Electronic Colloquium on Computational Complexity, Report No. 73 (2011)

Efficient Probabilistically Checkable Debates

Andrew Drucker*

Abstract

Probabilistically checkable debate systems (PCDSs) are debates between two competing provers, in which a polynomial-time verifier inspects a constant number of bits of the debate. It was shown by Condon, Feigenbaum, Lund, and Shor that every language in PSPACE has a PCDS in which the debate length is polynomially bounded. Using this result, they showed that the approximation versions of some natural PSPACE-complete problems are also PSPACE-complete.

We give an improved construction of these debates: for any language L that has an ordinary debate system definable by uniform circuits of size $s = s(n)$, we give a PCDS for L whose debate is of total bitlength $\widetilde{O}(s)$, with a verifier that uses only $\log_2 s + \log_2(\text{polylog}(s))$ bits of randomness. This yields a much tighter connection between the time complexity of natural PSPACE-complete problems and the time complexity of their approximation versions.

Our key ingredient is a novel application of error-resilient communication protocols, as developed by Schulman; we use the more recent protocol of Braverman and Rao. We show that by requiring ordinary debates to be encoded in an error-resilient fashion, we can endow them with a useful "stability" property. Stable debates can then be transformed into PCDSs, by applying efficient PCPPs (as given by Dinur). Our main technical challenge in building stable debates is to enforce error-resilient encoding by the debaters. To achieve this, we show that there is a constant-round debate system, with a very efficient verifier, to debate whether a communication transcript follows the Braverman-Rao protocol.

1 Introduction

1.1 Debate Systems

For many years, *debate systems* have played an important role in the study of complexity classes (e.g., in [CKS81, LFKN90, Sha92, BFL90, CFLS95]). A debate system is an interaction between two competing, computationally-unbounded *debaters*, supervised by a computationally-bounded *verifier*. Debate systems are often equivalently described in terms of *alternation* or *alternating nondeterminism* [CKS81].

In a debate system for a language L, an input $x \in \{0,1\}^n$ is given; the first debater (Player 1) tries to convince the verifier that $x \in L$, while the second debater (Player 0)

*CSAIL, MIT. Email: adrucker@mit.edu. Supported by a DARPA YFA grant.

ISSN 1433-8092

tries to convince the verifier that $x \notin L$. To this end, the debaters supply a sequence of strings $y = (y^1, \ldots, y^{k(n)})$. Here, each y^i is of a prespecified length depending only on n; y^i is supplied by a prespecified debater $type(i) \in \{0,1\}$, and is allowed to depend on $y^1, \ldots, y^{i-1}$. (We assume that $k(n), type(i)$, and the lengths $|y_i|$ are all computable in time $\text{poly}(n)$.)

Finally, the verifier applies a deterministic predicate $V(x,y)$ to determine an output $b \in \{0,1\}$. We say V defines a *debate system for* L if Player 1 can force $b = 1$ exactly when $x \in L$. If moreover $|y| \leq \text{poly}(n)$ and V is a polynomial-time algorithm, we say that V defines a *polynomial-time debate system for* L.

An important parameter is $k = k(n)$, the number of turns in the debate. If $k \geq 1$ is a constant, L has a k-round polynomial-time debate system if and only if L lies in the k^{th} level of the Polynomial Hierarchy (i.e., $\Sigma_k^p \cup \Pi_k^p$), as was essentially shown by Stockmeyer [Sto76] and Wrathall [Wra76]. Chandra and Stockmeyer [CS76] (see also [CKS81]) showed that when k is allowed to grow polynomially in the input length, polynomial-time debate systems characterize PSPACE:

Theorem 1. *[CS76, CKS81] A language L has a polynomial-time debate system if and only if $L \in$ PSPACE.*

Later Shamir [Sha92], building on [LFKN90], showed that every language $L \in$ PSPACE has an *interactive proof*—a polynomial-time debate system in which Player 0 plays completely random strings. If $x \in L$, then some Player 1 strategy causes the verifier to accept with probability 1, while if $x \notin L$, any Player 1 strategy causes the verifier to accept with probability at most $1/2$.

These results, as well as debate characterizations of other complexity classes, have been instrumental in determining the complexity of many natural computational problems—particularly 2-player games, which are readily expressed as debates. See [For05] for a fuller discussion of the importance of debate systems in complexity theory.

1.2 Probabilistically checkable debates

One of the most significant and surprising discoveries about debate systems is that they retain essentially all of their computational power under severe restrictions on the verifier V.

This discovery has its roots in the study of *probabilistically checkable proofs (PCPs)* for NP languages. We say that a randomized polynomial-time algorithm $V(x,y)$ is an $[r(n), q(n)]$-*restricted probabilistically checkable proof system* for the language L (with input $x \in \{0,1\}^n$ and proof string y of length $\text{poly}(n)$), if:

1. For all $x \in L$ there is a y with $\Pr[V(x,y) \text{ accepts}] = 1$;
2. For all $x \notin L$ and all y, $\Pr[V(x,y) \text{ accepts}] < 1 - \Omega(1)$;
3. V uses $r(n)$ bits of randomness and nonadaptively queries at most $q(n)$ bits of y.

The famous PCP Theorem of [ALM+98] states that we can provide an $[O(\log n), O(1)]$-restricted probabilistically checkable proof system for any $L \in$ NP. That is, there exists a special proof format that allows one to efficiently verify membership claims for L, while only looking at a constant number of bits of the proof!

An NP verifier for a language $L \in \mathsf{NP}$ can be viewed in the debate framework, as a debate system consisting of a single turn by Player 1. So, single-turn debates can be made probabilistically checkable with $O(1)$ queries, and it is natural to wonder whether the same can be done for more general classes of debates. To formalize this, let $V(x, y)$ be a polynomial-time verifier, where as before $y = (y^1, \ldots, y^k)$ are supplied by the competing debaters (we assume $|y| \leq \text{poly}(n)$). Now, however, V uses randomness. We call V an $[r(n), q(n)]$-*restricted probabilistically checkable debate system (PCDS)* for the language L if:

1. For all $x \in L$, there is a Player 1 strategy that forces $\Pr[V(x, y^1, \ldots, y^k) \text{ accepts}] = 1$ (note, we insist on *perfect completeness*);

2. For all $x \notin L$, there is a Player 0 strategy that forces $\Pr[V(x, y^1, \ldots, y^k) \text{ accepts}] < 1 - \Omega(1)$;

3. V uses $r(n)$ bits of randomness and nonadaptively queries at most $q(n)$ bits of y.

How powerful are PCDSs when we allow k to grow polynomially in n, but require $q(n) = O(1)$? Intuitively, this restriction seems quite severe, since only a constant number of the debate strings $y^1, \ldots, y^k$ will receive any queries at all. However, this intuition is deceptive: in [CFLS95], Condon, Feigenbaum, Lund, and Shor proved that PCDSs are essentially as strong as arbitrary polynomial-time debate systems:

Theorem 2. *[CFLS95] Every $L \in \mathsf{PSPACE}$ has an $[O(\log n), O(1)]$-restricted PCDS.*

Their proof used the PCP Theorem as a building block, along with several interesting new ideas. This result was complemented by several other works that studied various classes of debates; in each case it was shown that restricting the verifier to make $O(1)$ queries to the debate string does not reduce the power of the debates in question. Ko and Lin [KL94] showed that for any $k \geq 1$, if L is in $\Sigma_k^p \cup \Pi_k^p$ (the k^{th} level of the Polynomial Hierarchy), then there is a k-round PCDS either for L or for $\overline{L}$. Condon, Feigenbaum, Lund, and Shor [CFLS97], in a follow-up to their original work, showed that the interactive proofs in Shamir's result can be made probabilistically checkable; in their PCDSs, the verifier looks at only a constant number of the random bits written by Player 0 as well as a constant number of Player 1's bits. A corresponding result for AM protocols was shown in [Dru10].

1.3 The inapproximability connection

In addition to their inherent interest, probabilistically checkable debates also have a close connection to the complexity of approximation problems. The PCP Theorem (along with its many subsequent refinements) was the key to proving most of the known NP-hardness-of-approximation results for optimization problems whose decision versions lie in NP. Similarly, Theorem 2 implies that a number of reasonably natural computational problems lying in PSPACE are in fact PSPACE-hard to approximate, as was shown in [CFLS95]. As one simple example, suppose we are given a 3-CNF formula $\psi(x_1, \ldots, x_n)$, and we consider the game in which two players take turns in assigning values to the variables, with x_i assigned on the i^{th} round. Player 1 wants to maximize the fraction of satisfied clauses, while Player

0 wants to minimize this fraction. Let $\mathrm{Val}(\mathcal{G}_\psi) \in [0,1]$ denote the value of this game to Player 1. Using Theorem 1, one can show that it is PSPACE-complete to decide whether $\mathrm{Val}(\mathcal{G}_\psi) = 1$. However, from Theorem 2, the authors of [CFLS95] showed that for some constant $\varepsilon > 0$, it is PSPACE-hard even to distinguish formulas with $\mathrm{Val}(\mathcal{G}_\psi) = 1$ from formulas with $\mathrm{Val}(\mathcal{G}_\psi) < 1 - \varepsilon$.[1]

While this is a remarkable result, it is not completely satisfactory because the reduction involved, though polynomial-time, causes a large polynomial blowup of the 3-CNF instance sizes.[2] This blowup leads to somewhat weak conditional hardness results. Assume that any algorithm to correctly decide whether $\mathrm{Val}(\mathcal{G}_\psi) = 1$ for a 3-CNF ψ of description length n, requires time $\geq T(n)$ infinitely often ("i.o."), for some time bound $T(n) = n^{\omega(1)}$. Then, Theorem 2 implies that for small enough $\varepsilon > 0$, any algorithm achieving an ε-approximation to $\mathrm{Val}(\mathcal{G}_\psi)$ has a worst-case runtime $T(n^a)$ on length-n instances, for some explicit absolute constant $a < 1$.

We note that the reduction in the original PCP Theorem also incurred a similar blowup in parameters. However, in recent work Dinur [Din07, Thm. 8.1] gave a $[\log_2 n + \log_2(\mathrm{polylog}(n)), O(1)]$-probabilistically checkable proof system for length-n SAT instances. This yields much tighter conditional hardness statements for SAT: from an i.o.-lower bound $T(n) = n^{\omega(1)}$ on the runtime of algorithms for SAT, we get an i.o.-lower bound of $T(n/\log^c(n))$ on the runtime of algorithms to ε-approximate the maximum fraction of satisfiable clauses in a 3-CNF of description length n (for explicit constants $c, \varepsilon > 0$).

1.4 Our result

Our main result is the following quantitative strengthening of Theorem 2, which shows that polynomial-time debates can be made probabilistically checkable in a randomness-efficient way, with a debate string whose size is nearly-linear in the circuit complexity of the original verifier:

Theorem 3 (Main)**.** *Suppose L has a polynomial-time debate system with a verifier implementable by polynomial-time-constructible Boolean circuits of size $s = s(n) \leq \mathrm{poly}(n)$. Then L has a $[\log_2 s + \log_2(\mathrm{polylog}(s)), O(1)]$-restricted PCDS, with a debate string of total bitlength $\widetilde{O}(s)$.*

Many natural PSPACE-complete problems, like QBF-SAT (the set of true quantified Boolean formulas, under any standard encoding), have ordinary debate systems with circuits of size $s(n) = \widetilde{O}(n)$, and for such problems Theorem 3 yields a PCDS with debate string bitlength $\widetilde{O}(n)$ and randomness $\log_2(n) + \log_2(\mathrm{polylog}(n))$.[3] Then using Theorem 3, an i.o.-lower bound $T(n) = n^{\omega(1)}$ for determining whether $\mathrm{Val}(\mathcal{G}_\psi) = 1$ for 3-CNFs ψ allows

[1] The technique is the same as that used to derive inapproximability results for Maximum Satisfiability from PCP constructions.

[2] [CFLS95] does not provide an explicit polynomial bound. The bound arising in their work can be brought down somewhat by substituting state-of-the-art PCPs from [Din07] into their applications of PCPs. However, it appears that following their basic approach to leads to at least a cubic blowup.

[3] Actually, for QBF-SAT it is not hard to achieve $s(n) = O(n)$; see [Wil08, p. 1].

us to infer an i.o.-lower bound of $T(n/\log^c(n))$ on the runtime of algorithms to ε-approximate $\mathrm{Val}(\mathcal{G}_\psi)$, for explicit constants $c, \varepsilon > 0$.

We also mention that any language solvable in space $S(n) \leq \mathrm{poly}(n)$ has a polynomial-time debate system definable by uniform circuits of size $s(n) = \widetilde{O}(S(n)^2)$ [SM73, CKS81]. Thus, for such languages we get a PCDS with debate bitlength $\widetilde{O}(S(n)^2)$.

The PCDS construction in Theorem 3 has close to the best randomness-efficiency we can hope to achieve (for general L and s), barring a huge algorithmic breakthrough: if the randomness complexity could be brought down to $(1 - \Omega(1))\log_2(s)$, we could use the resulting PCDS for QBF-SAT to solve length-n QBF-SAT instances in time $2^{n^{1-\Omega(1)}}$. Now, it is also natural to wonder whether the debate string in the PCDS of Theorem 3 could always be made to have bitlength bounded in terms of $\ell = \ell(n)$, the bitlength of the ordinary debate-string for L (which may be much smaller than s). Any bound of form $\mathrm{poly}(\ell)$ would be very interesting. However, this seems unlikely. In the theory of probabilistically checkable proofs, the corresponding question is whether satisfiability of q-variable SAT instances can be proved with a PCP of proof-length $\mathrm{poly}(q)$; Fortnow and Santhanam showed that this cannot be done unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ [FS11].

1.5 Our techniques

To prove Theorem 3, we give a new method for converting standard debate systems into probabilistically checkable ones. The method has two steps. In the first (and more novel) step, we transform a standard debate into one that has a useful "stability" property; in the second step, we transform a stable debate into a probabilistically checkable debate.

1.5.1 Stable debates via error-resilient communication

We say a debate system $V(x, \cdot)$ for a language L is *stable* if for any $x \notin L$, Player 0 can not only force $V(x, y) = 0$, but can even force the debate string y to be $\Omega(1)$-far in relative Hamming distance from any y' for which $V(x, y') = 1$. (Thus, our stability notion is asymmetric with respect to the players.) The notion of stability was used before, implicitly or explicitly, in several ways in [CFLS95, CFLS97, Dru10]. However, in the previous works building many-round PCDSs [CFLS95, CFLS97], debates are endowed with a stability-like property[4] in a fairly inefficient way: roughly speaking, on each turn of the debate, the current player is asked to give a description of all previous moves along with their current move. This contributes a quadratic blowup in the overall debate-string length (in addition to blowups at other steps in the transformation).

In our transformation yielding a stable debate, we manage to avoid using such redundancy. We do so by drawing a new connection to *interactive coding*—the theory of error-resilient two-way communication. Interactive coding was pioneered by Schulman [Sch96],

[4]These papers' first step is to transform an ordinary debate system into one in which a probabilistic verifier inspects only a constant number of the individual player *moves* (y^i), which may be of polynomial length. Such debates play a role somewhat analogous to the role played by stable debates in our work; the analogy is loose, however, and stable debates turn out to be more easily and efficiently turned into PCDSs.

who showed that any two-party communication protocol can be converted into one that succeeds even in the presence of a noticeable amount of adversarial noise (an adversary that may adaptively corrupt a 1/240 fraction of the bits sent between the two parties). Moreover, this conversion increases the total communication by only a constant factor. Schulman's powerful result seems not to be obtainable from standard tools for resilient one-way communication (i.e., error-correcting codes).

Recently, Braverman and Rao [BR10] gave a new encoding method to achieve the same goal. Their encoding corrects from a much larger fraction of adversarially corrupted symbols—nearly 1/8 if bits are transmitted, or nearly 1/4 if a larger but constant-sized message alphabet is used. More importantly for us, their encoding is somewhat simpler and easier to work with. (We do not know whether the protocol of [Sch96] could also be used to prove our result.)

In our application of interactive coding, we begin with an ordinary debate system for a language L, defined by a verifier V implemented by uniform circuits of size $s(n)$. We then transform V into a second verifier V^{stab}, in which the two debaters are "forced" to encode their debate using the Braverman-Rao encoding. We show that the error-resilience property of the encoding can be used to ensure the stability property of V^{stab}.

But how can we force the debaters to follow the desired encoding? To do this, we construct an auxiliary *"encoding-checker"* debate system, that allows players to debate whether a communication transcript corresponds to a faithful, noise-free execution of the Braverman-Rao protocol.[5] The encoding-checker debate we construct has two important properties. First, it lasts for only $O(1)$ turns. This is important because an $O(1)$-turn debate can fairly easily be made *stable*, by asking for the moves to be encoded in an error-correcting code. With this stable encoding-checker debate in hand, we can make the entire debate stable. (Conceptually, this is the right picture; technically, we make the entire debate stable in one step, rather than first making the auxiliary debate stable.)

