- Stage $n+1$: Check if $[v_{n+1}]$ is true under the valuation $x_i = x_i^{\widehat{V}_n(i)}$. Then, define $\widehat{V}_{n+1}$ by

$$\widehat{V}_{n+1}(i) = \begin{cases} \widehat{V}_n(i) & \text{if } x_i \notin \text{VBL}([v_{n+1}]) \\ \widehat{V}_n(i) + 1 & \text{if } x_i \in \text{VBL}([v_{n+1}]). \end{cases}$$

The T-check passes if, for all j with $1 \leq j \leq k$, v_j is found true at stage j.

Suppose that a Moser tree is generated by the resample algorithm. The next proposition shows that the resample algorithm and the T-check have resampled the relevant variables an equal number of times when checking each vertex of T.

Proposition 1.3.8. *Suppose that the Moser tree T is associated with an initial segment $E_1, E_2, ..., E_m$ of the log of the resample algorithm run on a random source S. Let $v_1, v_2, ..., v_r$ be the ordering of the vertices of T used by the T-check. For each n, let V_n and $\widehat{V}_n$ be the records of resampling used by the resample algorithm and T-check, respectively, at stage n. Then, for each stage $n \leq m$ and $x_j \in \mathrm{VBL}([v_n])$,*

$$\widehat{V}_n(j) = V_{q(v_n)}(j).$$

Proof. Fix stage $n \leq m$ and $x_j \in \mathrm{VBL}([v_n])$. Let $P = \{k < q(v_n) : x_j \in \mathrm{VBL}(E_k)\}$. Let $P^T = \{v_k \in T : x_j \in \mathrm{VBL}([v_k]) \text{ and } k < n\}$. By definition of the resample algorithm and T-check, respectively, $V_{q(v_n)}(j) = |P|$ and $\widehat{V}_n(j) = |P^T|$. It remains to show that $|P| = |P^T|$. We claim that q is a bijection from P^T to P.

To see this, first note that q is injective because at most one vertex is added at each stage of constructing the Moser tree associated with $E_1, \ldots, E_m$. It remains to show that $q(P^T) \subset P$ and that $P \subset q(P^T)$.

Fix $k \in q(P^T)$ witnessed by $q(v_s) = k$ for some $v_s \in P^T$. Then, $E_k = [v_s]$ and $x_j \in \mathrm{VBL}([v_s])$, so $x_j \in \mathrm{VBL}(E_k)$. We now show that $k < q(v_n)$. $k \neq q(v_n)$ because q is injective. The fact that $s < n$ tells us that the T-check considers v_s before v_n and therefore that $d(v_s) \geq d(v_n)$ by definition of the T-check. The events $[v_s]$ and $[v_n]$ are neighbors, so by Proposition 1.3.3, their respective nodes cannot have the same depths on T, so $d(v_s) > d(v_n)$. By Proposition 1.3.3, $q(v_n) > q(v_s) = k$, giving the final property required to conclude that $k \in P$. Thus, $q(P^T) \subset P$.

Fix $k \in P$. We have $k < q(v_n)$ and $E_k \in \Gamma^+([v_n])$, so at stage k of constructing the Moser tree a node $v_s \in T$ is added with $q(v_s) = k$. Thus, $k \in q(T)$. Since $k = q(v_s) < q(v_n)$ it follows by Proposition 1.3.3 that $d(v_s) > d(v_n)$. Then, by

definition of the T check, $s < n$. Since $x_j \in E_k = [v_s]$, we have that $v_s = q^{-1}(k) \in P^T$. Thus, $P \subset q(P^T)$.

This shows that q is a bijection from P^T to P, so $|P^T| = |P|$ and hence $\widehat{V}_i(j) = V_{q(v_i)}(j)$. □

Now we can compare the T-check to the probability of producing T.

Proposition 1.3.9. *Fix a Moser tree T. The probability that the resample algorithm produces T is less than or equal to the probability that the T-check passes. Thus,*

$$\Pr(\textit{the resample algorithm produces } T) \leq \prod_{v \in T} \Pr([v]).$$

Proof. It suffices to show that, for any random source, the T-check passes if the resample algorithm produces T. Fix random source $\{x_i^j\}$ and suppose that the resample algorithm produces T when run on this random source. Fix an ordering, $v_1, v_2, ..., v_k$, of the vertices of T with non-increasing depth. By Proposition 1.3.8, we have that $\widehat{V}_i(j) = V_{q(v_i)}(j)$ for each $v_i \in T$ and $j \in \mathrm{VBL}([v_i])$. Thus, the T-check and resample algorithm always use the same variables to check if $[v_i]$ is true for each i. □

Each stage of the T-check uses an independent set of variables, so the probability that the T-check passes is $\prod_{v \in T} \Pr([v])$. For $A \in \mathcal{A}$ let $\mathcal{T}_A$ be the set of all possible Moser trees with root node labeled A. We use a Galton–Watson type process to bound $\sum_{T \in \mathcal{T}_A} \prod_{v \in T} \Pr([v])$. The process constructs a labeled tree, beginning with a single root vertex r labeled $[r] = A$ at stage 0 with $d(r) = 0$. At stage $s + 1$, the Galton–Watson process checks if there are any vertices of depth equal to s. If there are, then for each pair (v, E) of vertex v with $d(v) = s$ and $E \in \Gamma^+[v]$, it randomly chooses one of two things: with probability $z(E)$, it adds a new vertex to the tree as

a child of v and labels this new vertex E. If it does not do this, then it does nothing for the pair (v, E). If there are no $v \in G$ of depth s, then the process ends.

Each member of $\mathcal{T}_A$ can be produced by the Galton–Watson process but not all trees produced by the Galton–Watson process are members of $\mathcal{T}_A$. For example, the Galton–Watson process can produce trees that violate part (1) of Proposition 1.3.3.

Claim 1.3.10. Let T be a Moser tree with root vertex labeled A. The probability p_T that the above Galton–Watson process ends and produces exactly T is given by

$$p_T = \frac{1 - z(A)}{z(A)} \prod_{v \in T} \left(z([v]) \prod_{B \in \Gamma[v]} (1 - z(B)) \right).$$

Before proving the claim, note that this implies that

$$p_T \geq \frac{1 - z(A)}{z(A)} \prod_{v \in T} \Pr([v])$$

by the local lemma condition. Then, by Proposition 1.3.9,

$$\Pr(\text{the resample algorithm produces } T) \leq \prod_{v \in T} \Pr([v]) \leq \frac{z(A)}{1 - z(A)} p_T. \tag{1.3.11}$$

Summing over all trees with root node labeled A yields

$$\sum_{T \in \mathcal{T}_A} \Pr(\text{the resample algorithm produces } T) \leq \frac{z(A)}{1 - z(A)} \sum_{T \in \mathcal{T}_A} p_T.$$

The Galton–Watson process can only produce at most one tree (and not always a

Moser tree), so $\sum_{T \in \mathcal{T}_A} p_T \leq 1$. Thus,

$$\sum_{T \in \mathcal{T}_A} \Pr(\text{the resample algorithm produces } T) \leq \frac{z(A)}{1 - z(A)}.$$

Summing over all $A \in \mathcal{A}$ yields

$$\sum_{T} \Pr(\text{the resample algorithm produces } T) \leq \sum_{A \in \mathcal{A}} \frac{z(A)}{1 - z(A)}.$$

Let γ be the number of steps in the resample algorithm. Since each step of the resample algorithm produces exactly one tree,

$$\begin{aligned}
\mathbb{E}(\gamma) &= \mathbb{E}(\#\text{ of Moser trees produced by the resample algorithm}) \\
&= \sum_{T} \mathbb{E}\left(1_{\{\text{the resample algorithm produces } T\}}\right) \\
&= \sum_{T} \Pr(\text{the resample algorithm produces } T) \\
&\leq \sum_{A \in \mathcal{A}} \frac{z(A)}{1 - z(A)} \qquad (1.3.12) \\
&< \infty,
\end{aligned}$$

giving the statement of Theorem 1.3.1. It remains to show Claim 1.3.10

Proof of Claim 1.3.10. For each vertex $v \in T$, let $W_v \subset \Gamma^+([v])$ be the set of inclusive neighbors of $[v]$ that do not occur as a label of some child of v. p_T has a factor of

$$z([v]) \prod_{B \in W_v} (1 - z(B))$$

for each $v \in T$ except for the root r, which is guaranteed to appear and hence

does not contribute the leading factor of $z([r])$. Therefore, the probability that the Galton–Watson process produces exactly T is given by

$$\begin{aligned} p_T &= \left[\prod_{B\in W_r}(1-z(B))\right]\left[\prod_{v\in T, v\neq r}\left(z([v])\prod_{B\in W_v}(1-z(B))\right)\right] \\ &= \frac{1}{z(A)}\prod_{v\in T}\left(z([v])\prod_{B\in W_v}(1-z(B))\right). \end{aligned}$$

Multiplying and dividing by $\prod_{v\in T, v\neq r}(1-z([v]))$ yields

$$\begin{aligned} p_T &= \frac{1-z(A)}{z(A)}\prod_{v\in T}\left(\frac{z([v])}{1-z([v])}\prod_{B\in W_v}(1-z(B))\prod_{u \text{ child of } v}(1-z([u]))\right) \\ &= \frac{1-z(A)}{z(A)}\prod_{v\in T}\left(\frac{z([v])}{1-z([v])}\prod_{E\in\Gamma^+([v])}(1-z(B))\right). \end{aligned}$$

Absorbing $\frac{1}{1-z([v])}$ into the innermost product allows us to replace $\Gamma^+([v])$ with $\Gamma([v])$, so

$$p_T = \frac{1-z(A)}{z(A)}\prod_{v\in T}\left(z([v])\prod_{B\in\Gamma([v])}(1-z(B))\right),$$

as claimed. □

This completes the proof of Theorem 1.3.1.

1.4 The Infinite Case

So far, we have only considered finite applications of the Local Lemma. Before bringing the Local Lemma into the realm of computability, we discuss its infinitary state-

ment. We work over the same variable context as in Theorem 1.3.1 with the modifications that $\mathcal{X} = \{x_i : i \in \omega\}$ is an infinite set of independent random variables with finite range and $\mathcal{A}$ is an infinite set of events with each $A \in \mathcal{A}$ completely determined by a finite set $\mathrm{VBL}(A)$. As before, let $\Gamma(A) = \{B : \mathrm{VBL}(A) \cap \mathrm{VBL}(B) \neq \emptyset \text{ and } A \neq B\}$.

Theorem 1.4.1 (Lovász Local Lemma, Infinite Version)**.** *Let $\mathcal{X}$ and $\mathcal{A}$ be as above. If there exists a real-valued function $z : \mathcal{A} \to (0, 1)$ such that, for each $A \in \mathcal{A}$,*

1. *A is mutually independent from $\mathcal{A} \setminus \Gamma^+(A)$ and*

2. $$\Pr(A) \leq z(A) \prod_{B \in \Gamma(A)} (1 - z(B)),$$

then there exists an assignment of the variables in $\mathcal{X}$ that avoids all the events $A \in \mathcal{A}$.

Proof. This follows from the same argument as the infinite versions of Theorems 1.2.1 and 1.2.2, although we must make use of assumptions that were previously implicit. For each $s \in \omega$, let $\mathcal{A}_s$ be the set of $A \in \mathcal{A}$ such that $\mathrm{VBL}(A) \subset \{x_1, x_2, ..., x_s\}$. The range of each x_i is finite, so there are only finitely many unique events in $\mathcal{A}_s$. Thus, by the finite version of the Local Lemma, there is an assignment of the variables $x_1, x_2, ..., x_s$ avoiding all $A \in \mathcal{A}_s$. Furthermore, if some assignment σ of the variables makes all $A \in \mathcal{A}_s$ false, then $\sigma|_n$ makes all $A \in \mathcal{A}_n$ false. Thus, the set of assignments of finite initial segments $x_1, x_2, ..., x_s$ of $\mathcal{X}$ that avoid all $A \in \mathcal{A}_s$ forms an infinite tree T. T is finitely branching because each x_i has finite range. Therefore, T has a path X by Königs lemma. To see that X makes each $A \in \mathcal{A}$ false, note that because $\mathrm{VBL}(A)$ is finite there is some stage s such that $A \in \mathcal{A}_s$, so $X|_s$ makes A false. Therefore, X also makes A false. □

The infinite resample algorithm has the same definition as in the finite case. Rumyantsev and Shen's analysis of the infinite resample algorithm yields a computable version of the Local Lemma. The argument requires in an essential way that for each $x \in \mathcal{X}$, the set $\{A \in \mathcal{A} : x \in \mathrm{VBL}(A)\}$ is finite. We also need the following computability assumptions:

1. The events $\mathcal{A} = A_1, A_2, A_3, ...$ are computably presented. That is, $\mathrm{VBL}(A_i)$ and the finite set of assignments of the variables in $\mathrm{VBL}(A_i)$ that make A_i true are both uniformly computable with respect to i.

2. Each x_i has a rational-valued probability distribution that is uniformly computable with respect to i.

3. The code for the finite set of indices $\{j : x_i \in \mathrm{VBL}(A_j)\}$ is uniformly computable given i.

Theorem 1.4.2 (Lovász Local Lemma, Computable Version [27])**.** *Let $\mathcal{X}$ and $\mathcal{A}$ be as above. If there exists a rational constant $\alpha \in (0,1)$ and a computable rational-valued function $z : \mathcal{A} \to (0,1)$ such that, for each $A \in \mathcal{A}$,*

1. *A is mutually independent from $\mathcal{A} \setminus \Gamma^+(A)$, and*

2. $$\Pr(A) \leq \alpha z(A) \prod_{B \in \Gamma(A)} (1 - z(B)),$$

*then there exists a **computable** assignment of the variables in $\mathcal{X}$ that avoids all the events $A \in \mathcal{A}$.*

Note the extra factor of α on the right hand side of the inequality. It is used

to control the probability of long chains of resampling late events that cause earlier events to go from false to true.

1.5 Probabilistic Turing Machines

To model the resample algorithm, Rumyantsev and Shen use a custom model of computation that allows the output tape to be *rewritable* — that is, the machine can change the contents of each output cell arbitrarily often. We think of these machines as random variables from Cantor space equipped with the Cantor measure to partial functions from ω to $\{0,1\}$. In order to simulate this using standard models of computation, we treat inputs to a Turing functional as ordered pairs (i,s), where $\Phi(i,s)$ represents the contents of the i'th cell at stage s.

Definition 1.5.1 (Rewritable Probabilistic Turing Machine)**.** A **rewritable probabilistic Turing machine** is a random variable $\mathbf{\Phi} : 2^\omega \to \{$partial functions from ω to $\{0,1\}\}$ equipped with a total computable Turing functional Φ with the property that

$$\mathbf{\Phi}(B)(i) = \begin{cases} \lim_{s\to\infty} \Phi^B(i,s) & \text{if the limit exists} \\ \uparrow & \text{otherwise} \end{cases}$$

for all i.

We will be interested in the output distribution of rewritable probabilistic Turing machines $\mathbf{\Phi}$. Suppose that $\Pr(\mathbf{\Phi}(i)\downarrow \text{ for all } i) = 1$. Then, the probability distribution Q on the output of $\mathbf{\Phi}$ is determined by its values $Q(\Sigma_x) = \mu(\{B : \mathbf{\Phi}(B) \in \Sigma_x\})$ where Σ_x is the cone of infinite extensions of the binary string x. We say that Q is computable if $Q(\Sigma_x)$ is uniformly computable with respect to x. The following

proposition states that we can compute Q if there is a computable function $N(i,\delta)$ such that the probability that the i'th entry changes after step $s = N(i,\delta)$ is less than δ.

Proposition 1.5.2. *Let $\mathbf{\Phi}$ be a rewritable probabilistic Turing machine with Turing functional Φ. Suppose that there is a computable function $N(i,\delta)$ such that the probability that the i'th entry changes after step $N(i,\delta)$ is less than δ. Then,*

1. *For each i, $\Pr(\mathbf{\Phi}(i)\downarrow) = 1$.*

2. *The output distribution on $\mathbf{\Phi}(i)$ is uniformly computable w.r.t. i.*

We call rewritable probabilistic Turing machines satisfying the conclusions of Proposition 1.5.2 **layerwise computable mappings**.

Proof. It is sufficient to show that, given rational $\delta > 0$ and $x \in 2^{<\omega}$, we can approximate $Q(\Sigma_x)$ with error at most δ. Let $|x| = \text{length}(x)$. Let

$$k = \max(\{N(i,\delta/|x|) : i < |x|\}).$$

Because Φ is a total Turing functional, there is computable function $m : \omega \to \omega$ such that $\Phi^\sigma(i,n)\downarrow$ for every n, $i < n$ and $\sigma \in 2^{m(n)}$. Therefore, for all $n, i \leq n$ and $B \in 2^\omega$, $\Phi^{B|_{m(n)}}(i,n)\downarrow$. For the purpose of properly notating probabilities, let $I : 2^\omega \to 2^\omega$ be the identity random variable; that is, $I(B) = B$. Then, we approximate $Q(\Sigma_x)$ by

$$\begin{aligned}\widehat{Q}(\Sigma_x) &= \frac{\#\{y \in 2^{m(k)} : \Phi^y(i,k)\downarrow = x(i) \text{ for all } i < |x|\}}{2^{m(k)}} \\ &= \Pr(\Phi^{I|_{m(k)}}(i,k) = x(i) \text{ for all } i < |x|).\end{aligned}$$

Then,

$$\begin{aligned}
|Q(\Sigma_x) - \widehat{Q}(\Sigma_x)| &\leq \Pr\left((\exists i < |x|)(\exists s > k)[\Phi^I(i,s) \neq \Phi^I(i,k)]\right) \\
&= \sum_{y \in 2^{m(k)}} \Pr\left((\exists i < |x|)(\exists s > k)[\Phi^I(i,s) \neq \Phi^y(i,k)] | y \prec I\right) \Pr(y \prec I) \\
&\leq \sum_{y \in 2^{m(k)}} \sum_{i < |x|} \Pr\left((\exists s > k)[\Phi^I(i,s) \neq \Phi^y(i,k)] | y \prec I\right) \Pr(y \prec I) \\
&\leq \sum_{i < |x|} \frac{\delta}{|x|} \left(\sum_{y \in 2^{m(k)}} \Pr(y \prec I) \right) \\
&= \delta
\end{aligned}$$

so we have successfully approximated $Q(\Sigma_x)$ with error less than or equal to δ. □

Now that we have a computable output distribution for a layerwise computable mapping, we can find a computable element of its output.