The second important property of our encoding-checker debate is that it has a very efficient verifier—one that is definable by a Boolean circuit of size $O(\ell)$, where ℓ is the bitlength of the communication transcript being checked. As a result, our stable verifier V^{stab} can be implemented by a circuit of size $\widetilde{O}(s(n))$ for length-n inputs. This near-linear efficiency is important in the second step of our transformation, in which we make the debate probabilistically checkable.

Achieving these two strong properties in our encoding-checker debate is our main technical challenge. We will return to the issue of how this challenge can be met.

1.5.2 From stable to probabilistically checkable debates

In our second transformation step, we extend our stable debate $y = (y^1, \ldots, y^k)$ by a single, final turn, in which Player 1 gives a "proof" string z purporting to show that $V_x^{stab}(y) := V^{stab}(x, y) = 1$. We then define a verifier V^* that, given x, probabilistically checks $O(1)$ bits of y and z. For this proof/verification task, we use a powerful variant of PCPs known

[5] More precisely, they can debate whether a *particular* player is following the Braverman-Rao protocol. Checking one player's behavior turns out to be sufficient; the other player can be indirectly "incentivized" to follow the protocol. This is important for achieving perfect completeness in our PCDSs.

as *probabilistically checkable proofs of proximity (PCPPs)* [BSGH+06, DR06], applied to the circuit for the stable verifier V_x^{stab}. We mention that PCPPs were put to a similar use in [Dru10], and closely related techniques were used in [CFLS95, CFLS97].

If $x \in L$, then Player 1 is able to win the stable debate $(y^1, \ldots, y^k)$, so that $V_x^{stab}(y) = 1$, and then some proof z causes V^* to accept with certainty. On the other hand, if $x \notin L$, then by stability, Player 0 can guarantee that y is not even *close* to the set of debate strings for which $V_x^{stab} = 1$. This is precisely the condition in the definition of PCPPs that guarantees that V^* will reject with noticeable probability for any setting to z. Thus the probabilistic verifier V^* defines our desired PCDS for L.

How efficient is this verifier? The length of the final proof-string z and the randomness complexity of V^* are determined by two factors: the efficiency of the PCPP construction we use, and the size of the circuit for V_x^{stab} to which we apply the PCPP. We are fortunate in both respects. A very efficient construction of PCPPs is available, due to Dinur [Din07] (building on [BSS06]); also, our efficient construction of stable debates ensures that the circuit for V_x^{stab} is of size $\widetilde{O}(s(n))$. This yields a verifier and debate-string with the properties claimed in Theorem 3.

1.5.3 Building encoding-checker debates

Recall that our approach requires us to build "encoding-checker" debates, to check that a given party's (Alice's) behavior on a communication transcript corresponds to a correct execution of the Braverman-Rao protocol [BR10] for error-resilient communication.

A first complication is that the Braverman-Rao protocol is not even known to have an implementation in *polynomial* time, let alone *nearly-linear* time. Like Schulman's earlier protocol [Sch96], the protocol crucially relies on a special type of codes called *tree codes*, defined by Schulman. Tree codes have a "distance" parameter, that is loosely analogous to the minimum-distance parameter for error-correcting codes. No explicit construction is known of tree codes with distance large enough to run the protocols of [BR10, Sch96] (see [Bra11] for a recent subexponential-time construction). However, in [Sch96] an elegant probabilistic construction of tree codes was given. Importantly for us, this construction is very *randomness-efficient*, so that good tree codes exist with succinct representations. This is enough for our purposes: in our debate system, a computationally unbounded debater may succinctly propose a tree code, and establish *through debate* that it has the needed distance property.

The protocols of [Sch96, BR10] require the communicating parties to decode corrupted tree-code-encoded messages. In the model in which communication is corrupted adversarially (the relevant model for us), it is not known whether this decoding can be performed in polynomial time. However, we are again able to use the power of computationally unbounded debaters, this time to debate the correct values of tree-code encodings and decodings.

We mention in passing that tree codes are not known to be *necessary* for interactive coding. Indeed, in very recent papers by Moitra [Moi11] and, independently, Gelles and Sahai [GS11], it was shown that the tree-code definition can be *relaxed* to a weaker but still-useful kind of object that is easier to construct, yielding interactive coding schemes that are computationally efficient in the *random channel-noise* model. The papers use dif-

ferent relaxations and achieve similar but incomparable results. Achieving efficiency in the adversarial-noise model remains an important open question.

Now in the Braverman-Rao protocol, correct behavior for each player is defined relative to some *input* to the communication task. In our application, the intended input is a player-strategy in the original debate V for L (Alice is allowed to choose her input/strategy). Such a strategy is an object of exponential size, and cannot even be written down in the encoding-checker debate. Fortunately, any particular execution of the Braverman-Rao protocol only depends on a much smaller portion of the strategy. This crucially helps us, although naively encoding the relevant portion is not succinct enough for our purposes.

The player Alice, while executing the Braverman-Rao protocol, maintains data, and this data needs to be represented in the encoding-checker debate. We are fortunate that the data takes the simple form of a sequence $(a_1, a_2, \ldots)$ of symbols over a constant-size alphabet, with a_i being defined on the i^{th} round of communication and never modified. This allows a debater to succinctly present a global description of Alice's execution. However, a_i is defined in a complex way from previous values and from the messages received from Bob. To understand how to efficiently debate the proper settings to a_i, we will make a detailed study of a method used by the Braverman-Rao protocol to succinctly describe subsets of edges in a complete binary tree. Our encoding-checker debate system will be built up from a sequence of simpler debates used to reason about this description method and its use in the protocol.

1.6 Organization of the paper

Section 2 gives preliminaries that will be used throughout the paper. In Section 3, we show how Theorem 3 follows from an efficient construction of stable debate systems from ordinary ones. In Section 4, we present the necessary background about communication protocols and error-resilient communication, and we define the Braverman-Rao protocol we use.

The next two sections are the heart of the paper. In Section 5, we state our lemma on "encoding-checker" debates for the Braverman-Rao protocol, and use it to build stable debates. In Section 6 we build the needed encoding-checker debates. Finally, in Section 7 we conclude with some open problems.

2 Preliminaries

Given $k \geq 1$, an alphabet Σ, and $u, v \in \Sigma^k$, let $\Delta(u, v) := \{i \in [k] : u_i \neq v_i\}$ denote the generalized Hamming distance between u and v. We say that u is d-far from a set $S \subseteq \Sigma^k$ if $\Delta(u, v) \geq d$ for all $v \in S$.

Throughout the paper, when we refer to "uniform" circuit families $\{C_n\}_{n>0}$, we mean "polynomial-time constructible": there is an algorithm A that, on input 1^n, outputs a description of C_n in time $\mathrm{poly}(n)$.

2.1 Constraint Satisfaction Problems and Debate CSPs

For $k \geq 1$, a *k-local Constraint Satisfaction Problem*, or *k-CSP*, over finite alphabet Σ is a collection $\psi(x) = (\psi_1(x), \ldots \psi_m(x))$ of Boolean-valued functions ("constraints") on the input $x = (x_1, \ldots x_n) \in \Sigma^n$, where each ψ_j depends only on some k variables of x. Formally, ψ_j is specified by a k-tuple $I_j \subseteq [n]$ and a truth-table on these k variables. For an assignment x, define $\mathrm{Val}_\psi(x)$, the *value of ψ on x*, as the fraction of constraints ψ_j which are "satisfied", i.e., for which $\psi_j(x) = 1$.

We use CSPs to define 2-player *debates*, as follows. A *t-turn debate CSP* is a CSP $\psi(y)$, whose variables are partitioned into t designated *debate-blocks*, $y = (y^1, \ldots, y^t)$. The blocks y^j may be of unequal size, and each debate-block is designated a "0-block" or a "1-block"; we let $type(j) \in \{0, 1\}$ indicate the type of block y^j. (These types need not be strictly alternating.) Note that if $k = O(1)$ and $\psi(y)$ is a debate k-CSP of with m constraints, a natural description of ψ containing its block-information can be given using $n = O(m \log m)$ bits, similar to the description length of an ordinary k-CSP. We abuse notation and let "ψ" refer to the formula ψ along with its block-information.

To a debate CSP ψ we associate a turn-based, perfect-information game $\mathcal{G}_\psi$ between two players, Player 0 and Player 1. In this game, Player 1 tries to maximize the fraction of satisfied constraints, while Player 0 tries to minimize this fraction. On the j^{th} turn, Player $type(j)$ assigns values to the variables in y^j. We let $\mathrm{Val}(\mathcal{G}_\psi) \in [0, 1]$ denote the fraction of satisfied constraints under optimal play by both players. (This generalizes our definition of $\mathrm{Val}(\mathcal{G}_\psi)$ for 3-CNFs in the Introduction.)

2.2 Debate circuits

One can use circuits to define 2-player debates, similarly to the way we defined debates on CSPs. A *t-turn debate circuit* is a bounded-fanin Boolean circuit $\Phi(x, y)$ with two types of inputs: "start-variables" x, and "debate-variables" y. The debate-variables are partitioned into t designated *debate-blocks*, $y = (y^1, \ldots, y^t)$. We let $type(j) \in \{0, 1\}$ indicate the type of block y^j. If $\Phi(x)$ is a debate circuit of size m, a natural description of Φ containing its block information can be given using $O(m \log m)$ bits, similar to the description length of an ordinary circuit.

To a debate circuit Φ and any assignment x to the start-variables, we associate a game $\mathcal{G}_\Phi[x]$ between two players, Player 0 and Player 1. On the j^{th} turn, Player $type(j)$ assigns values to the variables in y^j. After t turns have been played and all debate-variables have been assigned, Player 1 wins the game if $\Phi(x, y) = 1$, otherwise Player 0 wins. We let $\mathrm{Val}(\mathcal{G}_\Phi[x]) \in \{0, 1\}$ denote the winner of this game under optimal play by both players.

2.3 PCPPs and error-correcting codes

Following Dinur [Din07], we will actually give a CSP-based definition of PCPPs.[6] The probabilistic-verifier description of PCPPs (as used in the Introduction) is essentially equivalent to this one.

[6] In [Din07], PCPPs are referred to as *assignment testers*.

Say we are given a Boolean circuit $C(x)$ on N input bits, a finite alphabet Σ, and a value $\beta > 0$. We assume that $\{0,1\} \subseteq \Sigma$. We say that a k-CSP ψ acting on variables over Σ is a *PCPP for C over Σ with security β* if:

1. ψ has variable-set (x, z), where x are the Boolean input variables to C and z are so-called "proof"-variables, taking values over Σ;
2. If $C(x) = 1$ then there exists a setting of z such that $\mathrm{Val}_\psi(x, z) = 1$;
3. If $C(x) = 0$, then for all z, $\mathrm{Val}_\psi(x, z) \leq 1 - \beta \cdot \Delta(x, C^{-1}(1))/N$.

The *proof size* of ψ is the number of variables in z.

We will use the following efficient construction of PCPPs, due to Dinur:

Theorem 4. *[Din07, Cor. 8.4] There is a constant-size alphabet Σ_0, a constant $\beta > 0$, and a polynomial-time algorithm that, given a description of a circuit $C(x)$ of size t, produces a 2-CSP $\psi_C(x, z)$ that is a PCPP for C over Σ_0 with security β. Moreover, the proof size of ψ_C and the number of constraints in ψ_C are each at most $\widetilde{O}(t)$.*

Next we review error-correcting codes and the efficiently decodable codes we will use. A (binary) *code* is an injective map $Enc : \{0,1\}^N \to \{0,1\}^{N'}$ (so $N' \geq N$). The *minimum distance* of the code is the minimum over distinct codewords $u = Enc(x), v = Enc(y)$ of $\Delta(u, v)$. An algorithm $Dec : \{0,1\}^{N'} \to \{0,1\}^N$ *decodes Enc from an η fraction of errors* if, whenever $\Delta(w, Enc(x)) \leq \eta N'$, we have $Dec(w) = x$. Note that such for such a decoder algorithm to exist, the minimum distance of E must be greater than $2\eta N'$.

We will use a family of codes due to Spielman [Spi96] with highly efficient decoders:

Theorem 5. *[Spi96] There is an $\eta > 0$ and a family of codes $\{Enc_N\}_{N>0}$ for all input lengths N, with output length $N' = O(N)$, such that Enc_N can be decoded from an η fraction of errors by a uniform circuit Dec_N of size $O(N \log N)$. Also, Enc_N can be computed by uniform circuits of size $O(N)$.*

In Spielman's work, the "decoder algorithm" attempts to output an error-free *codeword*, rather than a pre-encoded message. Spielman shows how to efficiently recover any codeword that has been corrupted by an η fraction of errors. However, the codes he constructs are *systematic*, that is, every message $x \in \{0,1\}^N$ appears in a fixed location within its own codeword $Enc_N(x)$. Thus from his work, we easily obtain Theorem 5.

3 Constructing PCDSs from stable debates

In this section we show how to derive the PCDSs claimed in Theorem 3 for the language L, from efficient debate systems for L with the "stability" property described in the Introduction (formally defined in Section 3.2). Subsequent sections will be devoted to constructing these stable debates.

3.1 A CSP formulation of the main theorem

We will actually show that stable debates imply an alternative form of our main result, stated in terms of debate CSPs. We will show:

Theorem 6. *There is a finite alphabet Σ and an absolute constant $\varepsilon > 0$ such that the following is true. Let L be any language with a polynomial-time debate system whose verifier is implementable by uniform Boolean circuits of size $s = s(n) \leq \text{poly}(n)$. Then there is a polynomial-time reduction R that takes as input a string $x \in \{0,1\}^n$ and outputs a debate 2-CSP ψ with variables over Σ. We have:*

1. *If $x \in L$, then $\text{Val}(\mathcal{G}_\psi) = 1$;*
2. *if $x \notin L$, then $\text{Val}(\mathcal{G}_\psi) < 1 - \varepsilon$.*

Moreover, the number of constraints in ψ is at most $\widetilde{O}(s)$.

Theorem 3 follows readily from Theorem 6, by a standard technique:

Proof of Theorem 3. Define a PCDS for L as follows. Given an input $x \in \{0,1\}^n$, the verifier V^* applies the reduction R of Theorem 6 to x, yielding a 2-CSP debate $\psi(y)$ with debate-blocks $y = (y^1, \ldots, y^m)$. V^* expects the debaters to supply (binary encodings of) assignments to $y^1, y^2, \ldots$ in turn, where Player $type(j)$ chooses the assignment to y^j. V^* then chooses a uniformly random constraint ψ_i of ψ, queries the assignments to the two variables appearing in ψ_i, and outputs $\psi_i(y) \in \{0,1\}$.

Observe that under optimal play by the debaters, $\Pr[V^* \text{ outputs } 1] = \text{Val}(\mathcal{G}_\psi)$; this equals 1 if $x \in L$, and is less than $1 - \varepsilon$ otherwise. Also, V^* makes $O(\log_2 |\Sigma|) = O(1)$ queries to y. Finally, the number of random bits needed to select a random constraint of ψ is at most $\log_2 c = \log_2 s + \log_2(\text{polylog}(s))$, where $c = \widetilde{O}(s)$ is the number of constraints in ψ. Thus V^* is the desired $[\log_2 s + \log_2(\text{polylog}(s)), O(1)]$-restricted PCDS for L. □

3.2 Stability and PCDSs

Consider a (deterministic) algorithm $V(x, y)$ with block-information for y defining a polynomial-time debate system for some language L. Fixing $\delta > 0$, say that $V(x, y)$ is a *0-stable debate system with security δ* if for all inputs $x \notin L$, there is a Player 0 strategy such that for all Player 1 strategies, the resulting debate string y is at Hamming distance at least $\delta|y|$ from any y' for which $V(x, y') = 1$.

In later sections we will prove the following lemma, implementing "step 1" of the debate transformation described in Section 1.5 of the Introduction:

Lemma 7. *There is a $\delta > 0$ for which the following holds. Let L be any language with a polynomial-time debate system whose verifier is implementable by uniform Boolean circuits of size $s = s(n) \leq \text{poly}(n)$. Then there is a 0-stable polynomial-time debate system $V^{stab}(x, y)$ for L with security δ. Moreover, for inputs x of length n, $V^{stab}(x, y)$ is implementable by a uniform circuit $C_{V^{stab},n}$ of size $O(s \log s)$.*

The log-factor arises due to our use of Spielman's quasilinear-size decoder circuits (Theorem 5). If we had a family of binary codes with constant rate, decoded from an $\Omega(1)$ fraction of errors by a linear-size circuit family, this would bring the size of $C_{V^{stab},n}$ down to $O(s(n))$, and save a hidden log factor in Theorems 3 and 6. However, it seems to be unknown whether such codes exist.

We now derive Theorem 6 from Lemma 7. (This is "step 2" of our transformation.)

Proof of Theorem 6. Given an input x of length n, let $V^{stab}(\cdot,\cdot)$ be the 0-stable debate system for L as given by Lemma 7, and $C_{V^{stab},n}$ the uniform circuit of size $\widetilde{O}(s(n))$ implenting V^{stab} on length n. Define the specialized circuit $C_{[x]}(y) := C_{V^{stab},n}(x,y)$.

Let $\psi(y,z) := \psi_{C_{[x]}}(y,z)$ be the PCPP 2-CSP for $C_{[x]}$ given by Theorem 4, with alphabet Σ_0. We turn ψ into a debate CSP, by arranging its debate-blocks as follows:

1. First, the blocks of y are assigned Boolean values by the players, exactly as in the debate circuit $C_{V^{stab},n}(\cdot,\cdot)$;[7]

2. Second, Player 1 assigns values to z (from the full alphabet Σ_0).

We let ψ, with these debate-blocks, be the output of our reduction R. Note, ψ is constructible in polynomial time from x since $C_{V^{stab},n}$ is uniform, and ψ has a constant-size alphabet $\Sigma = \Sigma_0$ as needed.

We analyze R. First, suppose $x \in L$. Then as V^{stab} is a polynomial-time debate system for L, there exists a Player 1 debate-strategy forcing $C_{V^{stab},n}(x,y) = 1$; call this strategy $\mathcal{S}_1$. For any Player 0 strategy played against $\mathcal{S}_1$, the outcome $C_{[x]}(y) = C_{V^{stab},n}(x,y) = 1$ along with property 2 in the definition of PCPPs implies that there exists a z such that $\mathrm{Val}_\psi(y,z) = 1$. We conclude that $\mathrm{Val}(\mathcal{G}_\psi) = 1$.

Suppose next that $x \notin L$. Then by the 0-stability property of V^{stab}, there exists a Player 0 strategy $\mathcal{S}_0$ that forces y to be $\delta|y|$-far from any y' for which $C_{[x]}(y') = 1$. In this outcome, property 3 from the definition of PCPPs implies that for any z, $\mathrm{Val}_\psi(y,z) \le 1 - \beta\delta$. It follows that $\mathrm{Val}(\mathcal{G}_\psi) \le 1 - \beta\delta$ in this case.

Thus conditions 1 and 2 of Theorem 6 are met, if we set $\varepsilon < \beta\delta$. Finally, by the efficiency property of Theorem 4, the number of constraints in ψ is at most $\widetilde{O}\left(\left|C_{[x]}\right|\right) = \widetilde{O}\left(s(n)\right)$, as needed. This proves Theorem 6. □

4 Error-resilient communication

In this section we review the definitions of communication protocols and error-resilient communication. We will then carefully describe the Braverman-Rao protocol for error-resilient communication [BR10]. Combinatorial objects called *tree codes* are central to the Braverman-Rao protocol, and we review their definition as well.

[7] Technically, in the definition of debate CSPs we have no way of forcing the players to use Boolean values rather than general values from Σ_0. However, in constraints ψ_i of ψ in which variables from y appear, we can modify ψ_i to simply interpret all symbols in $\Sigma_0 \setminus \{0,1\}$ as representing 0s.

4.1 Communication protocols

First we formally define communication protocols. In our application, it is important that the communication protocols we study define the behavior of each party on each possible transcript of previously sent messages, even when these previous messages deviate from the protocol. We will call such protocols *total*.

In a communication problem, two parties, Alice and Bob, hold inputs x and y respectively from respective domains D_A, D_B. They wish to evaluate some function $f(x, y)$, which may not be Boolean-valued. To do so, the parties executive a *communication protocol*, defined next, with the goal that after the protocol's execution both parties know the value $f(x, y)$. We focus on protocols lasting for a predetermined number $T > 0$ of steps, with a shared, possibly non-Boolean alphabet Σ for the messages. On an odd-numbered step $i \leq T$, Alice sends a symbol $u_i \in \Sigma$ to Bob; on an even-numbered step i, Bob sends a symbol $w_i \in \Sigma$ to Alice.

Formally, we define a (deterministic) *T-bit communication protocol* $\mathcal{P}$ as a pair

$$\mathcal{P} = (\mathcal{P}_A, \mathcal{P}_B),$$

consisting of an *Alice-protocol* $\mathcal{P}_A$ and a *Bob-protocol* $\mathcal{P}_B$. An Alice-protocol is a mapping

$$\mathcal{P}_A : (D_A \times \Sigma^*) \to \Sigma \cup \{\perp\},$$

where $\perp$ means "undefined". We say that $\mathcal{P}_A$ is *total* if $\mathcal{P}_A(x, z) \in \Sigma$ for all $z \in \Sigma^{2i}$ with $0 \leq 2i < T$. Similarly, a Bob-protocol is a mapping $\mathcal{P}_B : (D_B \times \Sigma^*) \to \Sigma \cup \{\perp\}$, and we say that $\mathcal{P}_B$ is total if $\mathcal{P}_A(x, z) \in \Sigma$ for all $z \in \Sigma^{2i+1}$ with $0 \leq 2i + 1 < T$. We say $\mathcal{P}$ is total if $\mathcal{P}_A, \mathcal{P}_B$ are both total.

We let u_i, w_i denote the i^{th} messages of Alice and Bob respectively. The semantics on inputs x, y are as follows: to begin, Alice sends the symbol $u_1 := \mathcal{P}_A(x, \lambda)$ to Bob, where λ denotes the empty string. Bob replies with $w_1 := \mathcal{P}_B(y, u_1)$. On step $2i - 1 \in \{3, \ldots, T\}$, Alice sends the symbol $u_i := \mathcal{P}_A(x, u_1, \ldots, u_{i-1}, w_1, \ldots, w_{i-1})$; on step $2i \in \{2, \ldots, T\}$, Bob sends the symbol $w_i := \mathcal{P}_B(x, u_1, \ldots, u_i, w_1, \ldots, w_{i-1})$.

Alice and Bob each produce an output after the T^{th} step, which depends on their input and the communication transcript. Formally, Alice's output is given by an *evaluation function*

$$Eval_A : D_A \times \Sigma^T \to \text{range}(f)$$

applied to (x, u, w), where $(u, w) = (u_1, \ldots, u_{\lceil T/2 \rceil}, w_1, \ldots, w_{\lfloor T/2 \rfloor})$. Similarly, Bob's output is given by some $Eval_B : D_B \times \Sigma^T \to \text{range}(f)$, applied to (y, u, w).

Let $Eval := (Eval_A, Eval_B)$; we will freely use the term *protocol* to refer jointly to the pair $(\mathcal{P}, Eval)$. We say the protocol *computes* f if on any inputs x, y, the process above leads both Alice and Bob to output $f(x, y)$.

4.2 The pointer-jumping problem

The *pointer-jumping problem* is a fundamental problem in the study of communication. To describe the problem we need some setup. Fix a $T > 0$, and let $\mathcal{T}$ be the complete binary

tree of depth T. The tree $\mathcal{T}$ has edge-set $\mathcal{X} \cup \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ denote the edges at odd and even depths of $\mathcal{T}$ respectively. (The *depth* of an edge e is the length of the path from the root of $\mathcal{T}$ ending at the child vertex of e. So, the edges leaving the root are of depth 1, and lie in $\mathcal{X}$.) A subset $Z \subseteq \mathcal{X} \cup \mathcal{Y}$ is called *consistent*, if no two edges in Z share a common parent. For any consistent edge-set Z, there exists a unique vertex in $\mathcal{T}$ of maximal depth, reachable from the root by a path contained in Z; we let $v(Z)$ denote this vertex.

In the pointer-jumping problem, Alice receives as input a *maximal* consistent subset $X \subseteq \mathcal{X}$; that is, every vertex with outgoing edges in $\mathcal{X}$ has *exactly* one outgoing edge in X. Bob receives a maximal consistent subset $Y \subseteq \mathcal{Y}$. Observe that $X \cup Y$ is consistent. The *pointer-jumping function* (with parameter T) is defined as

$$\mathrm{PJ}_T(X, Y) := v(X \cup Y).$$

PJ_T is computable by a T-bit communication protocol. Also, it is easy to see that any function $f(x, y)$ computable by a T-bit communication protocol is reducible (in a natural sense) to PJ_T: there exist a pair of mappings m_A, m_B where $(X, Y) = (m_A(x), m_B(y))$ is a valid input to PJ_T, and such that the value $f(x, y)$ can be determined from $\mathrm{PJ}_T(X, Y)$.

4.3 Faithful behavior in communication protocols

Suppose that $(\mathcal{P}, Eval)$ is a $2T$-bit protocol to compute some function $f(x, y)$, using messages in alphabet Σ. Let $x \in D_A$ be an input to Alice. Say that the transcript $(u, w) \in \Sigma^{2T}$ is *$\mathcal{P}_A$-faithful for x* if $\mathcal{P}_A(x, \lambda) = u_1$ and, for each $i \in \{2, \ldots, T\}$, $\mathcal{P}_A(x, u_1, \ldots, u_{i-1}, w_1, \ldots, w_{i-1}) = u_i$. That is, Alice behaves according to $\mathcal{P}_A(x, \cdot)$ under the message-sequence w from Bob (even though Bob might not be following the protocol correctly). Say that the transcript (u, w) is *$\mathcal{P}_A$-faithful* if it is $\mathcal{P}_A$-faithful for *some* input x.

Similarly, given $y \in D_B$, say that (u, w) is *$\mathcal{P}_B$-faithful for y* if $\mathcal{P}_B(y, u_1) = w_1$ and, for each $i \in \{2, \ldots, T\}$, $\mathcal{P}_B(y, u_1, \ldots, u_i, w_1, \ldots, w_{i-1}) = w_i$. Say that (u, w) is *$\mathcal{P}_B$-faithful* if it is $\mathcal{P}_B$-faithful for some y. If the transcript (u, w) is both $\mathcal{P}_A$-faithful and $\mathcal{P}_B$-faithful, we call it a *perfect execution* of $\mathcal{P}$.

4.4 Error-resilient communication

We consider the scenario in which messages sent between Alice and Bob are sometimes corrupted in transmission. Define a *noisy transcript* as a 4-tuple $(u, u', w, w') \in \Sigma^{4T}$. The interpretation is that Alice sends the sequence of symbols $u = (u_1, \ldots, u_T)$, but they are received by Bob as the corrupted sequence $u' = (u'_1, \ldots, u'_T)$; similarly, Bob sends the sequence w, and Alice receives the corrupted sequence w'.

Let f be a function on domain $D_A \times D_B$. A very desirable property of a communication protocol for f is that the protocol succeeds in computing $f(x, y)$ whenever the number of transmission errors is not too large; that is, each of Alice and Bob can deduce the value of $f(x, y)$ from their own part of the input, along with the sequence of corrupted messages received. Formally, let $(\mathcal{P}, Eval)$ be a $2T$-bit communication protocol for f. Say that the

evaluation rule $Eval$ is *resilient for* f with respect to inputs $x, y \in D_A \times D_B$ and noisy transcript $(u, u', w, w') \in \Sigma^{4T}$, if

$$Eval_A(x, u, w') = Eval_B(y, u', w) = f(x, y).$$

That is, Alice and Bob both succeed in evaluating f correctly when they apply their evaluation functions to their *actual* sent messages, and their *corrupted* received messages. Note, there is no assumption here that Alice and Bob are following $\mathcal{P}$.

Say that $(\mathcal{P}, Eval)$ is *δ-error-resilient for f* if for any $x, y \in D_A \times D_B$, $Eval$ is resilient for f with respect to x, y, and any noisy transcript (u, u', w, w') satisfying

1. (u, w') is $\mathcal{P}_A$-faithful for x, and (u', w) is $\mathcal{P}_B$-faithful for y;
2. $\Delta(u, u') + \Delta(w, w') \leq 2\delta T$.

That is, whenever Alice and Bob both follow the protocol $\mathcal{P}$, and the total fraction of transmission errors is at most δ, both players succeed in evaluating $f(x, y)$ correctly.

Braverman and Rao proved the following result:

Theorem 8. *[BR10, Theorem 1] Suppose there is a T-bit protocol to compute the (possibly non-Boolean) function f. Then for any $\varepsilon > 0$, there is an $O(T/\varepsilon)$-bit protocol $(P_\varepsilon, Eval_\varepsilon)$, that is $(1/4-\varepsilon)$-error-resilient for f and uses messages over a constant-size alphabet $\Sigma = \Sigma_\varepsilon$.*

They prove Theorem 8 for the case where f is the pointer-jumping problem, $f := \mathrm{PJ}_T$, from which the result for general f follows easily. Their result improves on an earlier result of Schulman [Sch96], who defined error-resilient communication protocols and gave a similar result, sending *binary* messages and tolerating a $1/240$ fraction of transmission errors. (In [BR10] the authors also construct $(1/8 - \varepsilon)$-error-resilient protocols using binary messages. The protocols from Theorem 8 will be easier for us to use, however.) The quantitative improvement in the more recent work is not needed for our work, but the algorithmic details of the Braverman-Rao protocol make it easier to apply.

4.5 Sensibleness

Related to error-resilience is a property of communication protocols we call *sensibleness*. Suppose that $\mathcal{P} = (\mathcal{P}_A, \mathcal{P}_B, Eval_A, Eval_B)$ is a T-bit communication protocol computing some function $f : D_A \times D_B \to S$, with message alphabet Σ. Given $x \in D_A$, define $Poss_A(x) \subseteq S$, the *possible outcomes for Alice with respect to x*, as

$$Poss_A(x) := \{s \in S : \exists y \in D_B \text{ such that } f(x, y) = s\}.$$

Similarly for $y \in D_B$, define the *possible outcomes for Bob with respect to y* as $Poss_B(y) := \{s \in S : \exists x \in D_A \text{ such that } f(x, y) = s\}$. We say that $\mathcal{P}_A$ is *sensible* with respect to some $S' \subseteq S$ if, for any input $x \in D_A$ to Alice and any transcript (α, β) that is $\mathcal{P}_A$-faithful with respect to x, we have

$$Eval_A(x, \alpha, \beta) \in Poss_A(x) \cup \overline{S'}.$$

That is, no matter how much Bob deviates from the protocol (or has his messages corrupted in transmission), Alice will never produce a guess s for $f(x, y)$ that lies in S', unless that guess is consistent with *some* value y that might be held by Bob.

We define sensibleness for $\mathcal{P}_B$ with respect to S' analogously. We say $\mathcal{P}$ is sensible with respect to S' if $\mathcal{P}_A, \mathcal{P}_B$ are both sensible with respect to S'. When we define it in Section 4.8, we will observe that the Braverman-Rao protocol has a useful sensibleness property.

4.6 Tree Codes

A *d-ary tree code* of depth n and alphabet Σ is defined by an *encoding function* $C : [d]^{\leq n} \to \Sigma$. For $(v_1, \ldots, v_k) \in [d]^{\leq n}$, define

$$\overline{C}(v_1, \ldots, v_k) := (C(v_1), C(v_1, v_2), \ldots, C(v_1, v_2, \ldots, v_k)).$$

The definition can be interpreted as follows. A sequence $(v_1, \ldots, v_k)$ defines a path of length $k \leq n$ in the complete, rooted, directed d-ary tree of depth n (call this tree $\mathcal{T}$): we begin at the root, and on the j^{th} step of the path, we follow the outgoing edge index by v_j. We may view C as a *labeling* of the edges of $\mathcal{T}$: namely, $C(v_1, \ldots, v_k)$ gives the label of the final edge followed on path $(v_1, \ldots, v_k)$, and $\overline{C}(v_1, \ldots, v_k) \in \Sigma^k$ gives the full sequence of labels seen along this path.

Given two sequences $u, v \in [d]^k$, where $k \leq n$, let $\ell(u, v) := k + 1 - \min_j\{u_j \neq v_j\}$. Considering u, v as vertices in $\mathcal{T}$ (where a length-k path is identified with its terminal vertex), $\ell(u, v)$ indicates the distance from w to u, where w is the closest common ancestor of u and v. We say the tree code C has *distance* $\mu > 0$ if for all $k \leq n$ and $u, v \in [d]^k$, we have

$$\Delta(\overline{C}(u), \overline{C}(v)) \geq \mu \cdot \ell(u, v).$$

(In words: $\overline{C}(u)$ and $\overline{C}(v)$ differ in at least a μ fraction of the positions where they could *possibly* differ, namely, in their final $\ell(u, v)$ symbols.)