Proposition 1.5.3. *Let $\mathbf{\Phi}$ be a layerwise computable mapping and let $F \subset 2^\omega$ be a closed set such that $\Pr(\Phi \in F) = 1$. Then, F has a computable element.*

Proof. Let $Q(\Sigma_x) = \Pr(x \prec \mathbf{\Phi})$. Part 1 of Proposition 1.5.2 implies that $Q(\Sigma_x) = Q(\Sigma_{x^\frown 0}) + Q(\Sigma_{x^\frown 1})$, so if $Q(\Sigma_x) > 0$ then either $Q(\Sigma_{x^\frown 0}) > 0$ or $Q(\Sigma_{x^\frown 1}) > 0$. Thus, if $Q(\Sigma_x) > 0$, we can computably define a computable $A \in 2^\omega$ such that $x \prec A$ and for each $\tau \prec A$, $Q(\Sigma_\tau > 0)$. If $A \notin F$, then, since F is closed, there is some $\tau \prec A$ such that τ has no extensions in F. But $Q(\Sigma_\tau) > 0$, contradicting that $X \in F$ with probability 1. Thus, $A \in F$. □

We can also define rewritable probabilistic Turing machines which compute from a fixed oracle. This works by modifying Definition 1.5.1 to allow the Turing functional

Φ to have an oracle. Fix rewritable probabilistic Turing machine $\mathbf{\Phi}$. For $C \in 2^\omega$, we write $\mathbf{\Phi}^C$ to denote the layerwise rewritable probabilistic Turing machine

$$\mathbf{\Phi}^C(B)(i) = \begin{cases} \lim_{s\to\infty} \Phi^{B\oplus C}(i,s) & \text{if the limit exists} \\ \uparrow & \text{otherwise} \end{cases}.$$

Note that the output of $\mathbf{\Phi}^C$ might be completely unrelated to the output of $\mathbf{\Phi}$.

Proposition 1.5.4. *Let $\mathbf{\Phi}$ be a rewritable probabilistic Turing machine with Turing functional Φ. Suppose that there is a Turning functional $N(i,\delta)$ such that for each $C \in 2^\omega$ we have that the probability that the i'th entry of $\mathbf{\Phi}^C(B)$ changes after step $N^C(i,\delta)$ is less than δ. Furthermore, suppose that, for each C, there is a closed $F_C \subset 2^\omega$ such that $\mathbf{\Phi}^C(B) \in F_C$ almost surely. Then there is a Turing functional Ψ such that for each C, $\Psi^C \in F_C$.*

Proof. Use N^C and apply Proposition 1.5.2 to compute a Turing functional Θ such that $\Theta^C(x,\cdot) = Q_C(\Sigma_x)$ as reals, where $Q_C(\Sigma_x)$ is $Q(\Sigma_x)$ for $\mathbf{\Phi}^C$. Then, compute Ψ^C by applying the construction in the proof of Proposition 1.5.3 using Θ^C. □

However, there is no guarantee that $\bigcap_C F_C \neq \emptyset$. In the case of non-repetitive binary sequence games, Proposition 1.8.8 guarantees that $\bigcap_C F_C = \emptyset$ when C ranges over player 2 strategies.

1.6 Computable Local Lemma

It remains to show that the resample algorithm produces a layerwise computable mapping as in Theorem 1.5.2. We can use elements of 2^ω as a random source by

coding the finite range of each $x \in \mathcal{X}$ and using a pairing function to determine values for each x_i^j. Let $\Phi^B(i,s)$ be the value of x_i at stage s of the resample algorithm with random source B. Let $\mathbf{\Phi}(B)(i) = \lim_{s\to\infty} \Phi^B(i,s)$. We need to show that the rewritable probabilistic Turing machine $\mathbf{\Phi}$ associated with Φ is a layerwise computable mapping. We also need to show that none of the bad events $A \in \mathcal{A}$ are true under the valuation of the $x \in \mathcal{X}$ given by $\mathbf{\Phi}(B)$ for almost every $B \in 2^\omega$. Then, we can apply Proposition 1.5.3 to the set F of valuations of the $x \in \mathcal{X}$ that make each $A \in \mathcal{A}$ false to show that F has a computable element.

To show that $\mathbf{\Phi}$ is layerwise computable, we need to compute $N(i,\delta)$ such that x_i changes with probability less than δ after stage $N(i,\delta)$. Because each x_i is involved in finitely many events and the set of events is uniformly computable, it is sufficient to find an $M(i,\delta)$ such that the probability that A_i is resampled after stage $M(i,\delta)$ is less than δ. Fix i and δ. Let k be such that $A_1, A_2, ..., A_k$ contains all events at distance m or less from A_i in the dependency graph for m such that

$$\alpha^m \frac{z(A_i)}{1 - z(A_i)} \leq \delta/2.$$

Let T_k be the random variable whose value is the first stage such that each of $A_1, A_2, ..., A_k$ are false. The event "A_i is resampled after time t" is covered by the events $t < T_k$ and $t \geq T_k$. We show that the probability of both of these events effectively converges to 0.

Because the resample algorithm resamples events with lower indices first, no A_j with $j > k$ can be resampled before T_k. Thus, we can use the bound in Line 1.3.12

to get

$$\mathbb{E}(T_k) < \sum_{i<k} \frac{z(A_i)}{1 - z(A_i)}.$$

By Markov's inequality,

$$\Pr(T_k \geq t) \leq \frac{\mathbb{E}(T_k)}{t},$$

which we can solve to find s large enough so that

$$\Pr(T_k \geq s) < \delta/2. \tag{1.6.1}$$

Finding such s is uniformly computable from i and δ. We set $M(i, \delta) = s$.

Since $\Pr(T_k \geq s) < \delta/2$, we also have that

$$\Pr(A_i \text{ is resampled after stage } s \text{ and } T_k \geq s) < \delta/2.$$

Now we bound the probability that A_i is resampled at some stage $t_1 > s$ and $T_k < s$. Consider the log of resampling $E_1, E_2, \ldots$. Suppose that A_i is resampled at some stage $t_1 > s > T_k$. We claim that there must be a sequence of stages $T_k < t_m < t_{m-1} \cdots < t_3 < t_2 < s < t_1$ such that for each $1 < j \leq m$, we have that $E_{t_{j-1}}$ is false before resampling E_{t_j} at stage t_j and true after resampling E_{t_j} at stage t_j.

We prove the claim by induction on the final index $n \leq m$ of the sequence. For $n = 1$, the sequence t_1 satisfies the requirements. Suppose there is a such a sequence of stages $T_k < t_{n-1} < t_{n-2} < \cdots < t_1$. For resampling $E_{t_{j+1}}$ to change the truth value of E_{t_j}, it must be the case that $\mathrm{Var}(E_{t_{j+1}}) \cap \mathrm{Var}(E_{t_j}) \neq \emptyset$ and hence $E_{t_{j+1}} \in \Gamma^+(E_{t_j})$.

It follows that $E_{t_{n-1}}$ is of distance at most $n \leq m$ from $E_{t_1} = A_i$. Because all events of distance $d \leq m$ from A_i are false at T_k, we have that $E_{t_{n-1}}$ is false at stage T_k. So, there must be some t' such that $T_k < t' < t_{n-1}$ such that $E_{t_{n-1}}$ goes from being false before resampling $E_{t'}$ at stage t' to being true after the resampling $E_{t'}$ at stage t'. This exactly the condition required for t_n, so we let $t_n = t'$. This completes the proof of the claim.

Recall that the LLL condition for the computable version (Theorem 1.4.2 requires an additional factor of $\alpha \in (0, 1)$. Then, Inequality 1.3.11 becomes

$$\Pr(\text{the resample algorithm produces } T) \leq \alpha^{|T|} \frac{z(A_i)}{1 - z(A_i)} p_T.$$

Consider the Moser tree S associated with $E_1, \ldots, E_{t_1} = A_i$. The subsequence $E_{t_m}, E_{t_{m-1}}, \ldots, E_{t_1}$ is a chain of neighbors, so S has at least m vertices and root node labeled A_i. Let B be the event that the resample algorithm produces a tree of size m with root node labeled A_i. It follows that

$$\begin{aligned}
\Pr(A_i \text{ is resampled at stage } t_0 > s \text{ and } T_k < t_1) &\leq \Pr(B) \\
&\leq \sum_{T \in \mathcal{T}_{A_i}, |T| \geq m} \alpha^{|T|} \frac{z(A_i)}{1 - z(A_i)} p_T \\
&\leq \alpha^m \frac{z(A_i)}{1 - z(A_i)} \sum_{T \in \mathcal{T}_{A_i}} p_T \\
&\leq \alpha^m \frac{z(A_i)}{1 - z(A_i)} \\
&\leq \delta/2.
\end{aligned}$$

Thus, we have that $\Pr(A_i \text{ is resampled after stage } s) \leq \delta$. Therefore, the resample

algorithm is a layer-wise computable mapping. To apply Proposition 1.5.3, we just need to check that the output sequence

$$\lim_{t\to\infty} x_1^{V_t(1)} \lim_{t\to\infty} x_2^{V_t(2)} \lim_{t\to\infty} x_3^{V_t(3)} \dots$$

makes each $A \in \mathcal{A}$ false almost surely, and that making each $A \in \mathcal{A}$ false is closed. To see that latter, note that each A being true is open, so the intersection of them all being false is closed. To see the former, fix $A_i \in \mathcal{A}$ and suppose that A is true in the output sequence. Then, the resample algorithm resamples some A_j with $j \leq i$ infinitely many times. However, this happens with probability 0 as we have just shown that as t goes to infinity, the probability that A_j gets resampled after time t goes to 0. Since $\mathcal{A}$ is countable, the union probability that *any* $A \in \mathcal{A}$ is true in the output sequence is also 0.

Thus, the resample algorithm is a layerwise computable mapping which almost surely converges in the closed set F of all valuations of the $x_i \in \mathcal{X}$ that make each $A \in \mathcal{A}$ false. By Proposition 1.5.3, F has a computable element. This completes the proof of Theorem 1.4.2.

1.7 The Lefthanded Local Lemma

We would like to prove computable versions of Theorems 1.2.1 and 1.2.2. However, although we have a computable version of the local lemma, we cannot apply it to these problems due to the additional constraint that for each x_i, the set $\{A : x_i \in \mathrm{VBL}(A)\}$ is finite. To see this, for Theorem 1.2.1, fix an ε and N_ε satisfying the conditions. For

fixed x_i, consider

$$S = \{A_{k,l,n} : x_i \in \mathrm{VBL}(A_{k,l,n}) \wedge n > N_\varepsilon \wedge l - k < (2-\varepsilon)^n\}.$$

Fix $A_{k_0,l_0,n_0} \in S$ such that $x_i \in [k_0, k_0 + n_0 - 1]$. Then, for each $n > n_0$, $A_{k_0,l_0,n} \in S$ as well, so S is infinite. A first step to solving this problem is restricting $\mathcal{A}$ to the $A_{k,l,n}$ such that $(l-k) = \lfloor (2-\varepsilon)^n \rfloor$.

This is still a sufficient set of events because if $A_{k,l,n}$ is false, then the interval $[k, k+n)$ does not have the same entries as $[l, l+n)$. This implies that for any $m > n$, $[k, k+m)$ also does not have the same entries as $[l, l+m)$. Therefore, if $A_{k,l,n}$ is false then $A_{k,l,m}$ is false for each $m > n$.

This seems helpful because for fixed l and k, there is only one n to worry about. However, we still find infinitely many $A_{k_0,l,n} \in \mathcal{A}$ by increasing both n and l simultaneously.

A similar problem occurs for the $A_{k,n}$ from the proof of Theorem 1.2.2. If $x_i \in [k_0, k_0 + n_0 - 1]$ then $x_i \in [k_0, k_0 + n - 1]$ for each $n > n_0$.

While there may be some clever way of further reducing the event space, we will solve this issue by resampling fewer variables for each event. In the proof of Theorem 1.4.2, the assumption that each x_i is part of finitely many events is required so that each x_i eventually stops getting resampled. In Theorems 1.2.2 and 1.2.1, note that resampling just the latter half of the variables of $A_{k,l,n}$ is enough to "reset" the probability of $A_{k,l,n}$, and likewise for $A_{k,n}$. We can take advantage of this during the resample algorithm.

In the case of Theorem 1.2.1, when $A_{k,l,n}$ needs to be resampled, instead of resampling all of $\mathrm{VBL}(A_{k,l,n})$ we only resample a subset $\mathrm{RSP}(A_{k,l,n}) := [l, l+n)$. We call

$\mathrm{RSP}(A_{k,l,n})$ the *resample variables* of $A_{k,l,n}$. The remaining variables, $\mathrm{STC}(A_{k,l,n}) := \mathrm{VBL}(A_{k,l,n}) \setminus \mathrm{RSP}(A_{k,l,n})$ remain the same. We call $\mathrm{STC}(A_{k,l,n})$ the *static variables* of $A_{k,l,n}$. Note the probability of $A_{k,l,n}$ after resampling $\mathrm{VBL}(A_{k,l,n})$ is equal to the probability of $A_{k,l,n}$ after resampling only $\mathrm{RSP}(A_{k,l,n})$. Given any valuation μ of the variables in $\mathrm{STC}(A_{k,l,n})$, let E_μ be the event that $x = \mu(x)$ for each $x \in \mathrm{STC}(A_{k,l,n})$. We have

$$\Pr(A_{k,l,n} \text{ is true after resampling the variables in } \mathrm{RSP}(A_{k,l,n}) | E_\mu) = 2^{-n},$$

which is equal to $\Pr(A_{k,l,n})$.

However, reducing the number of variables we resample makes it more difficult to couple the Moser–Tardos algorithm with a T-check. To ensure that the T-check and the Moser–Tardos algorithm resample events in the same order, we will need to assign priorities to the events in $\mathcal{A}$. In the case of Theorem 1.2.1, we will prioritize events by the right endpoints of their resample sets.

1.7.1 The Computable Lefthanded Local Lemma

Let $\mathcal{X} = \{x_1, x_2, x_3, ...\}$ be a set of independent random variables and let $\mathcal{A}$ be a countable set of events, each $A \in \mathcal{A}$ with a finite set $\mathrm{VBL}(A) \subset \mathcal{X}$ such that A completely depends on the variables in $\mathrm{VBL}(A)$. For each $A \in \mathcal{A}$ fix a subset $\mathrm{RSP}(A) \subset \mathrm{VBL}(A)$ that we will resample when A is true during the resample algorithm. Designate $\mathrm{STC}(A) := \mathrm{VBL}(A) \setminus \mathrm{RSP}(A)$ as the (finite) list of variables possibly dependent with A that we do not resample. We also require that $\mathrm{RSP}(A)$ is an interval $[a, b) = \{x_a, x_{a+1}, \ldots, x_{b-1}\}$ and that $\max(\mathrm{STC}(A)) < \min(\mathrm{RSP}(A))$ for

all $A \in \mathcal{A}$. We define the neighborhood relation $\Gamma(A)$ by $B \in \Gamma(A)$ if and only if $\mathrm{RSP}(A) \cap \mathrm{RSP}(B) \neq \emptyset$ and $A \neq B$. Note that it is no longer the case that $B \notin \Gamma(A)$ implies that A and B are independent. As before, let $\Gamma^+(A) = \Gamma(A) \cup \{A\}$.

Fix a total linear order $\succ$ of order type ω on $\mathcal{A}$ such that

$$[\max(\mathrm{RSP}(A)) > \max(\mathrm{RSP}(B))] \implies [A \succ B].$$

Note that such an order always exists because we will assume that for each $x \in \mathcal{X}$, the set of events $\{A \in \mathcal{A} : x \in \mathrm{RSP}(A)\}$ which can resample x is finite.

Definition 1.7.1. If $A \prec B$ and $A \notin \Gamma(B)$ then we write $A \ll B$.

The relation $A \ll B$ indicates that $A \prec B$ and that A and B have some amount of separation between their resample sets. We now develop some basic properties of $\prec$ and $\ll$.

It follows from definition chasing that for all events $A, B \in \mathcal{A}$, we have that $A \ll B$ if and only if $\mathrm{RSP}(A)$ is completely to the left of $\mathrm{RSP}(B)$. The latter half of this biconditional implies that $\mathrm{RSP}(B) \cap \mathrm{VBL}(A) = \emptyset$. This gives us that $A \ll B$ implies that resampling B has no effect on A. It also follows from the biconditional that $\ll$ is transitive.

Definition 1.7.2. The lefthanded resample algorithm follows the same steps as the resample algorithm, modified in two ways.

- The resample algorithm chooses the $\succ$-least true event to resample instead of going by an arbitrary indexing.
- When the resample algorithm resamples $A \in \mathcal{A}$, it resamples the variables in $\mathrm{RSP}\, A \subset \mathrm{VBL}\, A$ instead of all of $\mathrm{VBL}\, A$.

In our analysis of when the lefthanded resample algorithm converges, we need to keep track of the probability that each $A \in \mathcal{A}$ becomes false after resampling the variables $\mathrm{RSP}(A)$. However, this depends on the current valuation of $\mathrm{STC}(A)$. For μ an evaluation of some set of variables, let E_μ be the event that values of the variables match their valuation by μ. For $A \in \mathcal{A}$, let $\mathcal{E}_A$ be the set of valuations of the variables in $\mathrm{STC}(A)$.

The following theorem gives conditions for which this modified resample algorithm converges.

Theorem 1.7.3. *Let $\mathcal{A}$, $\mathcal{X}$, and Γ all be as above. Suppose there is $z : \mathcal{A} \to (0,1)$ and $P^* : \mathcal{A} \to (0,1)$ such that for each $A \in \mathcal{A}$,*

$$P^*(A) \geq \sup_{\mu \in \mathcal{E}_A} (\Pr(A|E_\mu))$$

and

$$P^*(A) \leq z(A) \prod_{B \in \Gamma(A)} (1 - z(B)).$$

Then, for any finite subset of events $\mathcal{B} \subset \mathcal{A}$, $t_{\mathcal{B}}$ = "the first stage at which all $B \in \mathcal{B}$ are false" has finite expectation when the lefthanded resample algorithm is run.

The rest of this section constitutes the proof of Theorem 1.7.3, broken up into subsections.

1.7.1.1 Logs and Moser Trees

We will need to precisely define the probability spaces we are working with. Let Ω be the probability space of valuations of $\mathcal{X}$. Let $\mathcal{X}' = \{x_i^j\}|_{i,j \in \mathbb{N}}$ such that each x_i^j

has the same probability distribution as x_i. Let Ω' be the probability space of all valuations the variables in $\mathcal{X}'$.

It will be helpful to temporarily decouple possible logs of the resample algorithm from the resample algorithm itself. We call a sequence of events $E_1, E_2, ..., E_n$ a *legal log* if for each $1 \leq i < n$, $E_i \not\gg E_{i+1}$.

Claim. Every log produced by a run of the resample algorithm is a legal log.

Proof. Let $E_1, \ldots, E_n$ be an initial segment of the log of the resample algorithm. Suppose that $E_i \gg E_{i+1}$. Then, E_{i+1} must be false at stage i, as otherwise, the resample algorithm would have picked E_{i+1} to resample at stage i. Therefore, E_{i+1} went from false to true after resampling E_i. But $E_i \gg E_{i+1}$ implies that $\mathrm{RSP}(E_i) \cap \mathrm{VBL}(E_{i+1}) = \emptyset$, so resampling E_i could not have changed the truth value of E_{i+1}. □

Furthermore, any sub-interval of a legal log is legal. We will use the following fact.