In the protocols we study, tree codes are intended to be decoded in the presence of errors. Given C as above, and given an arbitrary string $\sigma = (\sigma_1, \ldots, \sigma_k) \in \Sigma^k$ where $k \leq n$, define

$$D(\sigma) := \operatorname*{argmin}_{u \in [d]^k} \Delta\left(\overline{C}(u), \sigma\right)$$

as the message u whose encoding is closest to σ. In case of ties, D breaks ties nondeterministically; for the application of the tree codes in the Braverman-Rao protocol, such nondeterministic decoding suffices to give the desired behavior. However, to promote clarity we will work with a deterministic refinement of the decoder: we define

$$D^{lex}(\sigma) \in [d]^k$$

as the *lexicographically smallest* u that minimizes $\Delta\left(\overline{C}(u), \sigma\right)$.

The following theorem on the existence of tree codes was proved by Schulman in [Sch96], where the tree code and analysis given are attributed to collaboration with Rabani and Goldreich.

Theorem 9. *[Sch96, Lemma 1B] For every $\mu \in (0,1)$ and $n > 0$, there is a d-ary tree code $C = C_{\mu,n}$ of distance μ for depth n, with alphabet size $|\Sigma| = O_{d,\mu}(1)$.*

The code $C_{\mu,n}$ in [Sch96] is constructed probabilistically. No explicit construction is known; however, this will not be a major obstacle for our purposes, since we are working with computationally unbounded debaters. We will, however, need some constructive properties of the code. To establish these properties, we review the definition of the code from [Sch96].

To define $C_{\mu,n}$, we fix a prime number $p \geq d$, and a sequence $r = (r_1, \ldots, r_n)$ of elements of the finite field $\mathbb{F}_p$. We take $\Sigma := \mathbb{F}_p$, and define a code $C_{(r)} : [d]^{\leq n} \to \Sigma$ by the encoding rule

$$C_{(r)}(u_1, \ldots, u_k) := \sum_{j=1}^{k} (u_j - 1) r_{k+1-j},$$

with arithmetic over $\mathbb{F}_p$. It is shown in [Sch96] that if $p = O_{d,\mu}(1)$ is large enough, then the above code is a tree code of distance μ, for some "good" choice of $(r_1, \ldots, r_n)$. Importantly for us, such a code requires only $O_{d,\mu}(n)$ bits to specify.

4.7 Edge encodings for the Braverman-Rao protocol

A central ingredient in the pointer-jumping protocol of Braverman and Rao (specifically, the protocol that appears in [BR10, Section 5], with which the authors prove Theorem 8) is a method of encoding consistent subsets of the edges of a complete binary tree $\mathcal{T}$ of depth $T > 0$. In this subsection we describe the encoding method. While we follow [BR10] closely, we will also introduce a few new notations.

The tree $\mathcal{T}$ has edge-set $\mathcal{X} \cup \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the edges at odd and even depths of $\mathcal{T}$ respectively. (The *depth* of an edge e is the length of the path from the root of $\mathcal{T}$ ending at the child vertex of e. So, the edges leaving the root are of depth 1, and lie in $\mathcal{X}$.)

Given an edge $e \in \mathcal{X} \cup \mathcal{Y}$ and a sequence $h \in \{0,1\}^{>0}$, we define

$$edge(e; h) \in \mathcal{X} \cup \mathcal{Y} \cup \{\perp\}$$

as the final edge visited if we start at the child vertex of e, then following the path described by h. For example, $edge(e; 0)$ is the left edge out of the child vertex of e. If, in either case, the path described by h is ill-defined (because we hit a leaf node of $\mathcal{T}$ with additional steps left to take), we set $edge(e; h) := \perp$. We also define $edge(\emptyset; h)$ similarly to $edge(e; h)$, except we start our walk at the root vertex of $\mathcal{T}$.

Finally, define $vertex(e; h)$ as the child vertex of $edge(e; h)$, and similarly for $vertex(\emptyset; h)$.

Fix a parameter $R > 0$. (In the proof of Theorem 8, Braverman and Rao set $R = O(T/\varepsilon)$; we will use $\varepsilon := 1/8$, so $R = O(T)$.) Define the finite alphabet

$$\Pi := \{0, 1, 2, \ldots, R\} \times \{0,1\}^{\in\{1,2\}},$$

where $\{0,1\}^{\in\{1,2\}}$ are the binary strings of length 1 or 2. The first step in defining the Braverman-Rao encoding is to define a map[8]

$$E : \Pi^{\leq R} \to \mathcal{P}(\mathcal{X} \cup \mathcal{Y}),$$

[8]The definition we give is equivalent to the map E in [BR10, Section 4].

where $\mathcal{P}(\mathcal{X} \cup \mathcal{Y})$ denotes the set of subsets of $\mathcal{X} \cup \mathcal{Y}$. For $a = (a_1, \ldots, a_k) \in \Pi^k$, we will define a sequence $e_1, \ldots, e_k$ of edges of $\mathcal{T}$; some e_i's may remain undefined, an outcome we will indicate by $[e_i = \perp]$. Each e_i will be determined by $a_1, \ldots, a_i$. Then, we take

$$E(a) := \{e_i \mid i \in [k], e_i \neq \perp\}.$$

Following [BR10, Section 4], we inductively define the edges e_i for $i = 1, \ldots, k$. Given $a \in \Pi^k$, let $a_i = (r_i, s_i) \in \{0, 1, \ldots, R\} \times \{0,1\}^{\in\{1,2\}}$. (The idea is that r_i is a pointer to a_{r_i}.)

1. If $r_i = 0$, define $e'_i := edge(\emptyset; s_i)$.
2. If $r_i < i$ and $e_{r_i} \neq \perp$, set $e'_i := edge(e_{r_i}; s_i)$.
3. If $e'_i \neq \perp$, and there is no $j < i$ for which e_j, e'_i have a common parent, set $e_i := e'_i$. Otherwise, set $e_i := \perp$.

Next, following [BR10, Section 5], we build on the mapping E to give a second encoding of consistent edge-sets, where edge-sets are now to be encoded by sequences over the constant-sized alphabet

$$\Gamma := \{<, 0, 1, >, \emptyset\}.$$

We again fix some $R > 0$. We also fix some binary encoding of elements of Π, which we'll denote $desc : \Pi \to \{0,1\}^*$. $desc$ is selected so that the description $desc((\ell, s))$ is at most $c_0 \log \ell$ bits long, for each $(\ell, s) \in \Pi$ and for some fixed $c_0 > 0$.

For $k \leq R$, a sequence $\gamma = (\gamma_1, \ldots, \gamma_k) \in \Gamma^k$ defines a sequence $z(\gamma) = (z_1, \ldots, z_k) \in \Pi^k$. The definition of z_i given below may appear mysterious; the basic idea is that the encoding rule makes it possible to encode a value $z_i = (r_i, s_i) \in \Pi$ with $r_i < i$, using a string w that is *short* whenever r_i is not too much smaller than i. This allows the Braverman-Rao protocol to make progress quickly.

For $i = 1, \ldots, k$, we define $z_i \in \Pi$ as follows:

1. First suppose $(\gamma_1, \ldots, \gamma_i)$ is of form $(\gamma', <, w, >)$, where $\gamma' \in \Gamma^*$ and $w \in \{0,1\}^*$. In this case let $j < i$ be the index immediately preceding the substring w, so that $\gamma_j =$ "<" and $\gamma_i =$ ">".

   If, additionally, $w = desc((\ell, s))$ for some $(\ell, s) \in \Pi$, then set

   $$z_i := (\max\{j - \ell, 0\}, s).$$

   In this case we say that i is a *viable index* for γ.

2. If either of the two conditions in part 1 are not met (i is not viable), set $z_i := (i, 00)$.

Using this definitions, we define a second mapping[9]

$$E' : \Gamma^{\leq R} \to \mathcal{P}(X \cup Y),$$

by the rule

$$E'(\gamma) := E(z(\gamma)).$$

Note that from the definition of the edges $e_1, \ldots, e_k$ defined by $z(\gamma)$, it follows that $e_i = \perp$ whenever i is not a viable index for γ. However, some viable indices i for γ may still have $e_i = \perp$. If $e_i \neq \perp$, we say that i is an *effective index* for γ.

[9]The map E' is also denoted E in [BR10, Section 5]; we introduce a separate notation.

4.8 The Braverman-Rao protocol

We are now ready to define the Braverman-Rao protocol that we will use in our work (following [BR10, Section 5]). This is the error-resilient protocol $(\mathcal{P}_{BR,\varepsilon}, Eval_\varepsilon)$ used to prove Theorem 8 for the case where $f = \mathrm{PJ}_T$ is the pointer-jumping problem.

The protocol proceeds in $R = O(T/\varepsilon)$ *rounds*, where each round i consists of a message $\alpha_i \in \Sigma$ sent by Alice to Bob, followed by a message $\beta_i \in \Sigma$ sent by Bob to Alice. (We will define $\Sigma = \Sigma_\varepsilon$ shortly.) We let $\alpha, \beta \in \Sigma^R$ denote the sequences of messages sent by Alice and Bob. As the protocol is designed to be error-resilient, it is helpful to state the behavior of the two parties as determined by a (possibly-corrupted) sequence of received messages. We will use $\alpha'_i \in \Sigma$ to denote the possibly-corrupted i^{th} message received by Bob, and β'_i to denote the i^{th} message received by Alice.

Recall the tree codes $C_{(r)}$ from Section 4.6. We will apply these codes to sequences over the alphabet Γ (so, $d = 5$). We take $n := R$, $\mu := 1 - \varepsilon$, and let $p = O_\varepsilon(1)$ be an appropriately large prime number. Then we fix any "good" sequence $r \in \mathbb{F}_p^R$, i.e., one such that $C_{(r)} : \Gamma^{\leq R} \to \mathbb{F}_p$ is a 5-ary tree code of distance $1 - \varepsilon$. Let $D^{lex}_{(r)}$ be the deterministic decoding function for $C_{(r)}$, as defined in Section 4.6. We set $\Sigma := \mathbb{F}_p$.

In the protocol, Alice will maintain a sequence $(a_1, a_2, \ldots) \in \Gamma^*$, as well as a sequence $A_1, A_2, \ldots$ of subsets of $\mathcal{X} \cup \mathcal{Y}$. The values of a_i, A_i will be defined on the i^{th} round and be used to determine Alice's i^{th} message α_i. Similarly, Bob will maintain sequences $(b_1, b_2, \ldots) \in \Gamma^*$, and $B_1, B_2, \ldots \subseteq \mathcal{X} \cup \mathcal{Y}$, with b_i, B_i defined on the i^{th} round. We define these sequences first; the protocol will then be simple to state.

Specifically, we define[10]

$$A_i := \begin{cases} \emptyset & \text{if } i = 1, \\ (E'(a_1, \ldots, a_{i-1}) \cap \mathcal{X}) \cup \left(E'(D^{lex}_{(r)}(\beta'_1, \ldots, \beta'_{i-1})) \cap \mathcal{Y}\right) & \text{else,} \end{cases} \tag{1}$$

and similarly, we set

$$B_i := \begin{cases} \emptyset & \text{if } i = 1, \\ (E'(b_1, \ldots, b_{i-1}) \cap \mathcal{Y}) \cup \left(E'(D^{lex}_{(r)}(\alpha'_1, \ldots, \alpha'_{i-1})) \cap \mathcal{X}\right) & \text{else.} \end{cases} \tag{2}$$

Observe that A_i is indeed computable by Alice at and Bob respectively at the beginning of round i, since $a_1, \ldots, a_{i-1}$ and $b_1, \ldots, b_{i-1}$ are held by Alice and Bob respectively at that time.

For any $(a_1, \ldots, a_{i-1}, \beta'_1, \ldots, \beta'_{i-1})$, the set A_i is the union of a consistent subset of $\mathcal{X}$ with a consistent subset of $\mathcal{Y}$; so A_i is itself consistent. Similarly, B_i is consistent. Thus, the vertices $v(A_i), v(B_i)$ are each well-defined (recall the definitions from Section 4.2).

[10] The definition is equivalent to the one in [BR10], but stated slightly differently. In particular, whenever $a_1, \ldots, a_{i-1}$ are produced according to the Braverman-Rao protocol (under any amount of noise in the communication channel), we always have $E'(a_1, \ldots, a_{i-1}) \subseteq \mathcal{X}$ and so intersecting with $\mathcal{X}$ is unnecessary. However, we wish to have a version of the protocol whose behavior is well-defined on *all* sequences $(a_1, \ldots, a_{i-1}) \in \Gamma^{i-1}$.

We also differ slightly from [BR10] in that we work with the tree-code decoding function D^{lex} from Section 4.6. This does not affect correctness of the protocol, since D^{lex} is a deterministic refinement of the original decoder.

Next, the symbol $a_i \in \Gamma$ is defined in terms of $(a_1, \ldots, a_{i-1}, A_i)$, by the rule below. The definition appears complicated, but the idea is simple: if $v(A_i) \neq v(A_i \cup X)$, Alice is trying to encode the edge $e \in X$ that will extend the path in A_i. She can't do this in one step, since e takes several bits to encode; so, she extends the encoding one symbol at a time. Here is the formal definition:

1. If $v(A_i) = v(A_i \cup X)$, let $a_i :=$ "$\emptyset$".
2. If $v(A_i) \neq v(A_i \cup X)$, let e be the edge of X whose parent is $v(A_i)$. Now inspect the form of $(a_1, \ldots, a_{i-1})$. First, suppose it is of form $(u, <, w)$ where u is of some length $j - 1 \geq 0$, and such that:
   (a) $w \in \{0,1\}^*$ is a (possibly empty) prefix of a string $w' = desc((\ell, s))$, for some $(\ell, s) \in \Pi$; **and**,
   (b) **Either** $v(A_i)$ is the root of $\mathcal{T}$, $\ell = j$, and $edge(\emptyset; s) = e$;
   or, e is a descendant of $e_{j-\ell} = e_{j-\ell}(a_1, \ldots, a_{j-\ell})$, with $e_{j-\ell} \in \mathcal{X}$ and $e = edge(e_{j-\ell}; s)$.[11]

   In this case, set a_i so as to extend w according to w' (or, if $w = w'$, let $a_i :=$ ">").[12]
3. If $v(A_i) \neq v(A_i \cup X)$ but $(a_1, \ldots, a_{i-1})$ is not of the form described in item 2, set $a_i :=$ "$<$".

The definition of b_i for Bob is perfectly analogous; moreover, this definition is not even needed for our work. However, we state it below for completeness:

1. If $v(B_i) = v(B_i \cup Y)$, let $b_i :=$ "$\emptyset$".
2. If $v(B_i) \neq v(B_i \cup Y)$, let e be the edge of Y whose parent is $v(B_i)$. Now inspect the form of $(b_1, \ldots, b_{i-1})$. First, suppose it is of form $(u, <, w)$ where u is of some length $j - 1 \geq 0$, and such that:
   (a) $w \in \{0,1\}^*$ is a prefix of a string $w' = desc((\ell, s))$, for some $(\ell, s) \in \Pi$; **and**,
   (b) **Either** $v(B_i)$ is the root of $\mathcal{T}$, $\ell = j$, and $edge(\emptyset; s) = e$;
   or, e is a descendant of $e_{j-\ell} = e_{j-\ell}(b_1, \ldots, b_{j-\ell})$, with $e_{j-\ell} \in \mathcal{Y}$ and $e = edge(e_{j-\ell}; s)$.

   In this case, set b_i so as to extend w according to w' (or, if $w = w'$, let $b_i :=$ ">").
3. If $v(B_i) \neq v(B_i \cup X)$ but $(b_1, \ldots, b_{i-1})$ is not of the form described in item 2, set $b_i :=$ "$<$".

Now we can state the protocol.

[11] Again, when $a_1, \ldots, a_{i-1}$ are produced according to $\mathcal{P}_A$, we will always have $e_{j-\ell} \in \mathcal{X}$ if this edge is defined.

[12] Note that w, w' are uniquely determined, since we can have $e = edge(e_{j-\ell}; s)$ for at most one pair (ℓ, s) with $s \in \{0,1\}^{\in 1,2}$ and $e_{j-\ell} \in \mathcal{X}$. Here we are using that for any sequence $a_1, \ldots, a_{i-1}$ whatsoever, the defined edges among $e_1, \ldots, e_{i-1}$ are distinct.

Protocol $\mathcal{P}_{BR,\varepsilon}$: For $i = 1, \ldots, R$:

1. Alice transmits $\alpha_i := C_{(r)}(a_1, \ldots, a_i)$;
2. Bob transmits $\beta_i := C_{(r)}(b_1, \ldots, b_i)$.

Evaluation functions $Eval_\varepsilon = (Eval_A, Eval_B)$:

1. Alice outputs $Eval_A := v(A_R)$;
2. Bob outputs $Eval_B := v(B_R)$.

In [BR10] it is shown that $(\mathcal{P}_{BR,\varepsilon}, Eval_\varepsilon)$ is an $(1/4 - \varepsilon)$-error-resilient protocol for PJ_T, when $C_{(r)}$ is a tree code of distance $1 - \varepsilon$. However, note that the protocol is at least well-defined for *any* setting of $r \in \mathbb{F}_p^R$. (This is useful since in our debate systems, the debaters will have to supply the values of r themselves, and they may provide values that do not yield a good tree code.)

The output of both players' evaluation function in $\mathcal{P}_{BR,\varepsilon}$ is always some vertex of $\mathcal{T}$. Recalling the definition of sensibleness from Section 4.5, we have:

Claim 10. *The protocol $(\mathcal{P}_{BR,\varepsilon}, Eval_\varepsilon)$ defined above is sensible with respect to the set S' consisting of all leaf vertices of $\mathcal{T}$.*

Proof. We show sensibleness for Alice with respect to S'; Bob is handled similarly. It is not hard to see that for *any* sequence $\beta \in \Sigma^R$, the set $A_i \cap \mathcal{X} = E'(a_1, \ldots, a_{i-1}) \cap \mathcal{X}$ is always contained in X for each $i \in [R]$; this is because, when following $\mathcal{P}_A$, Alice only encodes edges of X with $a_1, \ldots, a_R$. It follows that the vertex $Eval_A(X, \alpha, \beta) = v(A_R)$ is always reachable by some path $\mathcal{P}$ in $\mathcal{T}$ whose intersection with $\mathcal{X}$ is contained in X. Thus if $v(A_R)$ is a leaf vertex v, we must have $v \in Poss_A(X)$. So $\mathcal{P}_A$ is sensible with respect to S'. □

5 Building stable debate systems

In this section we state our technical lemma (Lemma 11) on the existence of "encoding-checkers" for the Braverman-Rao protocol, and use it to prove Lemma 7 from Section 3.2, completing the proof of Theorem 6. Lemma 11 will be proved in Section 6.

5.1 Debate circuits for the Braverman-Rao protocol

Our main technical tool is Lemma 11 below, which exhibits a family of constant-turn debate circuits. Given a sequence r and communication transcript (α, β), this debate allows the players to debate whether Alice is faithfully following the Braverman-Rao protocol in (α, β) with respect to $C_{(r)}$, with a "good" choice of r.

Remark. In our work with debate circuits, we will frequently use a space-saving shorthand. We sometimes say that a debate circuit Φ takes as part of its start-variables an input sequence $h \in S^{\leq t}$, for some alphabet S and parameter $t \geq 1$. More formally, the actual input is a *binary representation* of a sequence $h' \in (S \cup \{\perp\})^t$, where we choose any fixed-length binary representation of the alphabet $S \cup \{\perp\}$ and use a concatenated encoding for h'. The debate circuit Φ expects h' to be of the form $\{S\}^k \times \{\perp\}^{t-k}$, for some $k \in \{0, 1, \ldots, t\}$. In Lemma 12 of Section 6.1, we will show that there is a linear-sized circuit that checks that h' is of this form, and also computes the value k.

The debate circuit Φ will be understood to output 0 if h' is not of the expected form. Similarly, we will often design debate circuits in which some Player $b \in \{0, 1\}$ is asked to give a sequence $h \in S^{\leq t}$. The implicit understanding is that Player b is expected to supply a sequence $h' \in S^k \times \{\perp\}^{t-k}$ for some $k \leq t$. The debate circuit tests this condition, and immediately outputs $1 - b$ if it is not satisfied. Thus, we may assume that under optimal play by both players, the sequences supplied are of the expected form.

Lemma 11. *Fix any $\varepsilon \in (0, 1/4)$ with finite binary expansion, and let $p = O_\varepsilon(1)$ be chosen as in Section 4.8. For $T > 0$, let $R = R_{T,\varepsilon} = O(T/\varepsilon)$ be as in Section 4.8, and let $\Sigma = \Sigma_\varepsilon = \mathbb{F}_p$. Given a sequence $r \in \mathbb{F}_p^R$, let*

$$\mathcal{P}_{BR,\varepsilon} = \mathcal{P}_{BR,\varepsilon}[r] = (\mathcal{P}_A, \mathcal{P}_B), \quad Eval_\varepsilon = (Eval_A, Eval_B)$$

be the protocol defined in Section 4.8 relative to tree code $C_{(r)}$ and parameter T (we suppress the dependence on r in our notation).

There exists a family $\left\{\Phi_T^{BR,\varepsilon}\right\}_{T>0}$ of debate circuits, where $\Phi_T^{BR,\varepsilon}$ has as start-variables a transcript $(\alpha, \beta) \in \Sigma^{2R}$, along with sequences $h \in \{0,1\}^{\leq T}$ and $r \in \mathbb{F}_p^R$.

We have $\mathrm{Val}(\mathcal{G}_{\Phi_T^{BR,\varepsilon}}[\alpha, \beta, h, r]) = 1$ if and only if:

1. *$C_{(r)}$ is a tree code of distance at least $1 - \varepsilon$;*
2. *There exists a maximal consistent set $X \subset \mathcal{X}$, for which (α, β) is $\mathcal{P}_A$-faithful with respect to X;*
3. *The set X meeting condition 2 can be chosen to satisfy $Eval_A(X, \alpha, \beta) = vertex(\emptyset; h)$.*

Also, $\Phi_T^{BR,\varepsilon}$ has $O(1)$ debate-blocks, is of size $O_\varepsilon(T)$, and is constructible in time $\mathrm{poly}_\varepsilon(T)$.

We will only need to apply Lemma 11 for one arbitrary choice of $\varepsilon \in (0, 1/4)$, but proving it for all ε in this range presents no extra difficulty.

5.2 Proof of Lemma 7

Proof. Let $V(x, y)$ be a verifier defining a polynomial-time debate system for L. By assumption, for length-n inputs, V can be implemented by a uniform circuit $C_{V,n}$ of size $s(n)$.

By modifying the debate for $C_{V,n}$, we may ensure that the debate-blocks $y^1, \ldots, y^k$ are strictly alternating between the players and consist of a single bit each, with Player 1 getting the first block ($type(1) = 1$). We do this by padding the debate with irrelevant bits as

necessary; this doesn't affect the debate length or the efficiency of the verifier by more than a constant factor.[13]

Let $T = T_n \leq O(s(n))$ be the number of turns used by the modified debate circuit $C_{V,n}$. Let $\varepsilon := 1/8$; let p be a sufficiently large prime and let $R = R_{T,\varepsilon} = O(T)$ be as in Section 4.8. Let $N = O(T)$ be defined as the total size of the circuit $\Phi^{BR} := \Phi_T^{BR,\varepsilon}$ from Lemma 11.

Let $Enc_N : \{0,1\}^N \to \{0,1\}^{N'}$ be the code given by Theorem 5 (we have $N' = O(T)$), and $\eta > 0$ the constant from that Theorem. We can encode and decode messages of variable length $m \leq N/2$ under Enc_N, by simply fixing a 2-bit encoding of $\{0, 1, \perp\}$. In doing so, we will abuse notation and regard Enc_N as a mapping from $\{0,1\}^{\leq N/2} \to \{0,1\}^{N'}$, and similarly will regard Dec_N as a mapping from $\{0,1\}^{N'} \to \{0,1\}^{\leq N/2}$. Often when we apply Dec_N, we will have an "expected" length of the decoded message; if the actual decoded message is of a different length, the implicit convention is that we will either truncate it or pad it with arbitrary values to get a message of the desired length. (The intention here is that, by encoding messages of various lengths with the same codeword length N', we make them "count equally"; this will help us to establish the 0-stability property.)

Our debate system $V^{stab}(x, \cdot)$ works in phases, as follows:

Phase 1: Player 1 is asked for a string $\hat{r} \in \{0,1\}^{N'}$. (Player 1's implicit claim is that $\hat{r} = Enc_N(r)$ encodes some sequence $r \in \mathbb{F}_p^R$ for which $C_{(r)}$ is a tree code with distance $\geq 1 - \varepsilon = 7/8$. We have chosen N large enough that a sequence $r \in \mathbb{F}_p^R$ can be so encoded.)

Next, V^{stab} computes $r := Dec_N(\hat{r})$, as well as $\hat{r}' := Enc_N(r)$. If $\Delta(\hat{r}, \hat{r}') \geq \eta N'/2$, then V^{stab} immediately outputs 0. Otherwise, we proceed to Phase 2.

Phase 2: The players are asked to write down binary descriptions of symbols $(\alpha_1, \beta_1, \ldots, \alpha_R, \beta_R) \in \Sigma^{2R}$ in that order, where each α_i is written by Player 1 and each β_i is written by Player 0. Let $(\alpha, \beta) := (\alpha_1, \ldots, \alpha_R, \beta_1, \ldots, \beta_R)$.

Next, Player 1 provides a string $\hat{h} \in \{0,1\}^{N'}$. (The implicit claim here is that Player 1 was following $\mathcal{P}_A$ for some maximal consistent input X, and that $Eval_A(X, \alpha, \beta) = vertex(\emptyset; h)$, where $h \in \{0,1\}^{\leq T}$ and $\hat{h} = Enc_N(h)$.)

V^{stab} computes $h := Dec_N(\hat{h})$, as well as $\hat{h}' := Enc_N(h)$. If $\Delta(\hat{h}, \hat{h}') \geq \eta N'/2$ then V^{stab} immediately outputs 0. Otherwise, we proceed to Phase 3.

Phase 3: Player 1 is asked to provide a string $\widehat{trans} \in \{0,1\}^{N'}$. The verifier computes $(\alpha^*, \beta^*) := Dec_N(\widehat{trans})$, where (α^*, β^*) are (binary encodings of) two sequences in Σ^{2R}.

V^{stab} computes $\widehat{trans'} := Enc_N((\alpha^*, \beta^*))$. If $\Delta(\widehat{trans}, \widehat{trans'}) \geq \eta N'/2$, then V^{stab} immediately outputs 0.[14]

Next, if $\Delta_\Sigma((\alpha^*, \beta^*), (\alpha, \beta)) \geq R/100$, then V^{stab} immediately outputs 0. Otherwise, we proceed to Phase 4.

[13] It may, however, blow up the total number of *turns* by a large amount; we do not attempt to minimize turns in this paper (but see question 2 in Section 7).

[14] Here, we write $\Delta_\Sigma(\cdot, \cdot)$ to clarify that our Hamming distances are taken over the message alphabet Σ.

Phase 4: Let v denote the debate variables for Φ^{BR}, and let $(v^1, \ldots, v^k)$ be the debate-blocks of v in their order of assignment (Lemma 11 tells us that $k = O(1)$). For $j \in [k]$, let $type(j) \in \{0,1\}$ be the player who gets to assign v^j in the debate circuit Φ^{BR}.

In Phase 4, the players provide strings $\hat{v}^1, \ldots, \hat{v}^k \in \{0,1\}^{N'}$, in that order, with Player $type(j)$ providing $\hat{v}^j$. (The idea is that the players are to be encoding an execution of the debate Φ^{BR}.) V^{stab} computes $v^j := Dec_N(\hat{v}^j)$ for each $j \in [k]$, as well as $\hat{w}^j := Enc_N(v^j)$.

If for some j we have $\Delta(\hat{v}^j, \hat{w}^j) \geq \eta N'/2$, then let j_0 be the *minimal* such value; V^{stab} immediately outputs $1 - type(j_0)$. Otherwise, we proceed to Phase 5.

Phase 5: V^{stab} evaluates Φ^{BR} applied to start-variables r (as produced in Phase 1) along with (α^*, β^*, h) (from Phase 2), and with the assignments $(v^1, \ldots, v^k)$ to the debate-variables produced in Phase 3. Let $b \in \{0,1\}$ denote the output of Φ^{BR}. If $b = 0$ then V^{stab} immediately outputs 0. If $b = 1$, V^{stab} outputs 1 if *either* of the following two conditions hold:

(i) The sequence $h \in \{0,1\}^{\leq T}$ is of length less than T;

(ii) h is of length T, and $C_{V,n}(x, h) = 1$.

If neither of (i), (ii) holds, V^{stab} outputs 0.

This completes the description of the verifier V^{stab}. First we will analyze the efficiency properties of V^{stab}; then, we will show it has the desired 0-stability property. For efficiency, V^{stab} can certainly be implemented in polynomial time, since Φ^{BR}, Enc_N, Dec_N all have uniform circuits (and $N, T = O(s) \leq \text{poly}(n)$). But observe that V^{stab} can also be implemented as a uniform circuit of size $O(N) = O(T)$, given oracle access to circuits $\Phi^{BR}, Enc_N, Dec_N, C_{V,n}$. (The only significant computation required is to compute Hamming distances, and we will show in Section 6.1 that this can be done by linear-sized circuits.) Since Φ^{BR}, Enc_N, Dec_N are all computable by uniform circuits of size at most $\widetilde{O}(T) \leq \widetilde{O}(s(n))$, and the (modified) circuit $C_{V,n}$ is uniform of size $O(s(n))$, we conclude that V^{stab} is implementable by a uniform circuit $C_{V^{stab},n}$ of size $\widetilde{O}(s(n))$ as required.

We now argue that V^{stab} defines a 0-stable debate system for L, with security parameter $\delta = \Omega(1)$ to be determined.

First, suppose $x \in L$. Then the debate $C_{V,n}(x, y)$ can be won by Player 1, with some strategy $\mathcal{S}_1$. Recall that $C_{V,n}$ has a debate consisting of T strictly alternating 1-bit turns, with Player 1 playing first. Let $\mathcal{T}$ be the complete binary tree of depth T. We may view $\mathcal{S}_1$ as a consistent subset of $\mathcal{X}$, the odd-depth edges of $\mathcal{T}$: namely, for $(h_1, \ldots, h_{2k+1}) \in \{0,1\}^{2k+1 \leq T}$, the condition $edge(\emptyset; (h_1, \ldots, h_{2k+1})) \in \mathcal{S}_1$ means that, if the first $2k$ variables of the debate $y_1, \ldots, y_{2k}$ have been assigned as $h_1, \ldots, h_{2k}$, then the strategy $\mathcal{S}_1$ directs Player 1 to set $y_{2k+1} := h_{2k+1}$. If necessary, we can extend $\mathcal{S}_1$ to be defined on every sequence $h_1, \ldots, h_{2k+1}$, so that $\mathcal{S}_1$ is a *maximal* consistent subset of $\mathcal{X}$. Note, this expansion doesn't alter the fact that $\mathcal{S}_1$ is a winning strategy.

Define a Player 1 strategy $\mathcal{S}_1'$ for $V^{stab}(x, \cdot)$, as follows:

1. In Phase 1, provide an encoding $\hat{r} = ENC_N(r)$ for some $r \in \mathbb{F}_p^R$ for which $C_{(r)}$ is a tree code with distance 7/8. Such an r exists for our sufficiently large setting of p. Note, this guarantees that we reach Phase 2.

2. In Phase 2, faithfully execute the Alice-protocol $\mathcal{P}_A$ defined relative to $C_{(r)}$, with $\mathcal{S}_1 \subset \mathcal{X}$ used as Alice's "input." (Recall that $\mathcal{P}_A$'s behavior is well-defined for any behavior by Bob in setting the values of the sequence β.) After (α, β) have been defined, set $\hat{h} := Enc_N(h)$, where $h \in \{0,1\}^{\ell \leq T}$ satisfies $Eval_A(X, \alpha, \beta) = vertex(\emptyset; h)$. Note that we will reach Phase 3.

3. In Phase 3, set $\widehat{trans} := Enc_N((\alpha, \beta))$. Thus, we have $(\alpha^*, \beta^*) = (\alpha, \beta)$, and we reach Phase 4.

4. In Phase 4, simulate an optimal Player-1 strategy for the debate Φ^{BR}, with start-variables r, α, β, h as defined previously. If in the simulation $type(j) = 1$ and the simulated Player 1 strategy wants to set $v^j := u^j$, we set $\hat{v}^j := Enc_N(u^j)$. If $type(j) = 0$, then to obtain a Player 0 move for the simulated Φ^{BR}-debate turn j, Player 1 looks at $\hat{v}^j$ and decodes to the closest codeword encoding some valid assignment u^j to v^j, using u^j as the simulated move by Player 0.

5. (Phase 5: there is nothing for either player to do.)

We now argue that $\mathcal{S}_1'$ is winning for V^{stab}. Consider any strategy used by Player 0 against $\mathcal{S}_1'$. In Phase 2, $\mathcal{S}_1'$ follows $\mathcal{P}_A$ faithfully with respect to $\mathcal{S}_1$. Now, $\mathcal{P}_A$ is sensible with respect to the set S' of leaf vertices of $\mathcal{T}$ (Claim 10). Also, $\mathcal{S}_1$ is winning for $C_{V,n}(x, \cdot)$. Thus, the resulting value $Eval_A(\mathcal{S}_1, \alpha, \beta) = vertex(\emptyset; h)$ is *either* (i) a non-leaf vertex, or (ii) is a leaf vertex for which $C_{V,n}(x, h) = 1$.