Lemma 1.7.4. *Let $E_1, E_2, ..., E_n$ be a legal log. Let $1 \leq i < j \leq n$ and suppose that $E_i \succ E_j$. Then, there is some k with $i \leq k < j$ such that $E_k \in \Gamma^+(E_j)$.*

Proof. If $E_i \in \Gamma^+(E_j)$ then we are done. Otherwise, $E_i \gg E_j$. Since $E_1, E_2, ..., E_n$ is a legal log, there is some k (in particular, $k = j-1$) with $i < k < j$ and that $E_k \not\gg E_j$. If $E_k \in \Gamma^+(E_j)$ then we are done. Otherwise, $E_k \ll E_j$. Then, we can fix the least s such that $i < s < j$ such that $E_s \ll E_j$. Thus, $E_{s-1} \not\ll E_j$. If $E_{s-1} \in \Gamma^+(E_j)$, then we are done. Otherwise, $E_{s-1} \gg E_j$. Since $E_j \gg E_s$ and $\gg$ is transitive, we have that $E_{s-1} \gg E_s$. This contradicts that $E_1, ..., E_n$ is a legal log. □

Similarly to the proof of the original algorithmic local lemma, we construct a Moser tree based on the log. To construct the Moser tree associated with the legal

log $E_1, E_2, ..., E_n$, we use the same process as before except that we use the new definition of $\Gamma(A)$.

Fix a Moser tree T and a legal log $E_1, E_2, ..., E_n$ that produces T. For $v \in T$, let $q(v)$ be the stage at which v is added to T, so that $[v] = E_{q(v)}$. Proposition 1.3.3 holds with the same proof.

We will see that requiring the resample algorithm to prioritize events by their $\prec$-order fixes the order of vertices added during the construction of a given Moser tree: for any Moser tree T, the most recently added vertex is the one whose label is $\prec$-least among all vertices of maximal depth among its neighbors. To be precise, define $M(T) = \{v : d(v) = \max(\{d(w) : [w] \in \Gamma^+([v])\})\}$. Since the labels of children of v are neighbors of $[v]$, we have that members of $M(T)$ must be leaves of $M(T)$, as otherwise they would have a child and hence a neighbor of greater depth. Furthermore, let $[M(T)] = \{[v] : v \in M(T)\}$. Then, let $G(T) = v$ such that $[v]$ is the $\prec$-least element of $[M(T)]$. This is well defined because if two vertices have the same label then they have different depths, so only one of them can be in $M(T)$. List the vertices of T as $v_1, v_2, ..., v_{|T|}$ by $v_1 = G(T)$ and $v_{i+1} = G(T \setminus \{v_1, ..., v_i\})$.

First we confirm that $G(T)$ indeed extracts the most recently added vertex of T.

Lemma 1.7.5. *Let $E_1, \dots E_n$ be a legal log producing a Moser tree T enumerated as above by $T = \{v_1, \dots, v_{|T|}\}$ with $v_1 = G(T)$ and $v_{i+1} = G(T \setminus \{v_1, ..., v_i\})$. Then,*

$$q(v_1) = \min\{q(w) : w \in T\}.$$

Proof. Let $w \in T$ be such that $q(w) = \min(\{q(v) : v \in T\}$. First we show that $w \in M(T)$. Then we show that $[w]$ is the $\prec$-least member of $[M(T)]$ and therefore that $w = G(T) = v_1$.

To see that $w \in M(T)$, note that w is the last vertex added to T and therefore has maximal depth among $\{v : [v] \in \Gamma^+(w)\}$.

To see that $[w]$ is the $\prec$-least member of $[M(T)]$, let v be such that $[v]$ is the $\prec$-least member of $[M(T)]$ and suppose that $w \neq v$. Then, $[v] \prec [w]$. Since $w, v \in M(T)$, $[w] \notin \Gamma^+([v])$. Therefore, $[w] \gg [v]$. By Lemma 1.7.4, there is some k with $q(w) < k < q(v)$ such that $E_k \in \Gamma^+([v])$. Then, at stage k of constructing the Moser tree, a vertex v' with label E_k would be added to T. Since $k < q(v)$, we have that v would already have been added to the tree. However, since $E_k \in \Gamma^+([v])$, we have that $d(v') > d(v)$, contradicting that $v \in M(T)$. □

Next, we show that removing $G(T)$ corresponds to removing an initial segment of a legal log producing T.

Lemma 1.7.6. *Let $E_1, \ldots E_n$ be a legal log producing a Moser tree T and let $v = G(T)$. Then, the legal log $E_{q(v)+1}, \ldots E_n$ produces the Moser tree $T \setminus \{v\}$.*

Proof. Let R be the Moser tree produced at stage $q(v) + 1$ by the Moser tree construction.

By Lemma 1.7.5, we have that for each $w \in T$, if $w \neq v$ then $q(w) > q(v)$. Hence, for each $w \in T$ such that $w \neq v$, $E_{q(v)+1}, \ldots E_n$ contains each $E_{q(w)}$. Therefore, for each $w \in T$ with $w \neq v$, we have that $w \in R$. Hence, $R = T \setminus \{v\}$.

Because the events are exactly the same and in the same order, the Moser tree produced by $E_{q(v)+1}, \ldots E_n$ is also $R = T \setminus \{v\}$. □

We are now ready to give a categorization of all legal logs producing T.

Lemma 1.7.7. *Let $E_1, \ldots E_n$ be a legal log producing a Moser tree T enumerated as above by $T = \{v_1, \ldots, v_{|T|}\}$ with $v_1 = G(T)$ and $v_{i+1} = G(T \setminus \{v_1, ..., v_i\})$. Then,*

$q(v_1) < q(v_2) < q(v_3) < ... < q(v_{|T|}) = n$. *Furthermore, if* $q(v_i) < k < q(v_{i+1})$ *then* $E_k \ll [v_j]$ *for all* $j > i$. *Additionally, if* $k < q(v_1)$ *then* $E_k \ll [v_j]$ *for all* j.

Proof. First we show that $q(v_1) < q(v_2) < q(v_3) < ... < q(v_{|T|})$ by induction.

Base case: The fact that $q(v_1) = \min(q(v) : v \in T)$ is exactly Lemma 1.7.5.

Induction step: Suppose that $q(v_1) < q(v_2) < ... < q(v_k)$. By iterated applications of Lemma 1.7.6, the legal log $E_{q(v_k)+1}, E_{q(v_k)+2}, ..., E_n$ produces $T \setminus \{v_1, v_2, ..., v_k\}$. By Lemma 1.7.5, the least value of $q(v)$ for $v \in T \setminus \{v_1, \ldots, v_k\}$ in the new legal log is obtained by $v = G(T \setminus \{v_1, v_2, ..., v_k\}) = v_{k+1}$. This completes the induction.

It remains to show that if $q(v_i) < k < q(v_{i+1})$ then $E_k \ll [v_j]$ for all $j > i$. For contradiction, suppose that some there are some k and i such that $q(v_i) < k < q(v_{i+1})$ and $E_k \not\ll [v_j]$ for some $j > i$. There are two cases: either $E_k \in \Gamma^+([v_j])$ or $E_k \gg [v_i]$. In both cases, there is some ℓ such $q(v_i) < \ell < q(v_{i+1})$ and $E_\ell \in \Gamma^+([v_j])$. In the first case, we can set $\ell = k$. In the second case, the existence of such an ℓ is given by Lemma 1.7.4. The existence of this ℓ is a contradiction: because E_ℓ is a neighbor of $[v_j]$, we would add another vertex to T at stage ℓ. However, $\ell \neq q(v)$ for any $v \in T$.

The same proof works in the case that $k < q(v_1)$. □

Let r be the root node of T. Since $q(r)$ is maximal, we have that $v_{|T|} = r$ as a consequence of Lemma 1.7.7. We abuse notation by also writing $r = |T|$, so that $r = v_{|T|} = v_r$, with context disambiguating the two notations.

1.7.1.2 The T-check

Fix a Moser tree T with root node r. We will use a different T-check than in the original algorithmic local lemma. Our new T-check does exactly what the resample algorithm would do until each $A \ll [v]$ for all $v \in T$ is false. Then, the T-check

resamples $[G(T)]$, regardless of any other truth values. Afterwards, it runs the $T \setminus \{G(T)\}$-check, starting with the assignment of the variables that the T-check passed on to it. If $T = \emptyset$ then the T-check acts exactly as the resample algorithm. For each $v_i \in T$, let $\hat{q}(v_i)$ be the stage at which the T-check starts doing the $T \setminus \{v_1, v_2, ..., v_i\}$-check. If this never occurs, then $\hat{q}(v_i) = \infty$. We say that the T-check passes if each $\hat{q}(v_i) < \infty$ and $[v_i]$ is true at stage $\hat{q}(v_i)$ before it is resampled. Since r is always of minimal depth among it's neighbors,

The reader may find the following equivalent description of the T-check useful. The T-check runs the resample algorithm until the resample algorithm wants to resample an event that occurs as a label in T. When that happens, the resample algorithm instead resamples the event $[G(T)]$. It then removes $G(T)$ from T and continues the resample algorithm. Continuing in this way, the T-check resamples the events that occur as labels in T in the order $[v_1], [v_2], \ldots, [v_r]$ (although it possibly will never finish doing so).

Claim. Let $S \in \Omega'$. If the resample algorithm run with random source S produces T, then the T-check passes when run on S.

Proof. Suppose that the resample algorithm produces T at some stage n. Then, by Lemma 1.7.7, we have that $q(v_1) < q(v_2) < \cdots < q(v_r)$. The T-check and resample algorithm coincide until the resample algorithm resamples an event that occurs as a label in T. Since the resample algorithm produces T, no labels of nodes in T can be resampled before $q(v_1)$, at which point the resample algorithm resamples $[v_1]$. The T-check would then resample $[G(T)] = [v_1]$, so the T-check and the resample algorithm continue to coincide, so $\hat{q}(v_1) = q(v_1)$. This process continues inductively for all stages. Thus, the log of the resample algorithm and the log of the T-check are

identical.

Since the logs are identical and both use the same random source, the T-check and resample algorithm also have identical valuations of the variables at each stage. In particular, since $\hat{q}(v_i) = q(v_i)$ for each $1 \leq i \leq r$, the T- check always finds $[v_i]$ to be true at stage $\hat{q}(v_i)$ because the resample algorithm does so as well. Therefore, the T-check passes. □

Thus,

$$\Pr(T \text{ is produced by the resample algorithm}) \leq \Pr(\text{The } T\text{-check passes}).$$

It remains to bound the probability that the T-check passes. First, we now develop some basic properties of the T-check.

Recall that $\hat{V}_s(i)$ is equal to the number times x_i has been resampled in the T-check at the beginning of stage s. We will compare various values of the T-check when the T-check is run on two different random sources $S_1, S_2 \in \Omega'$. We will freely think of these values as random variables. For example, $\hat{V}_t(S_1)(i)$ is the value of $\hat{V}_t(i)$ when the T-check is run on random source S_1. We also treat the log of the T-check as a random variable

$$\widehat{\log} : \Omega' \to \mathcal{A}^{\leq \omega}.$$

First we note that a few important sets are open and therefore measurable.

Proposition 1.7.8. *Let T be the Moser tree enumerated as $T = \{v_1, \ldots, v_r\}$. Then,*

(a) For any finite sequence of events $E_1, E_2, \ldots, E_n$, the set

$$\{S \in \Omega' : E_1, \ldots, E_n \text{ is an initial segment of } \widehat{\log}(S)\}$$

is clopen.

(b) For all $1 \leq i \leq r$ *and* $k \in \omega$*, the set*

$$\{S \in \Omega' : \hat{q}(v_i)(S) = k\}$$

is clopen.

(c) For all $1 \leq i \leq r$*,* $k \in \omega$*,* $n \in \mathrm{VBL}([v_i])$ *and* $\ell \in \mathrm{range}(x_n)$*, the set*

$$\{S \in \Omega' : \hat{q}(v_i)(S) = k \text{ and } \hat{V}_{\hat{q}(v_i)}(n) = \ell\}$$

is clopen.

Proof. Membership of the above sets in (a) - (c) are always determined at a finite stage in the T-check. Each stage of the T-check uses only finitely many of the valuations in S. Hence, the sets in (a) - (c) are determined by finitely many elements of $\mathcal{X}'$, so they are clopen. □

Next, we show that, for each j such that $x_j \in \mathrm{RSP}([v_i])$, the value of $\hat{V}_{\hat{q}(v_i)}(j)$ is constant as a random variable of the random source. In other words, whenever the T-check checks whether $[v_i]$ is true at stage $\hat{q}(v_i)$, the variables in $\mathrm{RSP}([v_i])$ have always been resampled the same number of times.

Lemma 1.7.9. *For each* $v_i \in T$ *and each* $x_j \in \mathrm{RSP}([v_i])$*, we have that* $\hat{V}_{\hat{q}(v_i)}(j)$ *is constant as a random variable from the subspace* $\{\hat{q}(r) < \infty\} \subset \Omega'$ *to* $\mathbb{N}$*.*

Proof. Fix v_i, x_j. Let $u(i,j) = \{s < i : x_j \in \mathrm{RSP}([v_s])\}$. We will show that $\hat{V}_{\hat{q}(v_i)}(j) = u(i,j)$ whenever $\hat{q}(r) < \infty$. We proceed by induction on $|u(i,j)|$.

Suppose $|u(i,j)| = 0$. Assume by way of contradiction that $\hat{V}_{\hat{q}(v_i)}(j) > 0$. Then there is some $k < \hat{q}(v_i)$ such that the k'th event E_k of the log has x_j as a resample variable. But then $E_k \in \Gamma([v_i])$, so in particular, $E_k \not\ll [v_i]$. Hence, each $A \prec E_k$ is false at stage k. This also implies that each $A \ll [v]$ for each $[v] \in T$ is false. Therefore, by definition of the T-check, $k = \hat{q}(v_\ell)$ for some $\ell < i$. Then, $\ell \in u(i,j)$, contradicting that $u(i,j) = 0$.

Suppose that the claim is true whenever $|u(i,j)| \leq n$. We show that it is true for $|u(i,j)| = n+1$. Then, $u(i,j) = \{v_{k_1}, v_{k_2}, ..., v_{k_{n+1}}\}$ for some increasing sequence $k_1, k_2, \ldots, k_{n+1}$. Since the sequence is increasing, $|u(i,k_{n+1})| = n$. By the induction hypothesis, $\hat{V}_{\hat{q}(v_{k_{n+1}})}(j) = n$. Hence, $\hat{V}_{\hat{q}(v_i)}(j) \geq n+1$. By a similar argument as in the $u(i,j) = 0$ case, $\hat{V}_{\hat{q}(v_i)}(j) \not> n+1$, so $\hat{V}_{\hat{q}(v_{k_{n+1}})}(j) = n+1$. □

We introduce some new notation. For any measurable function $\tau : \Omega' \to \Omega$, and any event $E \subset \Omega$, define the event $E^\tau := \tau^{-1}(E) = \{S \in \Omega' : \tau(S) \in E\}$. For $A \in \mathcal{A}$, let $\mathrm{RSP}^\tau(A) = \{x_i \circ \tau : x_i \in \mathrm{RSP}(A)\}$ (recall that $x_i : \Omega \to \mathrm{range}(x)$ is a random variable) and similarly for $\mathrm{VBL}^\tau(A)$. For each stage n, fix measurable functions $\tau_n : \Omega' \to \Omega$ such that $(x_i \circ \tau_n) = x_i^{\hat{V}_n(i)}$. In other words, τ_n sends each $S' \in \Omega'$ to the valuation of $\mathcal{X}$ at stage n of the T-check, mapping S' to μ such that $\mu(x_i) = S'(x_i^{\hat{V}_n(i)})$.

Recall that $\{S \in \Omega' : \hat{q}(r) < \infty\}$ is an open set. For each $v \in T$, define $\tau_v : \{S : \Omega' : \hat{q}(r) < \infty\} \to \Omega$ by

$$\tau_v = \tau_{\hat{q}(v)}.$$

The τ_v are measurable maps because the set

$$\tau_v^{-1}([v]) = \{S \in \Omega' : \tau_v(S) \text{ satisfies } [v]\}$$

is open. To see this, note that the condition for the set in is always witnessed at a finite stage of the T-check.

Note that, if $\hat{q}(r) < \infty$, then $\hat{q}(v) < \infty$ for each $v \in T$. Let $\Pr_{\hat{q}(r)<\infty}$ be the probability measure on the subspace $\{\hat{q}(r) < \infty\}$. When working in this probability space, we will need to refer to restrictions $X|_{\{\hat{q}(r)<\infty\}}$, $E_{\{\hat{q}(r)<\infty\}}$, and $\tau|_{\{\hat{q}(r)<\infty\}}$ of random variables X, events E and measurable functions $\tau : \Omega' \to \Omega$. We will, however, not write out the restriction when the context is clear.

By Lemma 1.7.9, we have that, $\hat{V}_{\hat{q}(v)}(j)$ is constant in $\{\hat{q}(r) < \infty\}$ for all $v \in T$ and all $x_j \in \mathrm{RSP}([v])$. Thus, $\mathrm{RSP}^{\tau_{v_i}}([v_i]) = \{x_j^k : x_j \in \mathrm{RSP}([v_i]) \text{ and } k = \hat{V}_{\hat{q}(v_i)}(j)\} \subset \mathcal{X}'$. Therefore, we can set $\mathcal{V}_{v_i} = \mathrm{RSP}^{\tau_{v_i}}([v_i])$. Furthermore, let

$$\begin{aligned}\mathcal{V} &= \bigcup_{1\le i\le r} \mathcal{V}_{v_i} \\ &= \{x_j^k : \text{there is some } v \in T \text{ such that } x_j \in \mathrm{RSP}([v]) \text{ and } k = \hat{V}_{\hat{q}(v)}(j)\}.\end{aligned}$$

The T-check is constructed so that $\mathcal{V}$ has no effect on the log, even if $\hat{q}(r) = \infty$.

Lemma 1.7.10. *Let $S_1, S_2 \in \Omega'$ such that $x_j^k(S_1) = x_j^k(S_2)$ for all $x_j^k \in \mathcal{X}' \setminus \mathcal{V}$. Then,*

$$\widehat{\log}(S_1) = \widehat{\log}(S_2).$$

Proof. Suppose that $\widehat{\log}(S_1) \neq \widehat{\log}(S_2)$. Then, let i be least such that $\widehat{\log}(S_1)(i) \neq \widehat{\log}(S_2)(i)$. Let k be greatest such that $\hat{q}(v_k)(S_1) < i$. Since $\widehat{\log}(S_1)|_{i-1} = \widehat{\log}(S_2)|_{i-1}$, we also have that $\hat{q}(v_k)(S_1) = \hat{q}(v_k)(S_2)$ as well as that k is greatest such that $\hat{q}(v_k)(S_2) < i$. Recall that the T-check resamples $[v_{k+1}]$ at stage i if and only if all $A \in \{A : A \ll [v_j] \text{ for each } j > k\}$ are false at stage i. This is equivalent to $\widehat{\log}(S)(i) = [v_{k+1}]$. To show that this condition being true at stage i is either true for

both S_1 and S_2 or neither S_1 nor S_2, it is sufficient to show that $S_1 \in A^{\tau_i}$ if and only if $S_2 \in A^{\tau_i}$ for each $A \ll [v_j]$ for each $j > k$.