In Phase 4, Player 1's honest behavior in the previous phases implies that properties 1-3 of Lemma 11 all hold. Also, Player 1 always encodes valid moves in Phase 4, so he is not punished at this stage. Since he is following an optimal strategy, Player 1 wins his own simulated debate in Phase 4.

If any move $\hat{v}^j$ by Player 0 in Phase 4 ($type(j) = 0$) is at distance $\geq \eta N'/2$ from a valid codeword, V^{stab} will output 1 in Phase 4. So assume the contrary. Then, for the Player-0 moves u^j used by Player 1 in his simulation in Phase 4, we have $u^j = Dec_N(\hat{v}^j) = v^j$. That is, the moves used by Player 1 in his own simulation are exactly those used by V^{stab} in evaluating Φ^{BR} in Phase 5. Thus in Phase 5, $b = 1$ (if we do not output 1 already in Phase 4). Also, our analysis of Phase 2 implies that one of conditions (i)-(ii) from Phase 5 holds. Thus, V^{stab} outputs 1. So $V^{stab}(x, \cdot)$ is winnable by Player 1 in this case, as needed.

Next, suppose $x \notin L$. Then the debate $V(x, \cdot)$ can be won by Player 0 with some strategy $\mathcal{S}_0$. As before, we can regard $\mathcal{S}_0$ as a consistent subset of edges in $\mathcal{T}$, this time a maximal consistent subset of $\mathcal{Y}$.

Define a Player 0 strategy $\mathcal{S}_0'$ for $V^{stab}(x, \cdot)$, as follows:

1. Phase 1: there is nothing for Player 0 to do. If Player 1 passes Phase 1, let $r \in \mathbb{F}_p^q$ be as computed by V^{stab} in this phase.

2. In Phase 2, faithfully execute the Bob-protocol $\mathcal{P}_B$ defined relative to $C_{(r)}$, with $\mathcal{S}_0 \subset \mathcal{Y}$ used as Bob's "input." ($\mathcal{P}_B$'s behavior is well-defined for any partial setting by Alice to α.)

3. (Phase 3: nothing for Player 0 to do.)

4. In Phase 4, simulate an optimal Player 0 strategy for the debate Φ^{BR}, with start-variables r, α^*, β^*, h as defined previously. The simulation and resulting settings to v^j (with $type(j) = 0$) are carried out analogously to the process in step 4 of the strategy $\mathcal{S}'_1$ defined earlier.

Now consider any Player 1 strategy played against $\mathcal{S}'_0$. Let $z \in \{0,1\}^M$ denote the complete resulting debate-string, where M denotes the debate-string length for V^{stab}; we have $M \leq Kn$ for some constant $K > 1$. We set $\delta := \eta/(100K) = \Omega(1)$. Let $z' \in \{0,1\}^M$ be any string at Hamming distance at most δM from z; we will show that $V^{stab}(x, z') = 0$. This will establish that V^{stab} is a 0-stable debate system for L with security δ, as needed.

For any debate-variable (or set of variables) u for V^{stab}, we let $u_{[z]}, u_{[z']}$ denote the assignments to u under z, z' respectively. We define $u_{[z]}, u_{[z']}$ similarly if u is some value computed by V^{stab} on computations $V^{stab}(x, z), V^{stab}(x, z')$ respectively.

Note that if $\Delta(\hat{r}_{[z']}, \hat{r}'_{[z']}) = \Delta(\hat{r}_{[z']}, Enc_N(r_{[z']})) > \eta N'/2$, then $V^{stab}(x, z')$ outputs 0 on Phase 1. On the other hand, if $\Delta(\hat{r}_{[z']}, Enc_N(r_{[z']})) \leq \eta N'/2$ (we may assume this holds), then $\Delta(\hat{r}_{[z]}, Enc_N(r_{[z']})) \leq \eta N'/2 + \delta M < \eta N'$. Now, Dec_N decodes from an η fraction of errors, so we then must have $r_{[z]} = Dec_N(\hat{r}_{[z]}) = r_{[z']}$.

Similarly, in Phase 2 on (x, z'), either V^{stab} outputs 0 or else we have $h_{[z]} = h_{[z']}$; and in Phase 3 on (x, z'), either V^{stab} outputs 0 or else $(\alpha^*, \beta^*)_{[z]} = (\alpha^*, \beta^*)_{[z']}$. In each case we assume the latter outcome.

In Phase 4 on (x, z), all of Player 0's assignments $\hat{v}^j_{[z]}$ (with $type(j) = 0$) are valid encodings of assignments $\hat{v}^j_{[z]} = Enc_N(v^j_{[z]})$, so that $\hat{v}^j_{[z]} = \hat{w}^j_{[z]}$. We then have $\Delta(\hat{v}^j_{[z']}, Enc_N(v^j_{[z]})) \leq \delta M < \eta N$, so that $v^j_{[z']} = v^j_{[z]}$ and $\hat{w}^j_{[z]} = \hat{w}^j_{[z']}$. Then, $\Delta(\hat{v}^j_{[z']}, \hat{w}^j_{[z']}) \leq \delta M < \eta N'/2$. It follows that in Phase 4 on (x, z'), the verifier V^{stab} cannot punish Player 0 by outputting 1.

Having established this, we can now conclude that, unless $V^{stab}(x, z')$ outputs 0 in Phase 4, we must have $v^j_{[z']} = v^j_{[z]}$ for all $j \in [k]$ with $type(j) = 1$. This is shown by the same argument we used to get $r_{[z]} = r_{[z']}$; we may assume the equality holds for each such j. We also draw attention again to the fact that $v^j_{[z']} = v^j_{[z]}$ when $type(j) = 0$. Summarizing, we have

$$(r, \alpha^*, \beta^*, h, v^1, \ldots, v^k)_{[z]} = (r, \alpha^*, \beta^*, h, v^1, \ldots, v^k)_{[z']}. \tag{3}$$

If $r_{[z]} = r_{[z']}$ defines a tree code $C_{(r)}$ of distance $< 7/8$, then condition 1 in Lemma 11 is violated by $(r, \alpha^*, \beta^*, h)_{[z]}$, and for the debate defined by Φ^{BR} we have

$$\mathrm{Val}(\mathcal{G}_{\Phi^{BR}}[(r, \alpha^*, \beta^*, h)_{[z]}]) = \mathrm{Val}(\mathcal{G}_{\Phi^{BR}}[(r, \alpha^*, \beta^*, h)_{[z']}]) = 0.$$

Now in $\mathcal{S}'_0$, Player 0 follows an optimal strategy in this debate, and $(v^1, \ldots, v^k)_{[z]} = (v^1, \ldots, v^k)_{[z']}$, so $b_{[z]} = b_{[z']} = 0$ and $V^{stab}(x, z') = 0$.

So suppose now that $r_{[z]}$ does define a tree code of distance $\geq 7/8$. We distinguish two further cases:

Case 1: $(\alpha^*, \beta^*)_{[z]}$ is not $\mathcal{P}_A$-faithful. In this case, on (x, z), Player 0 wins the debate Φ^{BR} with the start-variables $(r, \alpha^*, \beta^*, h)_{[z]}$, and the settings $(v^1, \ldots, v^k)_{[z]}$; i.e., in Phase 5 we have $b_{[z]} = 0$. Then Player 0 *also* wins the same debate on (x, z'), by the equalities in Eq. (3). Thus $V^{stab}(x, z') = 0$.

Case 2: $(\alpha^*, \beta^*)_{[z]}$ is $\mathcal{P}_A$-faithful, with respect to some Alice-input X. In this case, let us return our attention to Phase 3. On (x, z'), the second test in that phase implies that, for Player 1 to avoid losing in Phase 3, we must have $\Delta_\Sigma((\alpha^*, \beta^*)_{[z']}, (\alpha, \beta)_{[z']}) = \Delta_\Sigma((\alpha^*, \beta^*)_{[z]}, (\alpha, \beta)_{[z']}) < R/100$. Assume this holds. Then, as $\Delta(z, z') \leq \delta M$, we have $\Delta_\Sigma((\alpha^*, \beta^*)_{[z]}, (\alpha, \beta)_{[z]}) < R/100 + \delta M < R/4 = 2(1/4 - \varepsilon)R$.

Now, examine the noisy transcript[15] $(\alpha^*, \alpha, \beta, \beta^*)_{[z]}$; by our work above, it has fewer than $2(1/4 - \varepsilon)R$ transmission errors. The transcript $(\alpha^*, \beta^*)_{[z]}$ is $\mathcal{P}_A$-faithful with respect to X, and by the definition of $\mathcal{S}_0'$, the transcript $(\alpha, \beta)_{[z]}$ is $\mathcal{P}_B$-faithful with respect to Bob-input $\mathcal{S}_0$. As $\mathcal{P}_{BR,\varepsilon}$ is $(1/4 - \varepsilon)$-error-resilient, we conclude that

$$Eval_A(X, (\alpha^*, \beta^*)_{[z]}) = Eval_B(\mathcal{S}_0, (\alpha, \beta)_{[z]}) = PJ_T(X, \mathcal{S}_0).$$

(This is the first and only time we use the error-resilience property of $\mathcal{P}_{BR,\varepsilon}$.)

The value $PJ_T(X, \mathcal{S}_0)$ is a leaf vertex of $\mathcal{T}$, since $X, \mathcal{S}_0$ are maximal consistent subsets of $\mathcal{X}, \mathcal{Y}$; we have $PJ_T(X, \mathcal{S}_0) = vertex(\emptyset; h^*)$ for some $h^* \in \{0,1\}^T$. Also, because $\mathcal{S}_0$ is a winning strategy for $C_{V,n}(x, \cdot)$ for Player 0, we have $C_{V,n}(x, h^*) = 0$.

Now, there are two options for Player 1's behavior in the debate $V^{stab}(x, z)$: either he supplies an $\hat{h}_{[z]}$ that decodes to the "correct" value $h_{[z]} = h^*$, or he supplies an $\hat{h}_{[z]}$ that decodes to some other value $h_{[z]} \neq h^*$. In the first option, $V^{stab}(x, z)$ must output 0 in Phase 5 since conditions (i), (ii) of Phase 5 are violated by $h_{[z]}$. In the second option, Player 0 can and will win the debate Φ^{BR} on (x, z), as condition 3 in Lemma 11 is violated by the start-variables $(r, \alpha^*, \beta^*, h)_{[z]}$. In either event, $V^{stab}(x, z) = 0$. But then in either event, we also have $V^{stab}(x, z') = 0$, due to the equalities in Eq. (3).

Combining Cases 1 and 2 above, we conclude that $V^{stab}(x, z') = 0$, as was to be shown. This completes the proof of Lemma 7. □

6 Building encoding-checker debates

In this (fairly lengthy) section we prove Lemma 11 from Section 5.1.

6.1 Some functions computable by linear-size circuits

We will build up the debate circuits of Lemma 11 from simpler component debates. In all the debate circuits we construct, we will use certain basic circuits as building blocks. These building blocks are collected in the next lemma.

Lemma 12. *Let S be a finite set, and fix some standard binary encoding of $S \cup \{\perp\}$. The following functions can be computed by uniform circuits of size $O(n)$:*

[15] (as defined in Section 4.4)

(i) ***Input:*** *A pair (v, k), where $v \in S^n, k \in [n]$;*

Output: *the value $v_k \in S$.*

(ii) ***Input:*** *A string $x \in \{0,1\}^n$;*

Output: *$||x||_1$, the number of 1s in x.*

(iii) ***Input:*** *A pair (u, v), where $u, v \in S^n$;*

Output: *the distance $\Delta(u, v) \in \{0, 1, \ldots, n\}$.*

(iv) ***Input:*** *$v \in (S \cup \{\perp\})^n$;*

Output: *a pair $(b, k) \in \{0,1\} \times \{0, 1, \ldots, n\}$, where $b := 1$ iff $v \in S^j \times \{\perp\}^{n-j}$, for some $j \in \{0, 1, \ldots, n\}$. In this case $k := j$, otherwise $k := 0$.*

Proof. We describe the circuit constructions; in each case the uniformity follows by inspection. **(i):** Assume, by padding if necessary, that $n = 2^t$ for some $t \geq 0$. Also, it will be most convenient to assume that v is indexed as $v = (v_0, \ldots, v_{n-1})$ and that $k \in \{0, 1, \ldots, n-1\}$.

If $n = 1$, then $k = 0$ and we output v_0. Otherwise, define $v' \in S^{n/2}$ by the rule

$$v'_j := \begin{cases} v_j & \text{if } k \leq n/2, \\ v_{n/2+j} & \text{else.} \end{cases}$$

Which of the two cases we are in can be determined by looking at the most significant bit of the binary representation of k. Thus, v'_j can be determined with reference to $O(1)$ bits of v and k. Let $k' \in \{0, 1, \ldots, n/2 - 1\}$ be the number obtained from k by deleting the most significant bit of k. Then observe that $v_k = v'_{k'}$. Using a circuit of size $O(n)$, we have reduced from an instance of problem (i) with parameter n, to a single instance of problem (i) with parameter $n/2$. Iterating this procedure for $\log n$ steps reduces us to the base case $n = 1$, which we solve directly. The total circuit size is $O\left(\sum_{s \leq \log n} n2^{-s}\right) = O(n)$.

(ii): Again we may assume that $n = 2^t$ for some $t \geq 0$. Create a full binary tree $\mathcal{T}$ of depth t, whose leaf nodes are the bits of x. Then $||x||_1$ is the result if we place an integer addition gate at each non-leaf node of $\mathcal{T}$, and output the value at the root.

The addition gates at distance $s \leq t$ from the leaf nodes take as inputs two integers in the range $\{0, \ldots, 2^{s-1}\}$, which may be represented by bitstrings of length s. Addition of two s-bit numbers can be implemented by a circuit of size $O(s)$, and there are $2^{t-s} = 2^{-s}n$ additions to perform at level s. Thus the total contribution to the circuit size at level s is $O(s2^{-s}n)$. Summing over $s \leq t = \log_2 n$, the total circuit size is $O(n)$, since $\sum_{s>0} s2^{-s} < \infty$.

(iii): For each $i \in [n]$, we compute $x_i := [u_i \neq v_i]$. Each such bit can be computed using $O(1)$ operations. Then, $\Delta(u, v) \in \{0, 1, \ldots, n\}$ is equal to $||x||_1$, so we can apply part (ii).

(iv): For $i \in [n-1]$, let $p_i := \neg[v_i = \perp \ \wedge v_{i+1} \in S]$. Then p_i can be computed from v_i, v_{i+1} by a circuit of size $O(1)$. We can express the desired bit b as $b = \bigwedge_{i\in[n-1]} p_i$, allowing us to compute b with a circuit of size $O(n)$.

Next, for $j \in [n]$ let $x_j := [v_i \in S]$. Letting $k \in \{0, 1, \ldots, n\}$ be the value desired in problem (iv), we have

$$k = \begin{cases} ||x||_1 & \text{if } b = 1, \\ 0 & \text{else.} \end{cases}$$

Using our value b and our circuit from part (ii), we can compute k in size $O(n)$. □

6.2 Debate circuits to check tree-code encodings and decodings

The following lemma gives debate circuits to verify encodings under the tree codes $C_{(r)}$ defined in Section 4.6. The lemma applies, not just to a "good" choice of $r \in \mathbb{F}_p^n$ (i.e., one that yields the tree-code distance property claimed in Theorem 9), but to any setting of r.

Lemma 13. *Fix any $d > 1$ and a prime $p \geq d$. For each n, there exists a uniform debate circuit Φ_n, that takes as its start-variables a 3-tuple*

$$r \in \mathbb{F}_p^n, \quad u \in [d]^{\leq n}, \quad s \in \mathbb{F}_p^{\leq n}.$$

We have $\mathrm{Val}(\mathcal{G}_{\Phi_n}[r, u, s]) = 1$ *iff* $\overline{C_{(r)}}(u) = s$. *Also, Φ_n has 2 debate-blocks and is of total size $O(n)$.*

Proof. Rather than building the debate circuit Φ_n gate by gate, we describe the debate on a high level and argue that it can be implemented in size $O(n)$.

First, using part (iv) of Lemma 12, Φ_n checks that u, s are of equal length as sequences. If this test fails, Φ_n immediately outputs 0. Otherwise, suppose both are of length $k \leq n$; Φ_n retains the value k for future use. Next, Player 0 is asked to name a value $k^* \leq k$ for which he claims $C_{(r)}(u_1, \ldots, u_{k^*}) \neq s_{k^*}$. In support of his claim, Player 0 is also asked to provide a sequence $r' \in \mathbb{F}_p^{k^*}$, for which he claims $r' = (r_{k^*}, r_{k^*-1}, \ldots, r_1)$.