Without loss of generality, suppose $S_1 \in A^{\tau_i}$. We show that $S_2 \in A^{\tau_i}$. The event A^{τ_i} depends only on $\mathrm{VBL}^{\tau_i}(A) = \{x_n^{\hat{V}_i(n)} : x_n \in \mathrm{VBL}(A)\}$, so it is further sufficient to show that $x_n^{\hat{V}_i(n)}(S_1) = x_n^{\hat{V}_i(n)}(S_2)$ for each $x_n \in \mathrm{VBL}(A)$. Since $x_n^m(S_1) = x_n^m(S_2)$ for $x_n^m \notin \mathcal{V}$, it is enough to show that $x_n^{\hat{V}_i(n)} \notin \mathcal{V}$ for every $x_n \in \mathrm{VBL}(A)$ and every random source S. To see this, suppose that $x_n^{\hat{V}_i(n)} \in \mathcal{V}$. Then, $\hat{V}_i(n) = \hat{V}_{\hat{q}(v_j)}(n)$ for some j. We check that the following two cases are impossible:

Case 1: Suppose that $\hat{q}(v_j) < i$. Then, x_n gets resampled at stage $\hat{q}(v_j)$, so $\hat{V}_{\hat{q}(v_j)}(n) < \hat{V}_i(n)$, a contradiction.

Case 2: Suppose that $\hat{q}(v_j) \geq i$. Then, $j > k$ by definition of k. But then, $x_n \in \mathrm{VBL}(A)$ and $x_n \in \mathrm{RSP}([v_j])$, contradicting that $A \ll [v_j]$ for each $j > k$. This completes the claim that $\widehat{\log}(S_1)(i) = [v_{k+1}]$ is equivalent to $\widehat{\log}(S_1)(i) = [v_{k+1}]$.

However, it is not possible for both to be true, since $\widehat{\log}(S_1)(i) \neq \widehat{\log}(S_2)(i)$. Therefore, there are $A_1, A_2 \ll [v_j]$ for all $j > k$ such that $A_1^{\tau_i}(S_1)$ is true and $A_2^{\tau_i}(S_2)$ is true. We have already seen that if $A \ll [v_j]$ for all $j > k$ then $A^{\tau_i}(S_1)$ holds if and only if $A^{\tau_i}(S_2)$ holds. Since $A \prec B \ll C$ implies that $A \ll C$, it is also the case that the $\prec$-least true A at stage i is the same for both S_1 and S_2. Since $A \ll [v_j]$ for each $j > k$, $\widehat{\log}(S_1)(i) = A = \widehat{\log}(S_2)(i)$, contradicting that $\widehat{\log}(S_1)(i) \neq \widehat{\log}(S_2)(i)$. □

1.7.1.3 Bounding the Probability that the T-check Passes

To bound the probability of the T-check passing, we calculate

$$\Pr(\text{The } T\text{-check passes}) = \Pr(\text{The } T\text{-check passes} \wedge (\forall v \in T)(\hat{q}(v) < \infty))$$

$$
\begin{aligned}
&\quad + \Pr(\text{The } T\text{-check passes} \wedge (\exists v \in T)(\hat{q}(v) = \infty)) \\
&= \Pr(\text{The } T\text{-check passes} \wedge \forall (v \in T)(\hat{q}(v) < \infty)) \\
&= \Pr(\text{The } T\text{-check passes} \wedge \hat{q}(r) < \infty) \\
&= \Pr_{\hat{q}(r)<\infty} \left(\bigwedge_{r \geq j \geq 1} [v_j]^{\tau_{v_j}} \right) \Pr(\hat{q}(r) < \infty) \\
&\leq \Pr_{\hat{q}(r)<\infty} \left(\bigwedge_{r \geq j \geq 1} [v_j]^{\tau_{v_j}} \right) \\
&\leq \Pr_{\hat{q}(r)<\infty} \left([v_1]^{\tau_{v_1}} \Big| \bigwedge_{r \geq j \geq 2} [v_j]^{\tau_{v_j}} \right) \Pr_{\hat{q}(r)<\infty} \left(\bigwedge_{r \geq j \geq 2} [v_j]^{\tau_{v_j}} \right) \\
&= \prod_{r \geq i \geq 1} \Pr_{\hat{q}(r)<\infty} \left([v_i]^{\tau_{v_i}} \Big| \bigwedge_{r \geq j > i} [v_j]^{\tau_{v_j}} \right). \qquad (1.7.11)
\end{aligned}
$$

The final piece of the proof of Theorem 1.7.3 is to use the P^* to bound Line 1.7.11 factor-wise. To do so, we pull the defining property of P^* from Ω back to a subspace of Ω'. Before proving this, we need to define some notation for making disjoint covers of Ω and Ω' by valuations of the variables.

Let $\mathcal{M}(A)$ be the set of all valuations of $\mathrm{STC}(A)$. Then, the families $\{E_\mu : \mu \in \mathcal{M}(A)\}$ and $\{\tau^{-1}(E_\mu) : \mu \in \mathcal{M}(A)\}$ form a partition of Ω and Ω', respectively. For each $\mu \in \mathcal{M}(A)$ and $x \in \mathcal{X}$, define $x^\mu : \Omega \to \mathrm{range}(x)$ by

$$
x^\mu(S) = \begin{cases} \mu(x) & \text{if } x \in \mathrm{STC}(A) \\ S(x) & \text{if } x \notin \mathrm{STC}(A). \end{cases}
$$

Finally, let A^μ be the event that A is true under the valuation given by the x^μ.

Lemma 1.7.12. *For any event $B \subset \{\hat{q}_r < \infty\}$ entirely dependent on the variables*

in $\mathcal{X}' \setminus \mathcal{V}_v$,

$$\Pr_{\hat{q}(r)<\infty}(\tau_v^{-1}([v])|B) \leq P^*([v]).$$

Proof. Fix $v \in T$. For each $\mu \in \mathcal{M}([v])$, let $R(\mu)$ be the set of all valuations ν of $\mathrm{RSP}([v])$ such that if $x = \mu(x)$ and $y = \nu(x)$ for all $x \in \mathrm{STC}([v])$ and all $y \in \mathrm{RSP}([v])$, then $[v]$ is true. Then, $\{\tau_v^{-1}(E_\mu) : \mu \in \mathcal{M}([v])\}$ form an open cover of $\{\hat{q}_{[v]} < \infty\}$, so we get

$$\begin{aligned}\Pr_{\hat{q}(r)<\infty}(\tau_v^{-1}([v])|B) &= \sum_{\mu\in\mathcal{M}([v])} \Pr_{\hat{q}(r)<\infty}(\tau_v^{-1}([v])|B \wedge \tau_v^{-1}(E_\mu)) \Pr_{\hat{q}(r)<\infty}(\tau_v^{-1}(E_\mu)) \\ &= \sum_{\mu\in\mathcal{M}([v])} \sum_{\nu\in R(\mu)} \Pr_{\hat{q}(r)<\infty}(\tau^{-1}(E_\nu)|B \wedge \tau_v^{-1}(E_\mu)) \Pr_{\hat{q}(r)<\infty}(\tau_v^{-1}(E_\mu)).\end{aligned}$$

We have that $B \wedge \tau_v^{-1}(E_\mu)$ is independent from $\mathcal{V}_v$ because both B and $\tau_v^{-1}(E_\mu)$ depend entirely on the variables in $\mathcal{X}' \setminus \mathcal{V}_v$. This is true for B by assumption and for $\tau_v^{-1}(E_\mu)$ due to Lemma 1.7.10. Because $\tau_v^{-1}(E_\nu)$ depends entirely on the variables in $\mathcal{V}_v$, have that $\tau_v^{-1}(E_\nu)$ and $B \wedge \tau_v^{-1}(E_\mu)$ are independent. Thus, continuing the above calculation yields

$$\begin{aligned}\Pr_{\hat{q}(r)<\infty}(\tau_v^{-1}([v])|B) &= \sum_{\mu\in\mathcal{M}([v])} \sum_{\nu\in R(\mu)} \Pr_{\hat{q}(r)<\infty}(\tau^{-1}(E_\nu)) \Pr_{\hat{q}(r)<\infty}(\tau_v^{-1}(E_\mu)) \\ &= \sum_{\mu\in\mathcal{M}([v])} \sum_{\nu\in R(\mu)} \Pr_{\Omega}(E_\nu) \Pr_{\hat{q}(r)<\infty}(\tau_v^{-1}(E_\mu)) \\ &= \sum_{\mu\in\mathcal{M}([v])} \Pr_{\Omega}([v]|E_\mu) \Pr_{\hat{q}(r)<\infty}(\tau_v^{-1}(E_\mu)) \\ &\leq \sum_{\mu\in\mathcal{M}([v])} P^*([v]) \Pr_{\hat{q}(r)<\infty}(\tau_v^{-1}(E_\mu)) \\ &= P^*([v]).\end{aligned}$$

□

To apply Lemma 1.7.12 to bound Line 1.7.11, it remains to show that the event $\bigwedge_{r \geq j > i} [v_j]^{\tau_{v_j}}$ depends only on the variables in $\mathcal{X}' \setminus \mathcal{V}_{v_i}$ for all i.

Proof. Suppose that $S_1, S_2 \in \Omega'$ agree on $\mathcal{X}' \setminus \mathcal{V}_{v_i}$. Then, they also agree on $\mathcal{X}' \setminus \mathcal{V}$, so $\widehat{\log}(S_1) = \widehat{\log}(S_2)$. Therefore, $\hat{V}_k(n)(S_1) = \hat{V}_k(n)(S_2)$ for each k and n. For each $j > i$, we have that $\hat{q}(v_j) > \hat{q}(v_i)$, so $\hat{V}_{\hat{q}(v_j)}(n) > \hat{V}_{\hat{q}(v_i)}(n)$ for each $x_n \in \mathrm{RSP}([v_i])$. Therefore, $\{x_n^{\hat{V}_{\hat{q}(v_j)}(n)} : x_n \in \mathrm{VBL}([v_j])\}$ is disjoint from $\{x_n^{\hat{V}_{\hat{q}(v_i)}(n)} : x_n \in \mathrm{RSP}([v_i])\}$, so $[v_j]^{\tau_n}$ depends only on the variables in $\mathcal{X}' \setminus \mathcal{V}_{v_i}$. □

Finally, we obtain

$$\Pr(\text{The } T \text{ check passes}) \leq \prod_{v \in T} P^*([v])$$

and therefore that

$$\Pr(\text{The resample algorithm produces } T) \leq \prod_{v \in T} P^*([v]).$$

The rest of the proof goes exactly as the proof of Theorem 1.3.1 did, starting from Claim 1.3.10 with $\mathcal{A}$ replaced by $\mathcal{B}$ from the statement of Theorem 1.7.3. This completes the proof of Theorem 1.7.3.

1.7.2 Computable Lefthanded Local Lemma

We now state the computability conditions for the Computable Lefthanded Local Lemma. They are very similar to the computability conditions for the Computable Local Lemma, Theorem 1.4.2, but with a few instance of VBL replaced with RSP.

We require that for each $x \in \mathcal{X}$, the set $\{A \in \mathcal{A} : x \in \mathrm{RSP}(A)\}$ is finite. This is an improvement to Theorem 1.4.2 in that $\mathrm{RSP}(A) \subset \mathrm{VBL}(A)$. We also need the following computability conditions:

1. The events $\mathcal{A} = \{A_1, A_2, A_3, ...\}$ and the sets $\mathrm{RSP}(A_i)$ are computably presented. That is, $\mathrm{VBL}(A_i)$, $\mathrm{RSP}(A_i)$ and the finite set of assignments of the variables in $\mathrm{VBL}(A_i)$ that make A_i true are all uniformly computable with respect to i.

2. Each x_i has a rational-valued probability distribution that is uniformly computable with respect to i.

3. The code for the finite set of indices $\{j : x_i \in \mathrm{RSP}(A_j)\}$ is uniformly computable given i.

4. The order $\prec$ is computable.

Theorem 1.7.13 (Computable Lefthanded Local Lemma)**.** *Let $\mathcal{X}$ and $\mathcal{A}$ satisfy the conditions of Theorem 1.7.3 as well as the conditions above. Additionally, require that there exists a rational constant $\alpha \in (0, 1)$ such that, for each $A \in \mathcal{A}$,*

$$P^*(A) \leq \alpha z(A) \prod_{B \in \Gamma(A)} (1 - z(B)).$$

Then, there is a computable assignment of $\mathcal{X}$ under which each $A \in \mathcal{A}$ is false.

Proof. Just as for Theorem 1.4.2, we show that the resample algorithm constitutes a layerwise computable mapping. The proof is completely identical. □

1.7.3 Applying the Computable Lefthanded Local Lemma

We now prove computable versions of Theorems 1.2.1 and 1.2.2.

Theorem 1.7.14 (Computable Version of Theorem 1.2.1)**.** *Given arbitrary small $\varepsilon > 0$, there is some N_ε and a computable $\{0,1\}$-valued sequence such that any two identical intervals of length $n > N_\varepsilon$ have distance greater than $f(n) = (2-\varepsilon)^n$.*

Proof. Fix $\varepsilon > 0$ and $N > 0$. We will show that if N is large enough, then $N_\varepsilon = N$ witnesses the theorem. We set up the application of Theorem 1.7.13. Let $\mathcal{X} = \{x_1, x_2, ...\}$ be the bits in our binary sequence. Recall that $A_{k,l,n}$ is the event that $[k, k+n-1] = [l, l+n-1]$. Note that $A_{k,l,n}$ being false implies that $A_{k,l,m}$ is false for each $m > n$. It is therefore enough to only consider $A_{k,l,n}$ such that n is minimal for $l-k$, that is, such that $l-k = \lceil f(n) \rceil$. Let $\mathcal{A} = \{A_{k,l,n} : l-k = \lceil f(n) \rceil \text{ and } n > N\}$. Let $\mathrm{RSP}(A_{k,l,n}) = [l, l+n-1]$. Then, $\mathrm{STC}(A_{k,l,n}) = [k, k+n-1] \setminus [l, l+n-1]$. We first check the additional computability requirements:

- For each x_i, there are fewer than i^2 many elements in $\{A_{k,l,n} \in \mathcal{A} : x_i \in \mathrm{RSP}(A_{k,l,n})\} \subset \{A_{k,l,\lceil f(l-k) \rceil} : k < l \leq i\}$, so the set of events that have x_i in their resample sets is finite.
- From $A_{k,l,n} \in \mathcal{A}$, we have that $\mathrm{VBL}(A_{k,l,n})$, $\mathrm{RSP}(A_{k,l,n})$ and the set of valuations of $\mathrm{VBL}(A_{k,l,n})$ satisfying $A_{k,l,n}$ are all uniformly computable.
- Each x_i is a fair coin flip, and therefore has uniformly computable distribution.
- We can uniformly compute $\{(k,l,n) : x_i \in \mathrm{RSP}(A_{k,l,n})\}$.

To check the algorithmic conditions, we confirm that

- $\mathrm{RSP}(A_{k,l,n})$ is an interval

- $\max(\mathrm{STC}(A_{k,l,n})) < \min(\mathrm{RSP}(A_{k,l,n}))$.

Finally, we check the probabilistic conditions. Let $P^*(A_{k,l,n}) = 2^{-n}$. We need to show that for any valuation μ of $\mathrm{STC}(A_{k,l,n})$,

$$P^*(A_{k,l,n}) \geq \Pr(A_{k,l,n}|E_\mu).$$

In fact, $P^*(A_{k,l,n}) = \Pr(A_{k,l,n}|E_\mu)$. This is easy to see if $\mathrm{RSP}(A_{k,l,n}) \cap [k, k+n) = \emptyset$. If $\mathrm{RSP}(A_{k,l,n}) \cap [k, k+n) \neq \emptyset$, suppose the entries of $[k, l)$ are given by valuation μ and $l = k+j$. Then for $A_{k,l,n}$ to be true, x_{l+i} must equal $\mu(x_{k+i})$ for all $i \leq j$ as in the case where the compared intervals don't overlap. Then, when we consider the first non-overlap x_{l+j}, still do not have a choice, because x_{l+j} must be equal to x_l, whose value must be $\mu(x_k)$. Thus, there is only one valuation of the variables in $\mathrm{RSP}(A_{k,l,n})$ under which $A_{k,l,n}$ is true when the variables in $\mathrm{STC}(A_{k,l,n})$ have values given by μ. So, $\Pr(A_{k,l,n}|E_\mu) = 2^{-n} = P^*(A_{k,l,n})$.

Finally, we show that the main local lemma condition holds. As in the proof of Theorem 1.2.4, let $z(A_{k,l,n}) = \frac{1}{f(n)n^3}$. We need to show that there is $\alpha \in (0,1)$, such that each $A_{k_0,l_0,n_0} \in \mathcal{A}$,

$$P^*(A_{k_0,l_0,n_0}) = 2^{-n_0} \leq \alpha z(A_{k_0,l_0,n_0}) \prod_{A_{k,l,n} \in \Gamma(A_{k_0,l_0,n_0})} (1 - z(A_{k_0,l_0,n_0})).$$

Up to the additional constant factor of α which disappears in the limit, the proof of this is identical to the corresponding part in the proof of Theorem 1.2.4 as the neighborhoods are smaller while the probabilities are the same. □

Theorem 1.7.15 (Computable Version of Theorem 1.2.2)**.** *Given arbitrary small $\varepsilon > 0$ there is some N_ε and a computable $\{0,1\}$-valued sequence $a_1, a_2, a_3, \ldots$ such that*

any two adjacent intervals of length $n > N_\varepsilon$ differ in at least $(\frac{1}{2} - \varepsilon)n$ many places; that is, for each k and $n > N_\varepsilon$, $a_{k+i} \neq a_{k+n+i}$ for at least $(\frac{1}{2} - \varepsilon)n$ many i with $0 \leq i < n$.

Proof. Fix $\varepsilon > 0$ and $N > 0$. We will show that if N is large enough, then $N_\varepsilon = N$ witnesses the theorem. We again apply the lefthanded computable Lovász local lemma. Let $\mathcal{X} = \{x_1, x_2, ...\}$ be the bits in our binary sequence. Recall that $A_{k,n}$ is the event that blocks $[k, k+n-1]$ and $[k+n, k+2n-1]$ share at least $(\frac{1}{2} + \varepsilon)n$ many entries. Let $\mathcal{A} = \{A_{k,n} : n > N\}$ and $\mathrm{RSP}(A_{k,n}) = [k+n, k+2n-1]$. Then, $\mathrm{STC}(A_{k,n}) = [k+n, k+2n-1]$. We first check the additional computability requirements:

- For each x_i, there are fewer than i^2 many elements in $\{A_{k,n} \in \mathcal{A} : x_i \in \mathrm{RSP}(A_{k,n})\} \subset \{A_{k,n} : k+n \leq i\}$, so the set of events that have x_i in their resample sets is finite.
- $\mathcal{A}$, $\mathrm{RSP}(A_{k,n})$ and $\mathrm{VBL}(A_{k,n})$ are uniformly computable.
- Each x_i is a fair coin flip, and therefore has uniformly computable distribution.
- We can uniformly compute $\{(k,n) : x_i \in \mathrm{RSP}(A_{k,n}\}$.

To check the algorithmic conditions, we confirm that

- $\mathrm{RSP}(A_{k,l,n})$ is an interval
- $\max(\mathrm{STC}(A_{k,l,n})) < \min(\mathrm{RSP}(A_{k,l,n}))$.

Finally, we check the probabilistic conditions. The probability that $\mathrm{RSP}(A_{i,n})$ shares

exactly k entries with $\mathrm{STC}(A_{i,n})$ is $\binom{n}{k}2^{-k}$, so we set

$$P^*(A_{i,n}) = \Pr(A_{i,n}) = 2^{-n} \sum_{r=\lceil(\frac{1}{2}+\varepsilon)n\rceil}^{n} \binom{n}{r}.$$

We need to show that for any valuation μ of $\mathrm{STC}(A_{i,n})$,

$$P^*(A_{k,l,n}) \geq \Pr(A_{i,n}|E_\mu).$$

This follow directly from our original calculation of $\Pr(A_{i,n})$, since $\mathrm{RSP}(A_{k,n}) \cap [k, n+k-1] = \emptyset$ for all k and n, so $\Pr(A_{i,n}|E_\mu) = \Pr(A_{i,n})$.

Finally, it remains to show that the local lemma condition holds. Let $z_{i,n} = \frac{b^n}{n}$ for some $\alpha < b < 1$. Fix A_{i_0,n_0}. $\Gamma(A_{i_0,n_0}) = \{A_{i,n} : i_0 + n_0 - 2n + 1 \leq i \leq i_0 + 2n_0 - 1\}$, so it suffices for the Lovasz local lemma to check that

$$\frac{1}{2}b^n \prod_{n=N_\varepsilon}^{\infty} \prod_{i=i_0+n_0-2n+1}^{i_0+2n_0-1-m} (1-b^m) \geq n_0\alpha^{n_0}.$$

for large enough N_ε, with $\frac{1}{2}$ being the $\alpha < 1$ in from the computable lefthanded Lovász local lemma. Up to the additional factor of $\frac{1}{2}$, the proof of this is near-identical to the corresponding part in the proof of Theorem 1.2.7. □

1.8 Binary Sequence Games

Theorem 1.7.13 lets us apply the local lemma to sets of events which depend on arbitrarily long initial segments of the $\mathcal{X}$ other than the variables that we resample. We can make full use of this property by having a hypothetical opponent react to our

choices for the x_i. Consider a game called the *binary sequence game* in which two players take turns selecting bits in a binary sequence. Player 1 picks the odd bits and player 2 picks the even bits. The binary sequence game generates the sequence

$$a_1a_2a_3a_4a_5a_6\cdots = e_1d_2e_3d_4e_5d_6\ldots$$

where e_{2n+1} is chosen by player 1 with knowledge of all preceding bits in the sequence (but not future bits) and player 2 chooses d_{2n} similarly. Pegden (2011) studied Theorems 1.2.1 and 1.2.2 in the context of binary sequence games. He found that player 1 has a strategy to ensure that binary sequence game produces a non-repetitive sequence, as stated below:

Theorem 1.8.1 ([26]). *For every $\varepsilon > 0$ there is an N_ε such that Player 1 has a strategy in the binary sequence game ensuring that any two identical blocks $[i, i+n-1] = x_ix_{i+1}x_{i+2}\ldots x_{i+n-1}$ and $[j, j+n-1] = x_jx_{j+1}x_{j+2}\ldots x_{j+n-1}$ of length $n > N_\varepsilon$ have distance at least $f(n) = (2-\varepsilon)^{n/2}$. That is, if $x_{i+s} = x_{j+s}$ for all $0 \le s < n$ and $n > N_\varepsilon$, then $|i-j| < (2-\varepsilon)^{n/2}$.*

Theorem 1.8.2 ([26]). *For every $\varepsilon > 0$ there is an N_ε such that Player 1 has a strategy in the binary sequence game ensuring that any two adjacent blocks $[i, i+n-1]$ and $[i+n, i+2n-1]$ of length $n > N_\varepsilon$ differ in at least $n\left(\frac{1}{4}-\varepsilon\right)$ places. That is, if $n > N_\varepsilon$ then $x_{i+s} \neq x_{i+n+s}$ for at least $n\left(\frac{1}{4}-\varepsilon\right)$ many $0 \le s < n$.*

Unlike Theorems 1.2.1 and 1.2.2, even the finite versions of these theorems cannot be proven using the classical local lemma. This is because player two can change their moves depending on what player one does. This means that, when setting up the local lemma to find a strategy for player 1, each event depends on every move

that came before it. For example in the context of Theorem 1.8.1, the event $A_{k,l,n}$ that the intervals $[i, i+n-1]$ and $[l, l+n-1]$ are identical depends not only on the moves player 1 makes in those intervals, but on every move player 1 has made beforehand. This explodes the size of the neighborhood relation. To prove Theorems 1.8.1 and 1.8.2, Pegden introduces an extension of the LLL called the lefthanded local lemma. Pegden uses the lefthanded local lemma to prove the following finite version of Theorem 1.8.1.

Proposition 1.8.3. *For every $\varepsilon > 0$ there is an N_ε such that, for each $M > 0$ and player 2 strategy g in the binary sequence game of length M length, there is a sequence $e_1 e_3 e_5 ... e_M$ of player 1 moves that when played against Φ, any two identical blocks of length $n > N_\varepsilon$ in the resulting sequence have distance at least $(2-\varepsilon)^{n/2}$.*

The full Theorem 1.8.1 results from the following compactness argument:

Proof of 1.8.1. By open determinacy, it is sufficient to show that there is no winning strategy for player 2. Fix player 2 strategy g. Fix game length $M > 0$. By the proposition, there is a string of player 1 moves such that g does not win within M moves. Let T be the tree of such strings of player 1 moves. T is a finitely branching tree, so by König's lemma, there is an infinite path through T, so f is not a winning strategy for player 2. Since player 2 has no winning strategy and they are playing an open game, player 1 has a winning strategy by open determinacy. □

The proof of Theorem 1.8.2 follows the same pattern.

This proof is non-constructive on account of the use of both König's lemma and open determinacy. By using Theorem 1.7.13, we can effectivize this use of König's lemma: we will show that T has an f-computable path. It remains open whether or not effective versions of Theorems 1.8.1 and 1.8.2 are true.

To prove that T has an f-computable path, we once again apply Theorem 1.7.13.

To set up the local lemma in the proofs of the following two theorems, fix player 2 strategy g. Let the probability space be $2^{2\mathbb{N}+1} = \{e = e_1, e_3, e_5, ...\}$ with $\{x = x_1, x_3, x_5, ...\}$ as a random variable $x : 2^{2\mathbb{N}+1} \to \{0,1\}$ such that $x_{2i+1}(e) = e_{2i+1}$. Then, we can interpret player 2's moves $d_2, d_4, d_6, ...$ as functions $2^{2\mathbb{N}+1} \to \{0,1\}$ with $d_{2i}(e) = g(e_1, e_3, ..., e_{2i-1})$. Likewise, our sequence $a(x) = a_1(x), a_2(x), a_3(x), ...$ is a sequence of random variables with

$$a_k(e) = \begin{cases} x_k(e) & \text{if } k = 2n+1 \text{ for some } n \in \mathbb{N} \\ d_k(e) & \text{otherwise} \end{cases}.$$

Theorem 1.8.4. *For every $\varepsilon > 0$ there is an N_ε such that for each player 2 strategy g in the binary sequence game, there is a g-computable sequence $e_1e_3e_5...$ of player 1 moves that when played against g, any two identical blocks of length $n > N_\varepsilon$ in the resulting sequence have distance at least $f(n) = (2-\varepsilon)^{n/2}$.*

Proof. Fix $\varepsilon > 0$ and $N > 0$. We will show that if N is large enough, then $N_\varepsilon = N$ witnesses the theorem. Define $A_{k,l,n}$ as before. Since all of player 1s previous moves can effect the truth of $A_{k,l,n}$, $\text{VBL}(A_{k,l,n}) = [0, l+n) \cap \mathcal{X} = \{x_{2i+1} : 2i+1 < l+n\}$. Set $\text{RSP}(A_{k,l,n}) = [l, l+n-1] \cap \mathcal{X}$. Let $\mathcal{A} = \{A_{k,l,n} : l - k = \lceil f(n) \rceil \text{ and } n > N\}$. For the same reason as in the proof for the non-game version, avoiding each $A_{k,l,n} \in \mathcal{A}$ is sufficient. Although we do not need to, we will assume that N is large enough so that $f(n) > n$ for all $n > N$. Then, $l - k > n$ for each $A_{k,l,n} \in \mathcal{A}$, so $[k, k+n-1] \cap [l, l+n-1] = \emptyset$ for each $A_{k,l,n} \in \mathcal{A}$. The first things to check are the computability requirements of Theorem 1.7.13.

- For each x_{2i+1}, there are fewer than i^2 many elements in $\{A_{k,l,n} \in \mathcal{A} : x_{2i+1} \in$

$\mathrm{RSP}(A_{k,l,n})\} \subset \{A_{k,l,\lceil f(l-k)\rceil} : k < l \leq 2i+1\}$, so the set of events that have x_{2i+1} in their resample sets is finite.

- From $A_{k,l,n} \in \mathcal{A}$, $\mathrm{VBL}(A_{k,l,n})$ and $\mathrm{RSP}(A_{k,l,n})$ are uniformly computable. To determine the set of valuations of $\mathrm{VBL}(A_{k,l,n})$ which satisfy $A_{k,l,n}$, we need information about g, so these valuations are uniformly g-computable.

- Each x_i is a fair coin flip, and therefore has uniformly computable distribution.

- We can uniformly compute $\{(k,l,n) : x_i \in \mathrm{RSP}(A_{k,l,n})\}$.

The algorithmic conditions are true for the same reason as they were for the proof of the computable version of Theorem 1.2.1. To check the probabilistic conditions, we start with the P^* condition. Fix $A_{k,l,n} \in \mathcal{A}$. Our guiding principle is to pick $P^*(A_{k,l,n})$ as small as possible such that, for each valuation μ of $\mathrm{STC}(A_{k,l,n})$, $\Pr(A_{k,l,n} \mid E_\mu) \leq P^*(A_{k,l,n})$. Player 1 has control of half of the bits, so it seems intuitive that $P^*(A_{k,l,n}) = 2^{-(n-1)/(2)}$ is satisfactory. To see this, fix μ that is a valuation of $\mathrm{STC}(A_{k,l,n})$. Then, since $\mathrm{STC}(A_{k,l,n})$ is an initial segment of player 1 moves, μ fixes the initial segment $a_1, a_2, ..., a_{\min(\mathrm{RSP}(A_{k,l,n}))-1}$. Thus, for each $S, S' \in E_\mu$ and $i \leq \max(\mathrm{STC}(A_{k,l,n}))$, we have that $a_i(S) = a_i(S')$. Let $\mu(a_i)$ be this constant $a_i(S)$. Let $E_{\mu,A_{k,l,n}}$ be the event that for each $x_{2i+1} \in \mathrm{RSP}(A_{k,l,n})$, $x_{2i+1} = \mu(a_{2i+1-(l-k)})$. Then $(A_{k,l,n} \cap E_\mu) \subset E_{\mu,A_{k,l,n}}$. Trivially, $(A_{k,l,n} \cap E_\mu) \subset E_\mu$, so

$$(A_{k,l,n} \cap E_\mu) \subset (E_{\mu,A_{k,l,n}} \cap E_\mu).$$

Since E_μ and $E_{\mu,A_{k,l,n}}$ are statements about the of values of disjoint sets of independent

random variables, they are independent. Therefore,

$$\begin{aligned}\Pr(A_{k,l,n}|E_\mu) &= \frac{\Pr(A_{k,l,n} \cap E_\mu)}{Pr(E_\mu)} \\ &\leq \frac{\Pr(E_{\mu,A_{k,l,n}} \cap E_\mu)}{\Pr(E_\mu)} \\ &= \Pr(E_{\mu,A_{k,l,n}}).\end{aligned}$$

$|\operatorname{RSP}(A_{k,l,n})| \geq (n-1)/(2)$, so $\Pr(A_{k,l,n}|E_\mu) \leq \Pr(E_{\mu,A_{k,l,n}}) \leq 2^{-(n-1)/(2)}$, as desired.

Next we are left to show that there is $z : \mathcal{A} \to (0,1)$ such that

$$P^*(A_{k,l,n}) \leq z(A_{k,l,n}) \prod_{B \in \Gamma(A_{k,l,n})} (1 - z(B)). \tag{1.8.5}$$

The proof of Equation 1.8.5 is almost identical to the corresponding part of the proof of Theorem 1.2.1, with the definition of $f(n)$ updated. □

Theorem 1.8.6. *For every $\varepsilon > 0$ there is an N_ε such that for each player 2 strategy g in the binary sequence game, there is a g-computable sequence $e_1e_3e_5...$ of player 1 moves that when played against g, any two adjacent blocks of length $n > N_\varepsilon$ in the resulting sequence differ in at least $n(\frac{1}{4} - \varepsilon)$ many entries.*

Proof. Fix $\varepsilon > 0$ and $N > 0$. We will show that if N is large enough, then $N_\varepsilon = N$ witnesses the theorem. For each $k, n \in \mathbb{N}$ with $k < l$, let $A_{k,n}$ be the event that blocks $a_k, a_{k+1}, ..., a_{k+n-1}$ and $a_{k+n}, a_{k+n+1}, ..., a_{k+2n-1}$ share at least $(\frac{3}{4} + \varepsilon)n$ many entries. Then, $\operatorname{VBL}(A_{k,n}) = ([0, k+2n) \cap \mathcal{X}) = \{x_{2i+1} : k \leq 2i+1 \leq k+2n-1\}$. Set $\operatorname{RSP}(A_{k,n}) = [k+n, k+2n) \cap \mathcal{X}$. Let $\mathcal{A} = \{A_{k,n} : n > N\}$. First, we check the computability requirements of Theorem 1.7.13.

- For each x_{2i+1}, there are fewer than $(2i+1)^2$ many elements in $\{A_{k,n} \in \mathcal{A} :$

$x_{2i+1} \in \mathrm{RSP}(A_{k,n})\} \subset \{A_{k,n} : k+n \leq 2i+1\}$, so the set of events that have x_{2i+1} in their resample sets is finite.

- From $A_{k,n} \in \mathcal{A}$, $\mathrm{VBL}(A_{k,n})$ and $\mathrm{RSP}(A_{k,n})$ are uniformly computable. To determine the set of valuations of $\mathrm{VBL}(A_{k,n})$ which satisfy $A_{k,n}$, we need information about g, so these valuations are uniformly g-computable.

- Each x_i is a fair coin flip, and therefore has uniformly computable distribution.

- We can uniformly compute $\{(k,n) : x_i \in \mathrm{RSP}(A_{k,n})\}$.

The algorithmic conditions are true for the same reason as they were for the proof of the computable version of Theorem 1.2.2. To check the probabilistic conditions, we start with the P^* condition. Fix $A_{k,n} \in \mathcal{A}$. Our guiding principle is to pick $P^*(A_{k,l,n})$ as small as possible such that, for each valuation μ of $\mathrm{STC}(A_{k,n})$, $\Pr(A_{k,n} \mid E_\mu) \leq P^*(A_{k,n})$. Player 1 has control of half of the bits, and we cannot rely on player 2 not matching any bits, so we choose

$$P^*(A_{k,n}) = \frac{1}{2^{\lfloor n/2 \rfloor}} \sum_{r=\lceil (\frac{1}{4}+\varepsilon)n \rceil}^{\lfloor n/2 \rfloor} \binom{\lfloor n/2 \rfloor}{r}.$$

To see this suffices, fix μ that is a valuation $\mathrm{STC}(A_{k,n})$. Then, since $\mathrm{STC}(A_{k,n})$ is an initial segment of player 1 moves, μ fixes the initial segment $a_1, a_2, ..., a_{k+n-1}$. Thus, for each $S, S' \in E_\mu$ and $i \leq (k+n-1)$, we have that $a_i(S) = a_i(S')$. Let $\mu(a_i)$ be this constant $a_i(S)$. Fix $S \in E_\mu$. Then, $a_i(s) = \mu(a_i)$ for each $i \leq k+n-1$. If $S \in A_{k,n}$, then it is necessary that player 1s moves in $[k+n, k+2n)$ match at least $\lfloor (\frac{1}{4}+\varepsilon)n \rfloor$ many bits on their turn, even if player 2 always matches the corresponding bit each of their at most $\lceil n/2 \rceil$ moves. $P^*(A_{k,n})$ is the probability that player 1 matches at

least $\lfloor(\frac{1}{4}+\varepsilon)n\rfloor$ bits, which is required for $A_{k,n}$ to be true.

Next we are left to show that there is $z : \mathcal{A} \to (0,1)$ such that

$$P^*(A_{k,l,n}) \leq z(A_{k,l,n}) \prod_{B\in\Gamma(A_{k,l,n})} (1 - z(B)). \tag{1.8.7}$$

The proof of Equation 1.8.7 is almost identical to the corresponding part of the proof of theorem 1.2.2. □

Player 2 has a similar ability to defeat fixed sequences.

Proposition 1.8.8. *Let $\{e^k_{2i+1}\}_{i,k\in\omega}$ be a $\{0,1\}$-valued matrix. Then, for any odd N, there is a sequence $\{d_{2i}\}_{i\in\omega} \leq_T \{e^k_{2i+1}\}_{i,k\in\omega}$ such that, for any k, the sequence*

$$a_1a_2a_3... = e^k_1d_2e^k_3d_4e^k_5...$$

has at least one pair of identical adjacent intervals of length N.

Proof. We will show that for each k, we can defeat e^k_{2i} with finitely many moves. The full statement follows by concatenating these moves together.

We will compare the interval $[1, N]$ with $[N + 1, 2N]$. Thus, we need $a_i = a_{i+N}$ for $1 \leq i \leq N$. For $1 \leq t \leq \frac{N}{2}$, let $d_{2t} = e_{2t+N}$ and let $d_{1+N+2t} = e_{2t+1}$. Then, for even $i = 2t$, we have that $a_i = d_{2t} = e_{2t+N} = a_{i+N}$ and for odd $i = 2t + 1$ we have that $a_i = e_{2t+1} = d_{1+N+2t} = a_{2t+1+N}$. □

1.9 Conclusion

We conclude with some open questions arising from this work.