Using the value k^*, Φ_n checks that $\sum_{i=1}^{k^*}(u_i - 1)r'_i \neq s_{k^*}$ over $\mathbb{F}_p$. If this check fails, Φ_n immediately outputs 1. Otherwise, Player 1 is given a chance to challenge Player 0's claim that $r' = (r_{k^*}, r_{k^*-1}, \ldots, r_1)$. To do so, he names a value $k' \in [n]$. The debate circuit then outputs 1 iff $k' \leq k^*$ and $r'_{k'} \neq r_{k^*+1-k'}$.

Correctness of the debate follows immediately from the definition of $C_{(r)}$; also, Φ_n has 2 debate-blocks by construction. As for its efficiency, we just need to perform addition of $\leq n$ terms over a field of constant size; compare, add, and subtract $(\log n)$-bit numbers; and access the elements $r'_{k'}$, $r_{k^*+1-k'}$. The first two tasks are easily performed by a circuit of size $O(n)$, and for the third we use Lemma 12, part (i). □

The next lemma gives two debaters a way to debate the claim that a sequence $q \in \mathbb{F}_p^k$ decodes to $u \in [d]^k$ under the code $C_{(r)}$, using the deterministic decoder $D_{(r)}^{lex}$.

Lemma 14. *Fix $d > 1$ and a prime $p \geq d$. For each n, there exists a uniform debate circuit Λ_n, that takes as its start-variables three sequences*

$$r \in \mathbb{F}_p^n, \quad q \in \mathbb{F}_p^{\leq n}, \quad u \in [d]^{\leq n}.$$

We have $\mathrm{Val}(\mathcal{G}_{\Lambda_n}[r, q, u]) = 1$ *iff* $D^{lex}_{(r)}(q) = u$. *Moreover, Λ_n has $O(1)$ debate-blocks and is of total size $O(n)$.*

Proof. First, using Lemma 12, part (iv), Λ_n checks that q, u are of equal length $k \leq n$ as sequences; if not, Λ_n immediately outputs 0. Otherwise, we retain the value k.

In the first debate-block, Player 1 gives a sequence $s \in \mathbb{F}_p^{\leq n}$, claiming that $\overline{C_{(r)}}(u) = s$. This claim is then debated using the debate circuit Φ_n of Lemma 13. If Player 1 loses this debate, Λ_n immediately outputs 0. Thus we may assume from now on that Player 1 provides an s satisfying $\overline{C_{(r)}}(u) = s$.

Next, Player 0 is given a chance to name values $v \in [d]^{\leq n}, s' \in \mathbb{F}_p^{\leq n}$ for which he claims both of the following hold:

1. $s' = \overline{C_{(r)}}(v)$;
2. Either $[\Delta(s', q) < \Delta(s, q)]$, *or* $[\Delta(s', q) = \Delta(s, q)$ and $s' < s$ in lexicographic order$]$.

The players debate each of these conditions; Λ_n outputs 0 if Player 0 wins both debates, otherwise Λ_n outputs 1. By definition of $D^{lex}_{(r)}$, then, the debate is winnable by Player 0 iff $D^{lex}_{(r)}(q) \neq u$.

To debate condition 1, we use the debate circuit from Lemma 13, except that we now reverse the roles of the players in that circuit. For condition 2, the Hamming distances involved can be computed using Lemma 12, part (iii). The lexicographic comparison in condition 2 can be performed by an $O(n)$-sized circuit using a simple idea: by recursively comparing the first half of s with the first half of s', and the second half of s with the second half of s', we can determine whether $s' < s$. Thus we have $|\Lambda_n| = O(n)$, and Λ_n has $O(1)$ debate-blocks. □

6.3 Debate circuits for edge encodings

Next, we give efficient debate circuits that allow debaters to argue about the properties of specific edge-sets specified by the mapping E' defined in Section 4.7. Recall that Γ is the constant-sized alphabet used as inputs to E', and that elements of $\Pi = \{0, 1, \ldots, R\} \times \{0, 1\}^{\in\{1,2\}}$ are encoded by a mapping $desc : \Pi \to \{0, 1\}^{\leq c_0 \log R}$.

Our ultimate goal in this (long) subsection will be to prove the following technical lemma, which allows players to debate which edges are specified by an encoding over Γ^*.

Lemma 15. *Let $\mathcal{T}$ be the full binary tree of depth T, and let $R \geq T$. There exists a uniform debate circuit Ψ_R that takes as start-variables sequences $\gamma \in \Gamma^{\leq R}$ and $h \in \{0, 1\}^{\leq R}$. We have* $\mathrm{Val}(\mathcal{G}_{\Psi_R}[\gamma, h]) = 1$ *iff for the edge $e := edge(\emptyset; h)$ specified by h we have*

$$e \in E'(\gamma).$$

Ψ_R has $O(1)$ debate-blocks, and is of size $O(R)$.

Our first step is the next lemma, which allows the players to debate which indices of a sequence γ are viable.

Lemma 16. *There exists a uniform debate circuit Υ_R that takes as start-variables sequences $\gamma \in \Gamma^{\leq R}, i \in [R]$, and $w \in \{0,1\}^{\leq \lfloor c_0 \log R \rfloor}$. We have $\mathrm{Val}(\mathcal{G}_{\Upsilon_R}[\gamma, i, w]) = 1$ iff i is a viable index for γ, and w is equal to $desc(z_i)$, where $z_i = z_i(\gamma)$. Υ_R has a single debate-block, and is of size $O(R)$.*

Proof. First, Υ_R checks that i is at most the length of γ, and that $w = desc(a)$, for *some* $a \in \Pi$. The string w is $O(\log R)$ bits long, and this test can be performed by a circuit of size $O(\log R)$, since we're using a "reasonable" encoding function *desc*. If either test fails, Υ_R outputs 0. If both tests pass, we store $a \in \Pi$ as well as the length $|desc(a)|$.

Note that if the tests above pass, then condition 2 in the Lemma holds iff

$$(\gamma_1, \ldots, \gamma_i) = (\gamma', <, desc(a), >), \tag{4}$$

for some $\gamma' \in \Gamma^*$. Next, Player 0 is given a chance to provide a value $t \in \{0, 1, \ldots, |desc(a)| + 1\}$ for which either:

1. $t \in \{1, \ldots, |desc(a)|\}$ and $\gamma_{i-1-|desc(a)|+t} \neq w_t$;
2. $t = 0$, and $\gamma_{i-1-|desc(a)|} \neq$ "<";
3. $t = |desc(a)| + 1$, and $\gamma_i \neq$ ">".

Such a t can be found iff Eq. (4) is false. The values $\gamma_{i-|desc(a)|+t}$ and w_t can be computed and compared in $O(R)$ size, using part (i) of Lemma 12. □

As the next step towards proving Lemma 15, we introduce an auxiliary encoding function

$$\widetilde{E} : \Pi^{\leq R} \to \mathcal{P}(\mathcal{X} \cup \mathcal{Y}).$$

$\widetilde{E}$ is defined similarly to E from the last subsection, except that $\widetilde{E}$ is more "permissive" about allowing edges into the collections it defines. For $a = (a_1, \ldots, a_k) \in \Pi^k$, we define a sequence $\tilde{e}_1, \ldots, \tilde{e}_k$ of edges of $\mathcal{T}$, some of which remain undefined ($\tilde{e}_i = \perp$). We take

$$\widetilde{E}(a) := \{\tilde{e}_i \mid i \in [k], \tilde{e}_i \neq \perp\}.$$

Let $a_i = (r_i, s_i) \in \{0, 1, \ldots, R\} \times \{0,1\}^{\in\{1,2\}}$. Inductively, for $i = 1, \ldots, k$, define $\tilde{e}_i$ as follows:

1. If $r_i = 0$, let $\tilde{e}_i := edge(\emptyset; s_i)$.
2. If $r_i < i$ and $\tilde{e}_{r_i} \neq \perp$, let $\tilde{e}_i := edge(\tilde{e}_{r_i}; s_i)$.

By analogy with E', we also define a mapping $\widetilde{E}' : \Gamma^{\leq R} \to \mathcal{P}(\mathcal{X} \cup \mathcal{Y})$ by

$$\widetilde{E}'(\gamma) := \widetilde{E}(z(\gamma)).$$

Although $\widetilde{E}'(\gamma)$ is a larger set than $E'(\gamma)$, the edges $\{\tilde{e}_i\}$ still hold useful information about $E'(\gamma)$, and the mapping $\widetilde{E}'$ is simpler for the players for debate about than E'. We collect useful facts about $\widetilde{E}'$ in the next few lemmas. Lemma 17 below gives a debate circuit to debate edges encoded by $\widetilde{E}'$.

Lemma 17. *Let $\mathcal{T}$ be the full binary tree of depth T, and let $R \geq T$. There exists a uniform debate circuit $\widetilde{\Psi}_R$ that takes as start-variables sequences $\gamma \in \Gamma^{k \leq R}$, $h \in \{0,1\}^{\leq T}$, and an $i \in [k]$. We have $\mathrm{Val}(\mathcal{G}_{\widetilde{\Psi}_R}[\gamma, i, h]) = 1$ if and only if the following condition holds: letting $\tilde{e}_1, \ldots, \tilde{e}_k$ be defined in terms of $z(\gamma)$, we have*

$$\tilde{e}_i = edge(\emptyset; h).$$

$\widetilde{\Psi}_R$ has $O(1)$ debate-blocks, and is of size $O(R)$.

Proof. We collect some observations about $\widetilde{E}'$ before describing the the debate circuit. Let $z(\gamma_1, \ldots, \gamma_k) = (z_1, \ldots, z_k)$, and for $i \in [k]$, if $z_i \neq \perp$, let $z_i = (r_i, s_i) \in \Pi$. Observe that $\tilde{e}_i \neq \perp$ iff there is a sequence $1 \leq j_1 < \ldots < j_t = i$ of viable indices for $(\gamma_1, \ldots, \gamma_k)$, where:

1. $r_{j_1} = 0$;
2. If $t > 1$, then for $1 < t' \leq t$, $r_{j_{t'}} = j_{t'-1}$. That is, $z_{t'}$ "points back" to $z_{t'-1}$.

If these conditions are met, then we say that $(j_1, \ldots, j_t = i)$ is an *i-chain*. Observe also that if $(j_1, \ldots, j_t = i)$ is an i-chain, then $\tilde{e}_i$ is obtained from the concatenated path $s_{j_1} \ldots s_{j_t}$:

$$\tilde{e}_i = edge(\emptyset; s_{j_1} \ldots s_{j_t}).$$

Using this observation, the main part of the debate proceeds in two main phases, as follows.

Phase 1: First, Player 1 provides a sequence $1 \leq j_1 < \ldots < j_t$ of indices, which he claims is an i-chain. The sequence $(j_1, \ldots, j_t)$ is presented in the form of a string $w \in \{0,1\}^R$, where $w_\ell = 1$ encodes that ℓ is part of the chain. In the remainder of Phase 1, Player 0 is given an opportunity to challenge the claim that w encodes an i-chain. He is allowed to choose one of three ways of doing so, corresponding to the three possible ways that w may fail to encode an i-chain:

Option 1: Player 0 may name a value $\ell > i$, claiming that $w_\ell = 1$ (so that w cannot encode a proper i-chain); or, Player 0 may claim that $w_i = 0$. Either type of claim can be tested using part (i) of Lemma 12.

Option 2: Player 0 may claim that $r_{j_1} \neq 0$. To do this, he first provides a value $\ell \leq R$ for which he claims that $w_\ell = 1$ and $w_{\ell'} = 0$ for all $\ell' < \ell$ (so that $\ell = j_1$). The debate circuit may directly check that $w_\ell = 1$; the claim that $w_{\ell'} = 0$ for all $\ell' < \ell$ can be challenged by Player 1, who is given a chance to provide a counterexample value ℓ'.

If Player 0 passes this check, Player 0 then is asked to support his claim that $r_\ell \neq 0$. To do this, Player 0 may either claim that $z_\ell = \perp$, or that $z_\ell = (r_\ell, s_\ell)$ where $r_\ell \neq 0$. In either case, the claim can be evaluated using the debate circuit Υ_R from Lemma 16 (with roles assigned appropriately).

Option 3: Player 0 can argue that in the sequence $(j_1, \ldots, j_t)$ encoded by w, some value $t' \leq t$ fails condition 2 in the definition of an i-chain. Note that such a value t' exists exactly if there are indices $\ell < \ell'$ such that:

1. $w_\ell = w_{\ell'} = 1$;

2. for all $\ell'' \in \{\ell+1, \ldots, \ell'\}$, $w_{\ell''} = 0$;

3. Either ℓ' is not a viable index for $(\gamma_1, \ldots, \gamma_k)$, or ℓ' is viable but $r_{\ell'} \neq \ell$.

In Option 3, Player 0 is asked to provide indices $\ell < \ell'$ satisfying 1-3 above. Each condition can be tested efficiently by $\widetilde{\Psi}_R$: for the first condition we use part (i) of Lemma 12; for the second condition we ask Player 1 to provide a counterexample ℓ''; for the third condition, we apply the debate circuit Υ_R from Lemma 16.

If Player 0's challenge (using one of Options 1-3 above) succeeds, then $\widetilde{\Psi}_R$ outputs 0. Otherwise, we consider Player 1 to have established that w encodes an i-chain, and we proceed to Phase 2. Observe that Player 1 can withstand Player 0's challenge (under optimal play) exactly if w does encode an i-chain. Therefore, we will assume in the next phase that Player 1 does provide an i-chain (so in particular we have $\tilde{e}_i \neq \perp$), and successfully defends his claim.

Phase 2: Player 1 is asked to support his claim that $\tilde{e}_i = edge(\emptyset; h)$. To do so, he first provides a sequence $x \in \{0,1\}^{2R}$. For $\ell \in [R]$, the variables $(x_{2\ell-1}, x_{2\ell})$ are "intended" under honest play to be set to $(0,0)$ if $w_\ell = 0$. If $w_\ell = 1$, the intention is that they are set to $(1,0)$ if s_ℓ is of length 1, and set to $(1,1)$ if s_ℓ is of length 2. Player 0 is given an opportunity to challenge Player 1's assignment to x, using debate circuit Υ_R from Lemma 16. If Player 0's challenge succeeds, $\widetilde{\Psi}_R$ outputs 0. Thus, we may assume x is set honestly by Player 1. Note that in this case, $||x||_1$ is equal to the depth of $\tilde{e}_i$ in $\mathcal{T}$.

Next, $\widetilde{\Psi}_R$ uses part (ii) of Lemma 12 to compute $||x||_1$ and compare it to j, the length of h. If $||x||_1 \neq j$, then $\tilde{e}_i$ and $edge(\emptyset; h)$ are of different depths and cannot be equal; in this case, $\widetilde{\Psi}_R$ outputs 0. Otherwise, Player 0 is given a chance to argue that $\tilde{e}_i$ and $edge(\emptyset; h)$ are specified by different paths in $\mathcal{T}$. To do this, Player 0 provides a value $q \in [j]$, claiming that the two edges' defining paths differ at depth q—one follows an edge in $\mathcal{T}$ labeled "0", the other follows an edge labeled "1".

The value h_q, giving the behavior of the path specifying $edge(\emptyset; h)$ at depth q, can be computed in $O(R)$ size using Lemma 12, part (i). To compare this to the q^{th} bit specifying $\tilde{e}_i$, Player 0 is asked to provide a string $w' \in \{0,1\}^R$. $\widetilde{\Psi}_R$ expects w' to be a prefix of w, followed by 0s, with at least one 1 in w'. It tests this condition (easily done in $O(R)$ size), outputting 1 if the test fails; so we may assume that Player 0 supplies such a prefix, with $\ell^* \in [R]$ denoting the last 1-index in w'. (Player 0's implicit claim here is that ℓ^* is the i-chain index for which $z_{\ell^*} = (r_{\ell^*}, s_{\ell^*})$ determines the q^{th} step in the path specifying $\tilde{e}_i$.)

Player 0 is also asked to provide a string z', intended to be a description of z_{ℓ^*} under *desc*. The accuracy of the description z' can be tested using the circuit Υ_R from Lemma 16, so assume it is correct. In particular, we have access to the string $s_{\ell^*} \in \{0,1\}^{\in\{1,2\}}$.

Using w', the debate circuit computes a string $x' \in \{0,1\}^{2R}$, where

$$(x'_{2\ell-1}, x'_{2\ell}) := \begin{cases} (x_{2\ell-1}, x_{2\ell}) & \text{if } w'_\ell = 1, \\ (0,0) & \text{else.} \end{cases}$$

Note that x' can be computed from x and w' in $O(R)$ size. Also note that under our assumption on the forms of w' and of x, the Hamming weight $||x'||_1$ equals the depth D of

the edge $\tilde{e}_{\ell^*}$. The debate circuit expects $\tilde{e}_{\ell^*}$ to determine the q^{th} edge of the path specifying e: namely, it expects that $D - |s_{\ell^*}| < q$ while $D \geq q$. The circuit $\widetilde{\Psi}_R$ tests this condition, outputting 1 if the test fails. If the test succeeds, the q^{th} bit of the path specifying $\tilde{e}_i$ can be determined from the values D and s_{ℓ^*}, and compared to h_q. $\widetilde{\Psi}_R$ outputs 0 if these bits differ, and outputs 1 otherwise.