There is scant evidence to suggest that the conditions of Theorems 1.7.3 and 1.7.13 are optimal for making the modified resample algorithm a layerwise computable mapping. In particular, the interaction between the RSP sets and $\prec$ is somewhat unwieldy and part of the reason why the RSP sets must be intervals that cannot precede any element of their corresponding STC sets.

Question 1.9.1. How can the conditions of Theorems 1.7.3 and 1.7.13 be loosened?

While we can computably defeat any player 2 strategy in the binary sequence games, it is still unknown whether the indeterminacy argument that pastes these winning moves into a full strategy is computable.

Question 1.9.2. Are there computable winning strategies for the binary sequence games in Theorems 1.8.4 and 1.8.6?

The effective LLL has been applied to several problems with some relation to Hindman's theorem (HT) [19, 22, 8]. Restricted forms of HT have been of interest because they are often difficult to prove without proving the entirety of HT. It is possible that Theorem 1.7.13 has some applications in this area as well.

Question 1.9.3. How can Theorem 1.7.13 be applied in reverse math and computability theory in general?

Chapter 2

Cohesive Powers

2.1 Introduction

Cohesive powers are an effective analogue of the ultrapower construction. Fix a computable structure $\mathcal{M}$, and consider the behavior of partial computable functions $\varphi \to |\mathcal{M}|$. In general, φ can behave in a very disorderly way. Remarkably, if one looks at the behavior of φ on a cohesive set, they will find a well defined structure sharing many properties with $\mathcal{M}$.

An infinite set $C \subset \omega$ is cohesive if, for each computably enumerable set W, we have that $C \subset^* W$ or $C \subset^* \overline{W}$. Fix a computable language $\mathcal{L}$, a computable $\mathcal{L}$-structure $\mathcal{M}$ and a cohesive set C. The cohesive power $\prod_C \mathcal{M}$ of $\mathcal{M}$ over C is defined over the equivalence classes of $\sim_C = \{(\varphi, \psi) : \varphi \text{ and } \psi \text{ are partial computable functions and } C \subset^* \{x : \varphi(x)\downarrow = \psi(x)\downarrow\}\}$. The interpretations of the constant, function, and relation symbols are defined in a natural way.

Definition 2.1.1. The cohesive power $\prod_C \mathcal{M}$ of $\mathcal{M}$ over C is defined as follows.

- Let $D = \{\varphi \mid \varphi : \omega \to M$ is a partial computable function and $C \subset^* \mathrm{dom}(\varphi)\}$
- For $\varphi, \psi \in D$, define equivalence relation $\sim_C$ by $\varphi \sim_C \psi$ if and only if $C \subset^* \{x : \varphi(x)\downarrow = \psi(x)\downarrow\}$. Denote the equivalence class of φ under $\sim_C$ by $[\varphi]$.
- The domain of $\prod_C \mathcal{M}$ is $\{[\varphi] : \varphi \in D\}$.
- Let R be an n-ary relation symbol of $\mathcal{L}$. Interpret R in $\prod_C \mathcal{M}$ by

$$\prod_C \mathcal{M} \models R([\varphi_1], \ldots, [\varphi_n]) \text{ if and only if}$$

$$C \subset^* \{x : \varphi_i(x)\downarrow \text{ for all } i < n \text{ and } \mathcal{M} \models R(\varphi_1(x), \ldots, \varphi_n(x))\}.$$

- Let f be an n-ary function symbol of $\mathcal{L}$. Interpret f in $\prod_C \mathcal{M}$ by

$$f^{\prod_C \mathcal{M}}([\varphi_1], \ldots, [\varphi_n]) = [\psi]$$

where

$$\psi(x) = \begin{cases} f^{\mathcal{M}}(\varphi_1(x), \ldots, \varphi_n(x)) & \text{if } \varphi_i(x)\downarrow \text{ for } 1 \leq i \leq n \\ \uparrow & \text{otherwise.} \end{cases}$$

- Let c be a constant symbol of $\mathcal{L}$. Interpret c in $\prod_C \mathcal{M}$ by

$$c^{\prod_C \mathcal{M}} = [\psi]$$

where $\psi(x) = c^{\mathcal{M}}$ for all x.

Rumen Dimitrov [10] introduced cohesive powers and proved the following analogue to Łoś's theorem.

Theorem 2.1.2 (Fundamental Theorem of Cohesive Powers [10])**.** *Let $\mathcal{A}$ be a computable $\mathcal{L}$-structure and C be a cohesive set. Then,*

1. *Let $\Psi(x_1, \ldots, x_m)$ be a boolean combination of Σ_1 and Π_1 formulas. Then, for any $[\varphi_1], \ldots, [\varphi_m] \in \prod_C \mathcal{A}$,*

$$\prod_C \mathcal{A} \models \Phi([\varphi_1], \ldots, [\varphi_m]) \iff$$
$$C \subset^* \left\{ i : i \in \bigcap_{j \leq m} dom(\varphi_j) \text{ and } \mathcal{A} \models \Phi(\varphi_1(i) \ldots, \varphi_m(i)) \right\}.$$

2. *Let Φ be a Σ_3 $\mathcal{L}$-sentence. Then $\mathcal{A} \models \Phi$ implies $\prod_C \mathcal{A} \models \Phi$.*

3. *Let Φ be a boolean combination of Σ_2 and Π_2 $\mathcal{L}$-sentences. Then, $\prod_C \mathcal{A} \models \Phi$ if and only if $\mathcal{A} \models \Phi$.*

Every cohesive power $\Pi_C \mathcal{M}$ contains a copy of $\mathcal{M}$. We embed $\mathcal{M}$ into $\prod_C \mathcal{M}$ by the mapping sending $a \in M$ to th total constant functions which always outputs a. We call this the canonical embedding. Part 3 of Theorem 2.1.2 holds with parameters from $\mathcal{M}$ interpreted by the canonical embedding.

In this chapter, we classify the cohesive powers of computable equivalence structures (Theorem 2.2.3) and computable injection structures (Theorem 2.2.6). Finally, we extend a result of Dimitrov, Harizanov, Morozov, Shafer, Soskova, and Vatev [12] concerning the finite condensation of cohesive powers of computable linear orders of order type ω to also apply to orders of type ζ.

2.2 Some Classifications of Cohesive Powers

2.2.1 Equivalence Structures

Dimitrov, Harizanov, Morozov, Shafer, Soskova, and Vatev [11] study the isomorphism types of cohesive powers of linear orders. Cohesive powers of equivalence relations are relatively simpler.

Definition 2.2.1. An equivalence structure is a structure in the language $\mathcal{L} = \{\equiv\}$ such that $\equiv$ is a transitive, reflexive, and symmetric binary relation.

Since being transitive, reflexive, and symmetric is a Π_1 property, Theorem 2.1.2 tells us that $\prod_C \mathcal{M}$ is an equivalence structure for every cohesive set C and computable equivalence structure $\mathcal{M}$. In the language of equivalence structures, the existence of at least m equivalence classes of size n is expressible by a Σ_2 formula, so the existence of *exactly* m equivalance classes of size n is expressible by a boolean combination of Σ_2 and Π_2 formulas. Therefore, Theorem 2.1.2 tells us that cohesive powers preserve the number of equivalence classes of each finite size. Since cohesive powers are countable structures, analysis of the number of countable equivalence classes is enough to obtain a complete categorization. For an equivalence structure $\mathcal{M}$ and $a \in |\mathcal{M}|$, let $[a]_{\mathcal{M}}$ denote the $\equiv_{\mathcal{M}}$-equivalence class of a.

Theorem 2.2.2. *Let $\mathcal{M}$ be a computable equivalence structure and let C be a cohesive set. Then,*

1. *$\mathcal{M}$ and $\prod_C \mathcal{M}$ have the same number of equivalence classes of each finite size.*
2. *$\prod_C \mathcal{M}$ has an infinite equivalence class if and only if, for each $N \in \mathbb{N}$, $\mathcal{M}$ has an equivalence class of size larger than N.*

3. $\prod_C \mathcal{M}$ *has infinitely many infinite equivalence classes if and only if* $\mathcal{M}$ *has an infinite family* $\mathcal{F}$ *of equivalence classes such that for each* $N \in \mathbb{N}$*, the set* $\{\mathcal{E} \in \mathcal{F} : |\mathcal{E}| > N\}$ *is infinite.*

Proof. As mentioned above, (1) follows immediately from Theorem 2.1.2.

To prove (2), first suppose that $\prod_C \mathcal{M}$ has an infinite equivalence class. The sentence saying that x is in an equivalence class of size at least n is Σ_1, so the sentence

$$A_n : (\exists x)(x \text{ is in an equivalence class of size at least } n)$$

is also Σ_1. By assumption, $\prod_C \mathcal{M} \models A_N$ for every N, hence, by Theorem 2.1.2, $\mathcal{M} \models A_N$ for every N, which is exactly what we wanted to show.

Now, suppose that for each $N \in \mathbb{N}$, $\mathcal{M}$ has an equivalence class of size larger than N. We can construct a computable list $a_1, a_2, \cdots \in \mathcal{M}$ such that $||[a_n]|| \geq 2n$ for each n by searching $|\mathcal{M}|$ until we have found witnesses guaranteeing large enough equivalence classes. Then, $|[a_N] \setminus \{a_1, \ldots, a_{N-1}\}| \geq N$. Then, define $\varphi(i) = a_i$ for all i. To see that the equivalence class of $[\varphi]$ is infinite, define

$$\varphi_j(i) = \begin{cases} a_i & \text{if } i < j \\ \text{The } j\text{'th element of } [a_i] & \text{if } i \geq j \end{cases}.$$

Then, each φ_n is pointwise $\equiv^{\mathcal{A}}$-equivalent to φ and eventually pointwise non-equal to φ_m for each $m \neq n$. Hence, $[\varphi]$ has an infinite equivalence class in $\prod_C \mathcal{M}$.

To prove (3), first suppose that $\mathcal{M}$ has such a family $\mathcal{F}$ of equivalence classes. We first note that there exists a computable matrix $\{a_i^j\}$ with each $a_i^j \in |\mathcal{M}|$ and with the properties

- Each $[a_i^j]_{\mathcal{M}}$ has at least i elements.
- For each $j' \neq j$, we have $a_i^j \not\equiv^{\mathcal{M}} a_{i'}^{j'}$

We can compute such a matrix because, for each i, there are infinitely many non-equivalent elements of $\mathcal{M}$ whose equivalence class has size greater than i. We can enumerate these elements and confirm that they are not equivalent computably, allowing us to chose an a_i^j for each j. More formally, at step $s = \langle i, j \rangle$, pick a_i^j such that $|[a_i^j]| > i$ and $a_i^j \not\equiv^{\mathcal{M}} a_{i'}^{j'}$ for each $\langle i', j' \rangle < s$.

Let $\varphi^j(i) = a_i^j$. The first condition on a_i^j allows us to apply the argument in part (2) of this theorem to show that $[\varphi^j]$ has an infinite equivalence class. The second condition on a_i^j ensures that $[\varphi^j] \not\equiv_{\prod_C \mathcal{M}} [\varphi^{j'}]$ for each $j \neq j'$. Hence, $\prod_C \mathcal{M}$ has infinitely many infinite equivalence classes.

Now suppose that $\prod_C \mathcal{M}$ has infinitely many infinite equivalence classes. Let $B_{n,m}$ be a formula saying that there are at least m many equivalence classes of size at least n. Thus, $\prod_C \mathcal{M} \models B_{n,m}$ for each n and m. Also, $B_{n,m}$ is equivalent to a Σ_1 formula, so by Theorem 2.1.2, we have that

$$\mathcal{M} \models B_{n,m} \text{ for every } n, m \in \mathbb{N}.$$

The set of all equivalence classes of witnesses to the $B_{n,m}$ in $\mathcal{M}$ constitute an instance of the required family $\mathcal{F}$. □

We can restate Theorem 2.2.2 as conditions for when $\mathcal{M} \cong \prod_C \mathcal{M}$.

Theorem 2.2.3. *Let $\mathcal{M}$ be a computable equivalence structure and let C be a cohesive set. Then, $\prod_C \mathcal{M} \ncong \mathcal{M}$ if and only if both of the following hold.*

- *$\mathcal{M}$ has finitely many countable equivalence classes and*
- *For each $N \in \omega$, there is $n > N$ such that $\mathcal{M}$ has an equivalence class of size $n > N$.*

Furthermore, if $\prod_C \mathcal{M} \ncong \mathcal{M}$ then $\prod_C \mathcal{M}$ has countably many infinite equivalence classes.

Proof. Suppose that $\mathcal{M}$ has finitely many countable equivalence classes and for each $N \in \omega$, there is $n > N$ such that $\mathcal{M}$ has an equivalence class of size $n > N$. Then, by Theorem 2.2.2, the cohesive power $\prod_C \mathcal{M}$ has infinitely many countable equivalence classes and hence $\prod_C \mathcal{M} \ncong \mathcal{M}$.

Now suppose that $\prod_C \mathcal{M} \ncong \mathcal{M}$. We have already seen that $\mathcal{M}$ and $\prod_C \mathcal{M}$ have the same amount of each finite equivalence class. Hence, they must have different amounts of countable equivalence classes. Suppose for contradiction that $\mathcal{M}$ has a family $\mathcal{F}$ of infinitely many infinite equivalence classes. Then, $\mathcal{F}$ satisfies the condition of part (3) of Theorem 2.2.2, so $\prod_C \mathcal{M}$ has infinitely many infinite equivalence classes, a contradiction, so $\prod_C \mathcal{M} \ncong \mathcal{M}$ implies that $\mathcal{M}$ has finitely many countable equivalence classes.

Now suppose for contradiction that there is $N \in \omega$ such that all equivalence classes of $\mathcal{M}$ have size less than N. Then, $\prod_C \mathcal{M}$ has only finite equivalence classes. Since the number equivalence classes of each finite size is preserved by cohesive powers, we have that $\prod_C \mathcal{M} \cong \mathcal{M}$, a contradiction. Hence, $\prod_C \mathcal{M} \ncong \mathcal{M}$ implies that for each $N \in \omega$, there is $n > N$ such that $\mathcal{M}$ has an equivalence class of size $n > N$. $\square$

2.2.2 Injection Structures

Definition 2.2.4. An injection structure is a structure in the language $\mathcal{L} = \{f\}$ such that f is an injective unary function.

Cenzer, Harizanov, and Remmel [7] study computability theoretic properties of injection structures. Injection structures are similar to equivalence relations because they can be decomposed into finite cycles, ω-chains and $\mathbb{Z}$-chains. If $\mathcal{M}$ is a computable injection structure, define equivalence relation $\sim$ by $x \sim y$ if and only if x and y are part of the same cycle, ω-chain or $\mathbb{Z}$-chain. Then, $\sim$ is computably enumerable. Similarly to equivalence relations, if $\mathcal{M}$ has exactly m cycles of size n, then the same is true for $\prod_C \mathcal{M}$, since having at least m cycles of size n is expressible with a Σ_1 formula so having exactly m cycles of size n is expressible with a a boolean combination of Σ_1 and Π_1 formulas. Having at least m elements without a predecessor is expressible using a boolean combination of Σ_1 and Π_1 formulas, so having exactly m elements without a predecessor is expressible using a boolean combination of Σ_2 and Π_2 formulas. Therefore, the number of ω chains is also preserved. To obtain a full categorization, it remains to describe the number of $\mathbb{Z}$-chains in $\prod_C \mathcal{M}$.

Theorem 2.2.5. *$\prod_C \mathcal{A}$ has no $\mathbb{Z}$-chains if and only if $\mathcal{A}$ consists entirely of finite cycles whose sizes are all below some upper bound $N \in \mathbb{N}$. Otherwise, $\prod_C \mathcal{A}$ has infinitely many $\mathbb{Z}$-chains.*

Proof. For the backwards direction, suppose that $\mathcal{A}$ consists entirely of finite cycles whose sizes are all below some upper bound $N \in \mathbb{N}$. Let $A_n(x)$ be a quantifier free formula saying that x is in a cycle of length less then or equal to n. Then, $\mathcal{A} \models \forall x A_N(x)$, so by Theorem 2.1.2, $\prod_C \mathcal{A} \models \forall x A_N(x)$, so $\prod_C \mathcal{A}$ has no $\mathbb{Z}$-chains.

For the forwards direction, suppose that $\mathcal{A}$ does not consist entirely of finite cycles whose sizes are all below some upper bound $N \in \mathbb{N}$. Thus, we can compute a sequence $a_1, a_2, \cdots \in |\mathcal{A}|$ such that $f^i(a_i) \neq f^j(a_i)$ for each i and each $0 \leq j < i$. We can compute such a sequence by setting a_i to be the first element found such that $f^j(a_i) \neq a_i$ for all $0 < j \leq i$. Using this sequence, we will show that $\prod_C \mathcal{A}$ has infinitely many $\mathbb{Z}$-chains.

Define partial computable function $\varphi(i) = f^i(a_i)$. We have that, for each n, the set $\{i : (\exists j < n) f^n(\varphi(i)) = f^j(\varphi(i))\}$ is finite, $C \subset^* \{i : f^n(\varphi(i)) \neq \varphi(i)\}$, so $[\varphi]$ has infinitely many successors by Theorem 2.1.2. Similarly, the n'th predecessor of $[\varphi]$ is given by the equivalence class of

$$\psi(i) = \begin{cases} f^{i-n}(a_i) & \text{if } i \geq n \\ a_i & \text{otherwise} \end{cases}.$$

Thus, $[\varphi]$ is part of a $\mathbb{Z}$-chain.

For each rational number $n/m \in \mathbb{Q} \cap [0, 1]$, we can find an element of another $\mathbb{Z}$-chain by setting $\varphi_{\frac{n}{m}}(i) = f^{\lfloor \frac{ni}{m} \rfloor}(a_i)$. □

Along with the preceding discussion on ω-chains and cycles, Theorem 2.2.5 gives us the following categorization of cohesive powers of injection structures.

Theorem 2.2.6. *Let $\mathcal{M}$ be a computable injection structure and let C be a cohesive set. Then*

- *$\mathcal{M}$ and $\prod_C \mathcal{M}$ have exactly the same number of ω-chains and exactly the same number of each finite cycle of each size.*
- *$\prod_C \mathcal{A}$ has no $\mathbb{Z}$-chains if and only if $\mathcal{A}$ consists entirely of finite cycles whose*

sizes are all below some upper bound $N \in \mathbb{N}$. Otherwise, $\prod_C \mathcal{A}$ has infinitely many $\mathbb{Z}$-chains.

Therefore, $\mathcal{M} \cong \prod_C \mathcal{M}$ if and only if $\mathcal{M}$ contains only cycles whose sizes have uniform bound $N \in \mathbb{N}$.

Proof. Immediate from Theorem 2.2.5. □

2.3 Finite Condensation of Cohesive Powers of ζ

In this section, we categorize finite condensations of cohesive powers of copies of ζ by co-c.e. cohesive sets. The finite condensation of a linear order is the structure of the linear order when points that have only finitely many points in between them are identified.

Definition 2.3.1. Fix a linear order $\mathcal{A}$. For $x, y \in \mathcal{A}$, let $[x, y] = \{z \in |\mathcal{A}| : x \leq z \leq y\}$. For $x \in |\mathcal{A}|$, let $\mathbf{c}_{\mathrm{F}}(x) = \{y : [x, y]$ is finite or$y \leq x$ and $[y, x]$ is finite$\}$. The finite condensation $\mathbf{c}_{\mathrm{F}}(\mathcal{A})$ is the linear order on the image of $\mathbf{c}_{\mathrm{F}}$ on the universe of $\mathcal{A}$ induced by the order relation of $\mathcal{A}$.

For a linear order $\mathcal{A}$, let $\mathbf{c}_{\mathrm{F}}(A)$ be the finite condensation of $\mathcal{A}$. Also, we denote by $\mathbf{c}_{\mathrm{F}}^{\mathcal{A}}$ the function $\mathbf{c}_{\mathrm{F}} : \mathcal{A} \to \mathbf{c}_{\mathrm{F}}(\mathcal{A})$ specifically for $\mathcal{A}$ when it is not clear from the context. We also write $[x, y]^{\mathcal{A}}$ to be the interval between x and y taken as elements of $\mathcal{A}$.

Lemma 2.3.2. *Let $\mathcal{A}$ be a linear order and $I \subset A$ be an interval. Then, $\boldsymbol{c}_F^{\mathcal{A}}(I) \cong \boldsymbol{c}_F(I)$.*

Proof. We define an isomorphism $\varphi : \mathbf{c}_{\mathrm{F}}(I) \to \mathbf{c}_{\mathrm{F}}^{\mathcal{A}}(I)$ by $\varphi(\mathbf{c}_{\mathrm{F}}^{I}(x)) = \mathbf{c}_{\mathrm{F}}^{\mathcal{A}}(x)$. To see that this is well defined, suppose that $x, y \in I$ and that $\mathbf{c}_{\mathrm{F}}^{I}(x) = \mathbf{c}_{\mathrm{F}}^{I}(y)$ and without loss of generality that $x \leq y$. Then, $[x, y]^{I}$ is finite. Since I is in interval in $\mathcal{A}$, we have that $[x, y]^{I} = [x, y]^{\mathcal{A}}$, so $[x, y]^{\mathcal{A}}$ is also finite. Therefore, $\mathbf{c}_{\mathrm{F}}^{\mathcal{A}}(x) = \mathbf{c}_{\mathrm{F}}^{\mathcal{A}}(y)$.

To see that φ is onto, fix $P \in \mathbf{c}_{\mathrm{F}}^{\mathcal{A}}(I)$. Then there is $x \in I$ such that $\mathbf{c}_{\mathrm{F}}^{A}(x) = P$ and hence $\varphi(\mathbf{c}_{\mathrm{F}}^{I}(x)) = P$. To see that φ is 1-1, fix $P, Q \in \mathbf{c}_{\mathrm{F}}(I)$ and suppose that $\varphi(P) = \varphi(Q)$. Fix $p, q \in I$ such that $p < q$ and $\mathbf{c}_{\mathrm{F}}^{I}(p) = P$ and $\mathbf{c}_{\mathrm{F}}^{I}(q) = Q$. Then, $\mathbf{c}_{\mathrm{F}}^{\mathcal{A}}(p) = \varphi(P) = \varphi(Q) = \mathbf{c}_{\mathrm{F}}^{\mathcal{A}}(q)$, so the interval $[p, q]^{\mathcal{A}}$ is finite. Because $I \subset \mathcal{A}$, we have that $[p, q]^{I}$ is also finite. Thus, $\mathbf{c}_{\mathrm{F}}^{I}(p) = \mathbf{c}_{\mathrm{F}}^{I}(q)$ and hence $P = Q$. The map φ is order-preserving because both $\mathbf{c}_{\mathrm{F}}(\mathcal{A})$ and $\mathbf{c}_{\mathrm{F}}(I)$ inherit their ordering from $\mathcal{A}$. $\square$

Let $\mathcal{A}, \mathcal{B}, \ell_1, \ell_2, \ell_3$, and ℓ_4 be structures in the language of linear orders. Suppose that $\mathbf{c}_{\mathrm{F}}(\mathcal{A}) \cong \ell_1 + \ell_2$ and that $\mathbf{c}_{\mathrm{F}}(\mathcal{B}) \cong \ell_3 + \ell_4$. We abuse notation to interchange elements of $\mathbf{c}_{\mathrm{F}}(\mathcal{A})$ and $\mathbf{c}_{\mathrm{F}}(\mathcal{B})$ with their images in $\ell_1 + \ell_2$ and $\ell_3 + \ell_4$ respectively. Define $f_{\ell_1} : \mathbf{c}_{\mathrm{F}}(\mathcal{A}) \to \ell_1 + 1$ by

$$f_{\ell_1}(a) = \begin{cases} a & \text{if } a \in \ell_1 \\ 1 & \text{if } a \in \ell_2 \end{cases}$$

and $f_{\ell_4} : \mathbf{c}_{\mathrm{F}}(\mathcal{B}) \to 1 + \ell_4$ by

$$f_{l_4}(b) = \begin{cases} b & \text{if } b \in \ell_4 \\ 1 & \text{if } b \in \ell_3 \end{cases}$$

Finally, define $\varphi : \mathbf{c}_{\mathrm{F}}(\mathcal{A}+\mathcal{B}) \to \ell_1 + 1 + \ell_4$ by

$$\varphi(x) = \begin{cases} f_{\ell_1}(\mathbf{c}_{\mathrm{F}}(a)) & \text{if } (\exists a \in A)(x = \mathbf{c}_{\mathrm{F}}(i(a))) \\ f_{\ell_4}(\mathbf{c}_{\mathrm{F}}(b)) & \text{if } (\exists b \in B)(x = \mathbf{c}_{\mathrm{F}}(i(b))) \end{cases}.$$

Note that, for any $x \in \mathbf{c}_{\mathrm{F}}(A+B)$, if both conditions in the piecewise definition of $\varphi(x)$ hold, then $f_{\ell_1}(\mathbf{c}_{\mathrm{F}}(a)) = f_{\ell_4}(\mathbf{c}_{\mathrm{F}}(b)) = 1$, so φ is well defined.

Lemma 2.3.3. *Let $\mathcal{A}, \mathcal{B}, \ell_1, \ell_2, \ell_3$, and ℓ_4 be structures in the language of linear orders. Suppose that $\boldsymbol{c}_F(\mathcal{A}) \cong \ell_1 + \ell_2$ and that $\boldsymbol{c}_F(\mathcal{B}) \cong \ell_3 + \ell_4$. Then, the following diagram commutes:*

$$\begin{array}{ccccc} \mathcal{A} & \xrightarrow{\boldsymbol{c}_F} & \boldsymbol{c}_F(\mathcal{A}) & & \\ \downarrow{\scriptstyle i} & & & \searrow^{f_{\ell_1}} & \\ (\mathcal{A}+\mathcal{B}) & \xrightarrow{\boldsymbol{c}_F} & \boldsymbol{c}_F(\mathcal{A}+\mathcal{B}) & \xrightarrow{\varphi} & (\ell_1 + 1 + \ell_4) \\ \uparrow{\scriptstyle i} & & & \nearrow_{f_{\ell_4}} & \\ \mathcal{B} & \xrightarrow{\boldsymbol{c}_F} & \boldsymbol{c}_F(\mathcal{B}) & & \end{array}.$$

Furthermore, $\boldsymbol{c}_F(\mathcal{A}+\mathcal{B})$ can be decomposed into $\boldsymbol{c}_F(\mathcal{A}+\mathcal{B}) = \mathcal{L}_1 + \mathcal{L}' + \mathcal{L}_4$ where $\varphi(\mathcal{L}_1) = \ell_1$, $\varphi(\mathcal{L}_4) = \ell_4$ and $\mathcal{L}' \cong \boldsymbol{c}_F[(\boldsymbol{c}_F^{\mathcal{A}})^{-1}(\ell_2) + (\boldsymbol{c}_F^{\mathcal{B}})^{-1}(\ell_3)]$

Proof. That the diagram commutes follows directly from the definition of φ (there are only two loops). Let $\mathcal{L}' = \varphi^{-1}(1)$, $\mathcal{L}_1 = \varphi^{-1}(\ell_1)$ and $\mathcal{L}_4 = \varphi^{-1}(\ell_4)$. To see that $\mathbf{c}_{\mathrm{F}}(\mathcal{A}+\mathcal{B}) = \mathcal{L}_1 + \mathcal{L}' + \mathcal{L}_4$ fix $x_1 \in \mathcal{L}_1$, $x' \in \mathcal{L}'$ and $x_4 \in \mathcal{L}_4$. We need to show that $x_1 < x' < x_4$. $x_1 < x_4$ because $i^{-1}(\mathbf{c}_{\mathrm{F}}^{-1}(x_1)) \subset \mathcal{A}$ and $i^{-1}(\mathbf{c}_{\mathrm{F}}^{-1}(x_4)) \subset \mathcal{B}$. $x_1 < x'$ because $\mathbf{c}_{\mathrm{F}}(i^{-1}(\mathbf{c}_{\mathrm{F}}^{-1}(x_1))) \subset \ell_1 \subset \mathcal{A}$ and $\mathbf{c}_{\mathrm{F}}(i^{-1}(\mathbf{c}_{\mathrm{F}}^{-1}(x'))) \subset \ell_2 \cup \ell_3$. Similarly, $\mathbf{c}_{\mathrm{F}}(i^{-1}(\mathbf{c}_{\mathrm{F}}^{-1}(x_4))) \subset l_4$, so $x' < x_4$.

Because the diagram commutes,

$$\mathbf{c}_{\mathrm{F}}^{\mathcal{A}+\mathcal{B}}[i((\mathbf{c}_{\mathrm{F}}^{\mathcal{A}})^{-1}(\ell_2)) \cup i((\mathbf{c}_{\mathrm{F}}^{\mathcal{B}})^{-1}(\ell_3))] = \varphi^{-1}[f_{\ell_1}(\ell_2) \cup f_{\ell_4}(\ell_3)].$$

The right hand side is equal to $\mathcal{L}'$. On the right hand side, we can replace the $\cup$ with $+$ because $i((\mathbf{c}_{\mathrm{F}}^{\mathcal{A}})^{-1}(\ell_2) <_{\mathcal{A}+\mathcal{B}} i((\mathbf{c}_{\mathrm{F}}^{\mathcal{B}})^{-1}(\ell_3))$. By the previous lemma,

$$\mathbf{c}_{\mathrm{F}}^{\mathcal{A}+\mathcal{B}}[i((\mathbf{c}_{\mathrm{F}}^{\mathcal{A}})^{-1}(\ell_2) + i((\mathbf{c}_{\mathrm{F}}^{\mathcal{B}})^{-1}(\ell_3))] \cong \mathbf{c}_{\mathrm{F}}[i((\mathbf{c}_{\mathrm{F}}^{\mathcal{A}})^{-1}(\ell_2)) + i((\mathbf{c}_{\mathrm{F}}^{\mathcal{B}})^{-1}(\ell_3))].$$

Finally, i is an injection, so

$$\mathbf{c}_{\mathrm{F}}[i((\mathbf{c}_{\mathrm{F}}^{\mathcal{A}})^{-1}(\ell_2) + i((\mathbf{c}_{\mathrm{F}}^{\mathcal{B}})^{-1}(\ell_3))] \cong \mathbf{c}_{\mathrm{F}}[(\mathbf{c}_{\mathrm{F}}^{\mathcal{A}})^{-1}(\ell_2) + (\mathbf{c}_{\mathrm{F}}^{\mathcal{B}})^{-1}(\ell_3)].$$

Thus, $\mathcal{L}' \cong \mathbf{c}_{\mathrm{F}}[(\mathbf{c}_{\mathrm{F}}^{\mathcal{A}})^{-1}(l_2) + (\mathbf{c}_{\mathrm{F}}^{\mathcal{B}})^{-1}(\ell_3)]$, as desired. □

We combine Lemma 2.3.3 with a several theorems of Dimitrov, Harizanov, Morozov, Shafer, Soskova, and Vatev.

Theorem 2.3.4 ([11])**.** *For any linear orders $\mathcal{A}_1$, $\mathcal{A}_2$ and cohesive set C, we have*

- $\prod_C(\mathcal{A}_1 + \mathcal{A}_2) \cong \prod_C \mathcal{A}_1 + \prod_C \mathcal{A}_2$,
- $\prod_C \mathcal{A}_1^* \cong (\prod_C \mathcal{A}_1)^*$, *where * is the reversal operator,*

Proposition 2.3.5. *Let $\mathcal{A} \cong \zeta$ be a computable linear order and C be a cohesive, co-c.e. set. Then $\boldsymbol{c}_F(\Pi_C\mathcal{A}) \cong \eta$*

Proof. Fix $n \in A$. Let $U = \{x \in A : x \geq_{\mathcal{A}} n\}$ and $L = \{x \in A : x <_{\mathcal{A}} n\}$. Then $U \cong \omega$ and $L \cong \omega^*$. C is cohesive and co-c.e., so by theorems 4.4 and 4.5 of

[11], $\mathbf{c}_{\mathrm{F}}\Pi_C U \cong \mathbf{c}_{\mathrm{F}}\Pi_C L^* \cong 1+\eta$ with $\mathbf{c}_{\mathrm{F}}^{-1}(1) \cong \omega$. Then, by Lemma 2.3.3 and the operations from Theorem 2.3.4,

$$\begin{aligned}
\mathbf{c}_{\mathrm{F}}\left(\Pi_C \mathcal{A}\right) &\cong \eta^* + \mathbf{c}_{\mathrm{F}}[(\mathbf{c}_{\mathrm{F}}^L)^{-1}(1) + (\mathbf{c}_{\mathrm{F}}^U)^{-1}(1)] + \eta \\
&\cong \eta + \mathbf{c}_{\mathrm{F}}[\omega^* + \omega] + \eta \\
&\cong \eta + 1 + \eta \\
&\cong \eta.
\end{aligned}$$

□

Chapter 3

Compositional Second Order Part of a Problem in the Weihrauch Degrees

3.1 Introduction and the First Order Part of a Problem

Here we give a brief and narrow introduction to Weihrauch reducibility. For a more comprehensive view, see [14, 6].

In this chapter, we are interested in reducibility between problems.

Definition 3.1.1. A **problem** on Baire space is a partial multi-function P whose domain is a subset of $\mathbb{N}^{\mathbb{N}}$ and, for each $X \in \mathrm{dom}(P)$, the value P takes on X is a subset $P(X) \subset \mathbb{N}^{\mathbb{N}}$. We call elements of $\mathrm{dom}(P)$ *instances* of P and for $X \in \mathrm{dom}(P)$, we call each $Y \in P(X)$ a *solution* of P for X. We often write $P :\subseteq \mathbb{N}^{\mathbb{N}} \rightrightarrows \mathbb{N}^{\mathbb{N}}$ to denote a problem. The $\subseteq$ indicates that P is a partial on $\mathbb{N}^{\mathbb{N}}$ and the $\rightrightarrows$ indicate that

P is a multi-function.

Example 3.1.2. The problem RT^1_2 is the problem whose instances are $c : \mathbb{N} \to 2$ and whose solutions to c are an infinite $Y \subset \mathbb{N}$ such that $c(i) = c(j)$ for all $i, j \in Y$. The problem RT^1_2 represents Ramsey's Theorem for colorings of singletons by two colors.

In reverse mathematics, we are interested measuring when solving one problem P is reducible to solving another problem Q. This is a measure of relative strengths between the theorems "Every instance of Q has a solution" and "Every instance of P has a solution". One way of quantifying this is via Weihrauch reducibility.

Definition 3.1.3. Let P and Q be problems. Then P **is Weihrauch reducible to** Q if and only if there are Turing functionals Φ and Ψ such that for each $X \in \mathrm{dom}(P)$ we have that $\Phi(X) = \hat{X} \in \mathrm{dom}(Q)$ and for each $\hat{Y} \in Q(\hat{X})$ we have that $\Psi(\hat{Y}, X) \in P(X)$. If so, we write $P \leq_W Q$.

$$\begin{array}{ccc} X \in \mathrm{dom}(P) & \xrightarrow{\Phi} & \hat{X} \in \mathrm{dom}(Q) \\ \vdots_{P} \downarrow & & \downarrow_{Q} \\ Y \in P(X) & \xleftarrow[\Psi]{} & \hat{Y} \in Q(\hat{X}) \end{array} .$$

The relation $\leq_W$ is a pre-partial order on the space of problems. Hence, $\leq_W$ induces an equivalence relation $\equiv_W$ that divides the space of problems into equivalence classes that we call Weihrauch degrees.

Definition 3.1.4 (First Order Problems)**.** P is a first order problem if each of its solutions is a singleton set of natural numbers, e.g. $\{2\}$.

Below, whenever we write Turing functionals as part of the instances or solutions to a problem, we think each functional as each being represented by its index as a single natural number.

Definition 3.1.5 (First Order Part of a Problem [15])**.** Let P be a problem. The first order part of P, denoted by ${}^1P$, is defined to be the problem whose instances are triples $\langle f, \Phi, \Psi \rangle$ such that $f \in \omega^\omega$ and Φ and Ψ are Turing functionals such that $\Phi(f) \in \mathrm{dom}(P)$ and $\Psi(f, g)(0)\downarrow$ for all $g \in P(\Phi(f))$.

The ${}^1P$ solutions to $\langle f, \Phi, \Psi \rangle$ are all y with $\Psi(f, g)(0) \downarrow = y$ for some $g \in P(\Phi(f))$.

Dzhafarov, Solomon, and Yokoyama [15] showed that ${}^1P$ is maximal among first order problems reducible to P.

Theorem 3.1.6 ([15])**.** ${}^1P = \sup_{\leq_W}\{Q : Q \leq_W P \text{ and } Q \text{ is first order.}\}$.

It turns out that $\mathrm{RT}^1_2 \equiv_W {}^1\,\mathrm{RT}^1_2$. However, both are computable, in the sense that for every instance X of RT^1_2, there is a $Y \leq_T X$ with $Y \in \mathrm{RT}^1_2(X)$. It is also that case that *all* first order problems R are computable (assuming that there is a solution for every instance). To see this, fix $X \in \mathrm{dom}(R)$. Then, since R is first order, $R(X) \subset \mathcal{N} = \{\{1\}, \{2\}, \dots\}$. Each element of $\mathbb{N}$ is computable, so each element of $R(X)$ is computable from X.

The discussion in the previous paragraph can be summarized by saying that, for a computable problem P,

$$ {}^1P \equiv_W P.$$

We can interpret this to mean that "extracting" the first order part from computable problems reduces them to triviality. On the other hand, non-computable problems must have something "left over" once you extract their first order part. It is natural to ask whether this idea of "extraction" can be well posed.

Question 3.1.7. Let P be a problem. Is there a problem Q with trivial first order part such that $P \equiv_W {}^1P \boxdot Q$, with $\boxdot$ being some kind of product on Weihrauch

degrees?

The parallel product $\times$ is a natural candidate for $\boxdot$. If such a Q exists for all P under the operation $\times$, then Q would be some sort of quotient of P with ${}^1P$. We now define one of the products that we will consider.

Definition 3.1.8. Let P and Q be problems. The *parallel product* $P \times Q$ is the problem whose domain is the direct product $\mathrm{dom}(P) \times \mathrm{dom}(Q)$ and whose solutions to (X_1, X_2) are (Y_1, Y_2) such that $Y_1 \in (X_1)$ and $Y_2 \in P(X_2)$.

Quotients over the parallel product were first considered by Dzhafarov, Goh, Hirschfeldt, Patey, and Pauly [13], who introduce a notion of a parallel quotient between two problems.

Definition 3.1.9 (Quotient of a Problem [13])**.** For problems $P \geq_W Q$, the parallel quotient of their Weihrauch degrees is given by $P/Q := \sup_{\leq_W}\{R : R \times Q \leq_W P\}$, when this supremum exists.

This definition suggests an upper parallel quotient.

Definition 3.1.10. For problems $P \geq_W Q$, the upper parallel quotient of their Weihruach degrees is given by $P/_u Q := \inf_{\leq_W}\{R : R \times Q \geq P\}$, when this infimum exists.

The upper parallel quotient is an interesting notion because if $P/Q \equiv_W P/_u Q$, then the quotient is in fact a decomposition of $P \equiv_W Q \times P/Q$. However, there is no reason to expect the upper and lower quotients to always coincide, and as noted by Dzhafarov, Goh, Hirschfeldt, Patey, and Pauly, nor is there any reason expect that they always exist.

The compositional product is another candidate for $\boxdot$. We give the definition below.

Definition 3.1.11. Let P and Q be problems. The *compositional product* $P \star Q$ is the problem whose instances are of the form (f, Θ, Γ) such that $\Theta(f) \in \mathrm{dom}(Q)$ and for each $Y_1 \in Q(\Theta(f))$ we have that $\Gamma(f, Y) \in \mathrm{dom}(P)$. Solutions to (f, Θ, Γ) are of the form (Y_1, Y_2) such that $Y_1 \in Q(\Theta(f))$ and $Y_2 \in P(\Gamma(f, Y_1))$.

The same considerations about the parallel product apply for upper and lower compositional quotients. Additionally, the compositional product is not commutative, so we also have notions of left and right quotients for both the upper and lower variations.

Definition 3.1.12. Let $P \geq_W Q$ be problems. Then, define

- the lower left compositional quotient by $P \star^{-1}_{\ell,\ell} Q := \sup_{\leq_W}\{R : Q \star R \leq_W P\}$, when this supremum exists,
- the upper left compositional quotient by $P \star^{-1}_{u,\ell} Q := \inf_{\leq_W}\{R : Q \star R \geq_W P\}$, when this infimum exists,
- the lower right compositional quotient by $P \star^{-1}_{\ell,r} Q := \sup_{\leq_W}\{R : R \star Q \leq_W P\}$, when this supremum exists,
- the upper right compositional quotient by $P \star^{-1}_{u,r} Q := \inf_{\leq_W}\{R : R \star Q \geq_W P\}$, when this infimum exists,

It turns out that $P \star^{-1}_{u,\ell} {}^1P$ is always defined and belongs to a special class of problems whose first order parts are relatively weak compared to the problem itself.

Definition 3.1.13 (Purely Oracular)**.** Let P be a problem. We say that P is *purely oracular for*

- *the parallel product* if, for all Q, we have that ${}^1P \times Q \geq_W P$ implies that $Q \geq_W P$.
- *composition form the right* if, for all Q, we have that $Q \star {}^1P \geq_W P$ implies that $Q \geq_W P$.
- *composition from the left* if, for all Q, we have that ${}^1P \star Q \geq_W P$ implies that $Q \geq_W P$.

P being purely oracular for an operation captures the notion that the first order part of P cannot assist in computing P via that operation. Note that being purely oracular for either form of composition implies being purely oracular for the parallel product.

In this chapter, we show that $P \star^{-1}_{u,\ell} {}^1P$ always exists and is purely oracular for composition from the left. In fact, we will show that $P \star^{-1}_{u,\ell} R$ exists for any first order R.

3.2 Compositional Second Order Part

We begin by showing that one can always find the upper left compositional quotient by a first order problem. First, we require introduce notation that allows our solutions to be given in a c.e. way.

Definition 3.2.1. Let $Y \in \mathbb{N}^{\mathbb{N}}$. Define $\vec{Y} :\subseteq \mathbb{N} \to \mathbb{N}$ by $\vec{Y}(n) = Y(\langle n, t+1 \rangle)$ where t is least such that $Y(\langle n, t \rangle) \neq 0$. If $\vec{Y}$ is total, then we also think of $\vec{Y}$ as an element

of $\mathbb{N}^{\mathbb{N}}$.

For a Turing functional Ψ, let $\hat{\Psi}$ be the Turing functional such that

$$\hat{\Psi}(Y)(\langle n,t\rangle) = \begin{cases} 0 & \text{if } \Psi_t(Y)(n)\uparrow \\ \Psi(Y)(n) & \text{if } \Phi_t(Y)(n)\downarrow. \end{cases}$$

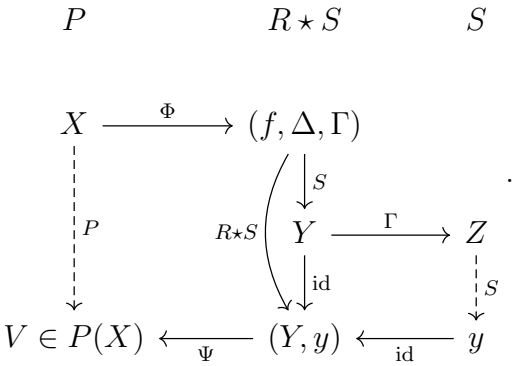
Theorem 3.2.2. *Let P be a problem and let R be a first order problem. Then, the upper left compositional quotient $P \star^{-1}_{u,\ell} R$ always exists.*

Proof. Let Q be the problem whose instances are elements $X \in \mathrm{dom}(P)$ and whose solutions are of the form $(\langle Y_0, Y_1, \dots\rangle, Z)$ for $Z \in \mathrm{dom}(R)$ such that for each $y \in R(Z)$ we have that $\vec{Y_y} \in P(X)$.

We claim that $Q \equiv_W P \star^{-1}_{u,\ell} R$. To prove this, first we show that $P \leq_W R \star Q$. Let $X \in \mathrm{dom}(P)$. Let $\Gamma = \pi_2 \circ \pi_2$ be the Turing functional that takes the second component of the second component of its input. Send X to the instance (X, id, Γ) of $R \star Q$ such that for all $W = (\langle Y_1, Y_2, \dots\rangle, Z) \in Q(X)$ we have that $\Gamma(X, W) = Z$, an instance of R. Therefore the solutions to (X, id, Γ) are all of the form $((\langle Y_0, Y_1, \dots\rangle, Z), y)$ such that $y = R(Z)$ implies that $\vec{Y_y} \in P(X)$, solving our instance of P.

Now we show that $Q = \inf_{\leq_W}\{S : R \star S \geq_W P\}$. Suppose that $R \star S \geq_W P$ via Φ and Ψ. Our calculations follow the diagram below. Fix $X \in \mathrm{dom}(P)$. Then, $\Phi(X) = (f, \Delta, \Gamma) \in \mathrm{dom}(R \star S)$ such that $\Gamma(f, Y) = Z \in \mathrm{dom}(R)$ for each $Y \in S(\Delta(f))$. Furthermore, for each $Y \in S(\Delta(f))$ and each $y \in R(Z)$, we have that

$\Psi(X, Y, y) \in P(X)$.



We use this reduction to construct a reduction from Q to S. Let $X \in \mathrm{dom}(Q) = \mathrm{dom}(P)$. We have that $\Phi(X) = (f, \Delta, \Gamma)$. Send X to $\Delta(f) \in \mathrm{dom}(S)$. For $Y \in S(\Delta(f))$, we have that $\Gamma(f, Y) = Z \in \mathrm{dom}(R)$. Return $(\langle \hat{\Psi}(X, Y, 0), \hat{\Psi}(X, Y, 1), \ldots \rangle, Z) \in Q(X)$. □

In particular, we can always divide P by ${}^1P$.

Corollary 3.2.3. *Let P be a problem. Then there exists problem ${}^2P = P \star^{-1}_{u,\ell} {}^1P := \inf \leq_W \{R : {}^1P \star R \geq_W P\}$.*

For the sake of being explicit, we describe the instances and solutions of an equivalent formulation of ${}^2P$.

Let P be a problem. The second order part ${}^2P$ of P is the problem whose instances are all $X \in \mathrm{dom}(P)$ and whose solutions to X are all triples $(\langle Y_1, Y_2, \ldots, \rangle, Z, \Xi)$ such that

- $Z \in \mathrm{dom}(P)$
- $\Xi(Z, W)(0)\downarrow$ for all $W \in P(Z)$.
- For all $W \in P(Z)$, if $\Xi(Z, W)(0)\downarrow = y$, then $Y_y \in P(X)$.

Because $P \geq_W {}^2P$, we also have that $P \geq_W {}^1({}^2P)$, so we immediately obtain the following corollary.

Corollary 3.2.4. *Let P be a problem. Then, ${}^2P$ is purely oracular for composition from the left.*

Proof. Suppose that $S \star R \geq_W {}^2P$ for some first order problem S such that ${}^2P \geq_W S$. Then we have that ${}^1P \star (S \star R) \geq_W P$. We use monotonicity and associativity of the compositional product. Since $P \geq_W {}^2P \geq_W S$ and ${}^1P \star S$ is a first order problem, we have that ${}^1P \geq_W {}^1P \star S$. Hence, ${}^1P \star R \geq_W P$, so by Corollary 3.2.3 we conclude that $R \geq_W {}^2P$, as desired. □

Unfortunately, there is no reason to expect that $P \equiv_W {}^1P \star {}^2P$, leaving us one step away the possibility of a true decomposition in the Weihrauch degrees. Instead, we get the following.

Theorem 3.2.5. *Let P be a problem. Then, P decomposes into first order part ${}^1P$ and compositional second order part ${}^2P$ such that ${}^1P \star {}^2P \geq_W P$ and such that for each R, if ${}^1P \star R \geq_W P$ then $R \geq_W {}^2P$.*

This is a preliminary result in a new research direction. I conjecture that the situation is better in the degree structure obtained by fixing instances of P. We leave this to future work.

Additionally, theorem 3.2.2 suggests the ability to to stratify the problems Q such that such that there exists a uniform forwards functional Φ witnessing $Q \leq_c P$. It remains to check whether there is a correspondence between such Q and first order R such that $Q = P \star^{-1}_{u,\ell} R$. For fixed Q, such an R would have the property that $R = P \star^{-1}_{\ell,r} R$. We also leave this and other consequences to the algebraic structure of the Weihrauch degrees to future work.

Bibliography

[1] Dimitris Achlioptas and Themis Gouleakis. "Algorithmic improvements of the Lovász Local Lemma via cluster expansion." In: *Leibniz International Proceedings in Informatics, LIPIcs*. Vol. 18. 2012, pp. 16–23. ISBN: 9783939897477. DOI: 10.4230/LIPIcs.FSTTCS.2012.16.

[2] Noga Alon. "A parallel algorithmic version of the local lemma." In: *Random Structures & Algorithms* 2 (4 Dec. 1991), pp. 367–378. ISSN: 10429832. DOI: 10.1002/rsa.3240020403.

[3] Noga Alon, Joel Spencer, and Paul Erdős. *The Probabilistic Method*. John Wiley & Sons, 1992. ISBN: 0-471-53588-5.

[4] József Beck. "An algorithmic approach to the Lovász local lemma. I." In: *Random Structures & Algorithms* 2 (4 Dec. 1991), pp. 343–365. ISSN: 10429832. DOI: 10.1002/rsa.3240020402.

[5] József Beck. "An Application of Lovasz Local Lemma: There Exists an Infinite 01-Sequence Containing No Near Identical Intervals." In: *Finite and Infinite Sets* 37 (1984), pp. 103–107. DOI: 10.1016/b978-0-444-86893-0.50011-5.

[6] Vasco Brattka, Guido Gherardi, and Arno Pauly. *Weihrauch Complexity in Computable Analysis*. Ed. by Vasco Brattka and Peter Hertling. 2021. DOI: 10.1007/978-3-030-59234-9_11. URL: https://doi.org/10.1007/978-3-030-59234-9_11.

[7] Douglas Cenzer, Valentina Harizanov, and Jeffrey Remmel. "Computability-Theoretic Properties of Injection Structures." In: *Algebra and Logic* 53 (Sept. 2014), pp. 39–69. DOI: 10.1007/s10469-014-9270-0.

[8] Barbara Csima, Damir Dzhafarov, Denis Hirschfeldt, Carl Jockusch, Reed Solomon, and Linda Westrick. "The reverse mathematics of Hindman's Theorem for sums of exactly two elements." In: *Computability* 8 (3-4 2019), pp. 253–263. ISSN: 22113576. DOI: 10.3233/COM-180094.

[9] Artur Czumaj and Christian Scheideler. "Coloring Nonuniform Hypergraphs: A New Algorithmic Approach to the General Lovász Local Lemma." In: *Random Structures & Algorithms* 17 (3-4 2000), pp. 213–237. DOI: https://dl.acm.org/doi/10.5555/360708.360729.

[10] Rumen Dimitrov. "Cohesive Powers of Computable Structures." In: *Godishnik na Sofiskiya Universitet "Sv. Kliment Ohridski". Fakultet po Matematika i Informatika. Annuaire de l'Universite de So a "St. Kliment Ohridski". Faculte de Mathematiques et Informatique* 99 (2009), pp. 193–201.

[11] Rumen Dimitrov, Valentina Harizanov, Andrey Morozov, Paul Shafer, Alexandra Soskova, and Stefan Vatev. "Cohesive Powers of Linear Orders." In: *Computing with Foresight and Industry*. Ed. by Florin Manea, Barnaby Martin, Daniël Paulusma, and Giuseppe Primiero. Springer International Publishing, 2019, pp. 168–180. ISBN: 978-3-030-22996-2.

[12] Rumen Dimitrov, Valentina Harizanov, Andrey Morozov, Paul Shafer, Alexandra A Soskova, and Stefan V Vatev. *On cohesive powers of linear orders*. 2022. DOI: 10.48550/ARXIV.2009.00340. URL: https://arxiv.org/abs/2009.00340.

[13] Damir Dzhafarov, Jun Le Goh, Denis Hirschfeldt, Ludovic Patey, and Arno Pauly. "Ramsey's theorem and products in the Weihrauch degrees." In: *Computability* 9 (2020), pp. 85–110. ISSN: 2211-3576. DOI: 10.3233/COM-180203.

[14] Damir Dzhafarov and Carl Mummert. *Reverse Mathematics*. Springer International Publishing, 2022. ISBN: 978-3-031-11366-6. DOI: 10.1007/978-3-031-11367-3.

[15] Damir D Dzhafarov, Reed Solomon, and Keita Yokoyama. "On the First-Order Parts of Problems in the Weihrauch Degrees." In: (2023).

[16] P. Erdős and L. Lovász. "Problems and results on 3-chromatic hypergraphs and some related questions." In: *Infinite and finite sets* 2 (2 1975), pp. 609–627.

[17] Nicholas J.A. Harvey and Jan Vondrak. "An Algorithmic Proof of the Lovasz Local Lemma via Resampling oracles." In: *SIAM Journal on Computing* 49 (2 2020), pp. 394–428. ISSN: 10957111. DOI: 10.1137/18M1167176.

[18] Kun He, Qian Li, and Xiaoming Sun. "Moser–Tardos Algorithm: Beyond Shearer's Bound." In: *Proceedings* (Jan. 2023), pp. 3362–3387. DOI: 10.1137/1.9781611977554.CH129. URL: https://epubs.siam.org/doi/10.1137/1.9781611977554.ch129.

[19] Dennis Hirschfeldt and Sarah Reitzes. "Thin Set Versions of Hindman's Theorem." In: *Notre Dame Journal of Formal Logic* (to appear).

[20] Kashyap Kolipaka, Babu Rao, and Mario Szegedy. "Moser and Tardos Meet Lovász." In: *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery, 2011, pp. 235–244. ISBN: 9781450306911. DOI: 10.1145/1993636.1993669. URL: https://doi.org/10.1145/1993636.1993669.

[21] Kashyap Kolipaka, Mario Szegedy, and Yixin Xu. "A Sharper Local Lemma with Improved Applications." In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Ed. by Klaus, Rolim José, Servedio Rocco Gupta Anupam, and Jansen. Springer Berlin Heidelberg, 2012, pp. 603–614. ISBN: 978-3-642-32512-0.

[22] Lu Liu, Benoit Monin, and Ludovic Patey. "A computable analysis of variable words theorems." In: *Proceedings of the American Mathematical Society* 147 (2 2018), pp. 823–834. ISSN: 0002-9939. DOI: 10.1090/proc/14269.

[23] Michael Molloy and Bruce Reed. "Further algorithmic aspects of the local lemma." In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*. ACM Press, 1998, pp. 524–529. ISBN: 0897919629. DOI: 10.1145/276698.276866.

[24] Robin A. Moser and Gábor Tardos. "A Constructive Proof of the General Lovász Local Lemma." In: *Journal of the ACM* 57 (2 2010), pp. 1–12. ISSN: 00045411. DOI: 10.1145/1667053.1667060.

[25] Wesley Pegden. "An Extension of the Moser-Tardos Algorithmic Local Lemma." In: *SIAM Journal on Discrete Mathematics* 28 (2 2014), pp. 911–917. ISSN: 08954801. DOI: 10.1137/110828290.

[26] Wesley Pegden. "Highly nonrepetitive sequences: Winning strategies from the local lemma." In: *Random Structures & Algorithms* 38 (1-2 2011), pp. 140–161. DOI: https://doi.org/10.1002/rsa.20354. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rsa.20354.

[27] Andrei Rumyantsev and Alexander Shen. "Probabilistic constructions of computable objects and a computable version of Lovász local lemma." In: *Fundamenta Informaticae* 132 (1 May 2014), pp. 1–14. ISSN: 01692968. DOI: 10.3233/FI-2014-1029. URL: http://arxiv.org/abs/1305.1535.

[28] Aravind Srinivasan. "Improved Algorithmic Versions of the Lovász Local Lemma." In: *Proceedings of the nineteenth annual ACM –SIAM symposium on Discrete algorithms* (Jan. 2008), pp. 611–620.