Assuming honest play by Player 1 in Phases 1 and 2 (which we've argued is an optimal strategy), Player 0 can win exactly if he can give an index q for which h_q differs from the q^{th} step specifying $\tilde{e}_i$. This is possible iff $edge(\emptyset; h) \neq \tilde{e}_i$. Thus $\widetilde{\Psi}_R$ has the desired behavior. Also, by construction it has $O(1)$ debate-blocks, and is of $O(R)$ size as needed. □

The next two lemmas relate the mappings $\widetilde{E}'$ and E'.

Lemma 18. *Fix $\gamma \in \Gamma^{\leq R}$, and suppose that i is an effective index for γ (that is, $e_i \neq \perp$). Then, $\tilde{e}_i = e_i$.*

Proof. The proof is by induction on i. The base case is immediate from the definitions. Assume the statement holds for all $i' < i$, and suppose $e_i \neq \perp$. Then $z_i = (r_i, s_i)$ where $r_i < i$ and $e_{r_i} \neq \perp$. By inductive assumption, $\tilde{e}_{r_i} = e_{r_i}$. Then from the definitions of $e_i, \tilde{e}_i$ it follows that the two are equal, as claimed. □

Lemma 19. *Fix $\gamma \in \Gamma^k$, where $k \leq R$, and let $I \subseteq [k]$. We have $I = \{i : i$ is an effective index for $\gamma\}$ iff the following two conditions hold:*

1. *For every $i \in I$, we have:*
   - *(a) i is a viable index for γ;*
   - *(b) $z_i = (r_i, s_i)$ satisfies $r_i < i$, and either $r_i = 0$ or $r_i \in I$;*
   - *(c) there is no $i' \in I$ with $i' < i$, for which $\tilde{e}_{i'}$ has the same parent vertex in $\mathcal{T}$ as $\tilde{e}_i$.*
2. *For every $j \notin I$, at least one of conditions (a), (b), (c) above fails for $i := j$.*

Proof. First, suppose that indeed $I = \{i : i$ is an effective index for $\gamma\}$, and consider any $i \in I$. Conditions 1.(a) and 1.(b) follow immediately from the definitions. Also, given $i \in I$ with $i' < i$, Lemma 18 tells us that $\tilde{e}_i = e_i$ and $\tilde{e}_{i'} = e_{i'}$, so condition 1.(c) must hold as well. So condition 1 is true for I.

Now consider any $j \notin I$, which by assumption is a non-effective index (for γ, implicit from now on); we show one of conditions (a), (b), (c) must fail for $i := j$. Suppose that (a), (b) hold for j: that is, j is viable, and $z_j = (r_j, s_j)$ where $r_j < j$ and either $r_j = 0$ or $r_j \in I$ is effective.

Assume $r_j \in I$; the case $r_j = 0$ is handled similarly. Now for $e_j = \perp$ to hold (as it does), it must be the case that e'_j is defined ($\neq \perp$), but shares a parent vertex with some $e_{j'} \neq \perp$ for which $j' < j$. (For otherwise, we would have $e_j = e'_j$.) Then j' is an effective index, and by Lemma 18, $\tilde{e}_{j'} = e_{j'}$.

Consult the definition of $\tilde{e}_j$: it is obtained by starting from the child vertex of $\tilde{e}_{r_j} (= e_{r_j}$, since r_j is an effective index), and taking the steps described by s_j. Now, e'_j is obtained by taking the steps s_i starting from e_{r_j}, so in fact $e'_j = \tilde{e}_j$. Thus $\tilde{e}_j$ shares a common parent with $\tilde{e}_{j'}$, so condition (c) fails for j. Thus, condition 2 is true for I.

For the converse direction, suppose $I \neq \{i : i \text{ is an effective index for } \gamma\}$, and let $\ell \in [k]$ be the *minimal* index for which the equality fails. Let us first suppose that ℓ is an effective index but $\ell \notin I$. Then by definition, ℓ is viable (so, condition (a) holds for $i := \ell$). Let $z_\ell = (r_\ell, s_\ell)$. If $r_\ell = 0$, then condition (b) holds for ℓ. On the other hand, if $r_\ell > 0$, then r_ℓ must be an effective index, with $r_\ell < \ell$. By the minimality of ℓ, it must be the case that $r_\ell \in I$. Thus condition (b) holds for ℓ whether or not $r_\ell = 0$.

The fact $e_\ell \neq \perp$ implies that e_ℓ has no common parent with $e_{\ell'}$, for any effective index $\ell' < \ell$. But by Lemma 18, $e_\ell = \tilde{e}_\ell$, and so for any effective $\ell' < \ell$, $\tilde{e}_\ell$ has no common parent with $e_{\ell'} = \tilde{e}_{\ell'}$. Thus condition 2.(c) holds for ℓ. Since each of conditions (a)-(c) hold for $i := \ell$ but $\ell \notin I$, condition 2 fails for I.

Suppose now that ℓ, the minimal index chosen earlier, is in I but is not effective. If condition (a) or (b) fails for $i := \ell$, we are done, so assume that ℓ is viable, $r_\ell < \ell$, and either $r_\ell = 0$ or $r_\ell \in I$. By minimality of ℓ, $r_\ell < \ell$ is an effective index: e_{r_ℓ} is defined. Then for $e_\ell = \perp$ to hold (as it does), e'_ℓ must share a parent vertex with some edge $e_{\ell'}$ where $\ell' < \ell$. By Lemma 18, we have $e_{\ell'} = \tilde{e}_{\ell'}$; also, arguing as in a previous case, we find that $e'_\ell = \tilde{e}_\ell$. Thus $\tilde{e}_\ell$ shares a parent vertex with $\tilde{e}_{\ell'}$, and condition (c) fails for $\ell \in I$. Thus condition 1 fails for I. This completes the proof. □

We can now prove Lemma 15.

Proof of Lemma 15. The debate proceeds in phases.

Phase 1: The circuit computes $k \leq R$, the length of γ. Player 1 is asked to give a set $I \subseteq [k]$ which he claims is the set of effective indices for γ. The set I is specified by a sequence $x \in \{0,1\}^k$.

Now we use Lemma 19 to debate Player 1's claim. Player 0 is asked to give, either an $i \in I$ for which one of conditions (a)-(c) in Lemma 19 fails, or an $i \notin I$ for which (a)-(c) all hold. By Lemma 19, such an i can be found iff I is not the set of effective indices for γ.

For the index i chosen by Player 0, it can be checked whether the index lies in I or not using Lemma 12, part (i). First suppose that $i \in I$. Then the players debate in turn whether each of conditions (a)-(c) in Lemma 19 hold for i, with with Player 1 attempting to argue that each of these conditions do hold. Call these three sub-debates Debate (a), Debate (b), and Debate (c). We will explain how Debates (a)-(c) are to be implemented shortly. Ψ_R outputs 0 if Player 1 loses *any* of Debates (a)-(c). Otherwise the debate moves on to Phase 2 below.

If, on the other hand, $i \notin I$, then the players again debate whether each of conditions (a)-(c) hold for i, this time with Player 1 attempting to argue that each of these conditions do *not* hold. Ψ_R outputs 0 if Player 1 loses *all* of these (role-reversed) Debates (a)-(c). Otherwise the debate moves on to Phase 2.

As for the implementations of Debates (a)-(c), a debate subcircuit of size $O(R)$ for Debates (a) and (b) can be defined using Lemma 16. For Debate (c), suppose that for $b \in \{0,1\}$, Player b wishes to argue that condition (c) in Lemma 19 fails for i. To do so, he is asked to name an $i' \in I$ with $i' < i$, for which he claims $\tilde{e}_{i'}, \tilde{e}_i$ have the same parent vertex. He then gives representations g, g' for which he claims

$$\tilde{e}_i = edge(\emptyset; g), \quad \tilde{e}_{i'} = edge(\emptyset; g'). \tag{5}$$

Ψ_R may easily verify in $O(R)$ size that $i' \in I$, $i' < i$, and that the edges defined by g, g' share a parent. The claims in Eq. (5) above may then be debated in $O(R)$ size, using Lemma 17. Thus Phase 1 can be implemented in $O(R)$ size.

Lemma 19 implies that, under optimal play by Player 0, Player 1 can reach Phase 2 iff he sets $I = I(x)$ as the set of all effective indices for γ. In Phase 2 we will assume that he does so.

Phase 2: Player 1 specifies an index $i^* \in I$ (this membership claim can be easily tested, so assume it holds). He claims that the edge e_{i^*}, determined by γ, satisfies

$$\tilde{e}_{i^*} = edge(\emptyset; h). \tag{6}$$

The claim that Eq. (6) holds is then debated using the circuit $\widetilde{\Psi}_R$ from Lemma 17. Whoever wins that sub-debate is named as the final winner by Ψ_R.

Since I equals the effective indices for γ (by our assumption in Phase 2), Lemma 18 tells us that $e_i = \tilde{e}_i$ for $i \in I$. Thus an $i^* \in I$ satisfying Eq. (6) can be found iff $edge(\emptyset; h) \in E'(\gamma)$. It follows that the debate defined by Ψ_R is winnable by Player 1 iff $edge(\emptyset; h) \in E'(\gamma)$. Finally, Ψ_R has $O(1)$ debate-blocks and is of $O(R)$ size, by construction. This proves Lemma 15. □

6.4 Characterizing faithful behavior in the Braverman-Rao protocol

Our next lemma will give a useful alternative characterization of faithful behavior in the protocol $\mathcal{P}_{BR,\varepsilon}$ of Section 4.8. We focus on characterizing faithful behavior for Alice (an analogous characterization holds for Bob, but we don't need it). First we need some notation. In that protocol, we defined the set A_i in terms of the sequences $a_1, \ldots, a_{i-1}$ and $\beta'_1, \ldots, \beta'_{i-1}$; let

$$A_i = \mathcal{A}_i\left(a_1, \ldots, a_{i-1}, \beta'_1, \ldots, \beta'_{i-1}\right)$$

denote the rule defining A_i in Eq. (1). This rule implicitly depends on the choice of tree code $C_{(r)}$ used (but is well-defined for any $r \in \mathbb{F}_p^R$). Note, the rule $\mathcal{A}_i$ does not depend on Alice's input X. Note also that we can apply the rule $\mathcal{A}_i$ to arbitrary sequences from $\Gamma^{i-1} \times \Sigma^{i-1}$; the resulting set $A_i \subseteq \mathcal{X} \cup \mathcal{Y}$ will always be consistent, since the mapping E' always outputs consistent sets.

We also defined a_i in terms of $A_i, a_1, \ldots, a_{i-1}$, *and* the input X. An important observation, however, is that the dependence on X is very limited. Namely, given A_i, the rule defining a_i depends only on one question about X: is $v(A_i)$ distinct from $v(A_i \cup X)$, and if so, does X extend the path in A_i ending at $v(A_i)$ by a left or a right outgoing edge?

For a vertex v of $\mathcal{T}$, define

$$\boldsymbol{\theta}(v, X) := \begin{cases} \perp & \text{if } X \text{ contains no outgoing edges of } v, \\ 0 & \text{if } X \text{ contains a 0-outgoing edge of } v, \\ 1 & \text{if } X \text{ contains a 1-outgoing edge of } v, \end{cases}$$

and note the definition is consistent since X is a consistent set. Then, we can express the definition of a_i in the protocol of Section 4.8 as a rule of form

$$a_i = \mathbf{a}_i\left(A_i, a_1, \ldots, a_{i-1}, \boldsymbol{\theta}(v(A_i), X)\right).$$

Formally, $\mathbf{a}_i = \mathbf{a}_i\left(C, c_1, \ldots, c_{i-1}, \theta\right)$ takes as inputs any consistent subset $C \subset \mathcal{X} \cup \mathcal{Y}$, any sequence $c_1, \ldots, c_{i-1} \in \Gamma^{i-1}$, and a $\theta \in \{0, 1, \perp\}$. Modifying the protocol statement appropriately, the output of $\mathbf{a}_i$ is defined as follows:

1. If $\theta = \perp$, let $\mathbf{a}_i :=$ "$\emptyset$".
2. Else, if $v(C)$ has outgoing edges from $\mathcal{X}$, let e be the θ-outgoing edge of $v(C)$. Now inspect the form of $(c_1, \ldots, c_{i-1})$. Suppose it is of form $(u, <, w)$ where u is of some length $j - 1 \geq 0$, and such that:
   (a) $w \in \{0,1\}^*$ is a (possibly empty) prefix of a string $w' = desc((\ell, s))$, for some $(\ell, s) \in \Pi$; **and**,
   (b) **Either** $v(C)$ is the root of $\mathcal{T}$, $\ell = j$, and $edge(\emptyset; s) = e$;
   or, e is a descendant of $e_{j-\ell} = e_{j-\ell}(c_1, \ldots, c_{j-\ell})$, with $e_{j-\ell} \in \mathcal{X}$ and $e = edge(e_{j-\ell}; s)$.

   In this case, set $\mathbf{a}_i$ so as to extend w according to w' (or, if $w = w'$, let $\mathbf{a}_i :=$ ">").
3. Else, set $\mathbf{a}_i :=$ "<".

Our next lemma uses the mappings $\mathcal{A}_i$ and $\mathbf{a}_i$ to characterize faithful behavior for Alice in $\mathcal{P}_{BR,\varepsilon}$.

Lemma 20. *Let $\mathcal{P}_{BR,\varepsilon} = (\mathcal{P}_A, \mathcal{P}_B)$. The transcript (α, β) is $\mathcal{P}_A$-faithful, if and only if there exists a pair of sequences*

$$\tilde{a} = (\tilde{a}_1, \ldots, \tilde{a}_R) \in \Gamma^R, \quad (\theta_1, \ldots, \theta_R) \in \{0, 1, \perp\}^R,$$

such that the following conditions hold for all $i \in [R]$:

1. *$\alpha_i = C_{(r)}(\tilde{a}_1, \ldots, \tilde{a}_i)$;*
2. *We have*
$$\tilde{a}_i = \mathbf{a}_i\left(\tilde{A}_i, \tilde{a}_1, \ldots, \tilde{a}_{i-1}, \theta_i\right),$$
where we define
$$\tilde{A}_i := \mathcal{A}_i\left(\tilde{a}_1, \ldots, \tilde{a}_{i-1}, \beta_1, \ldots, \beta_{i-1}\right);$$
3. *$\theta_i = \perp$ iff $v(\tilde{A}_i)$ has no outgoing edges from $\mathcal{X}$;*
4. *For all $j \in [R]$ for which $v(\tilde{A}_i) = v(\tilde{A}_j)$, we have $\theta_i = \theta_j$.*

Moreover, if $\tilde{a}$ and $(\theta_1, \ldots, \theta_R)$ exist satisfying 1-4 above, then (α, β) is $\mathcal{P}_A$-faithful with respect to some valid input X for which the sequence $a = (a_1, \ldots, a_R)$ defined by $\mathcal{P}_A$ (with respect to X and the sequence β of received messages from Bob) satisfies $a = \tilde{a}$.

Proof. One direction is easy. Suppose (α, β) is $\mathcal{P}_A$-faithful with respect to some input $X \subset \mathcal{X}$, with the associated sequences $(a_1, \ldots, a_R, A_1, \ldots, A_R)$. Then for $i \in [R]$, let $\tilde{a}_i := a_i$, and let $\theta_i := \boldsymbol{\theta}(v(A_i), X)$. Conditions 1-4 above follow immediately from the definition of $\mathcal{P}_A$.

Next, suppose that for the transcript (α, β), there exist sequences $(\tilde{a}_1, \ldots, \tilde{a}_R), (\theta_1, \ldots, \theta_R)$ satisfying conditions 1-4 above; we will define a maximal consistent subset $X \subset \mathcal{X}$ and argue that (α, β) is $\mathcal{P}_A$-faithful with respect to X. First, define

$$V_{term} := \{v(\tilde{A}_i) : i \in [R]\} \subseteq V(\mathcal{T})$$

as the set of terminal vertices on the maximal paths from the root within $\tilde{A}_1, \ldots, \tilde{A}_R$. Define X^* as the union of all b-outgoing edges, for every $v = v(\tilde{A}_i) \in V_{term}$ with $\theta_i = b$ (for $b \in \{0, 1\}$). By condition 3, $X^* \subset \mathcal{X}$ and every vertex in $\widetilde{V}$ with outgoing edges in $\mathcal{X}$ also has an outgoing edge in X^*. Also, by condition 4, X^* is consistent. Next, extend X^* arbitrarily to a *maximal* consistent set X satisfying $X^* \subseteq X \subset \mathcal{X}$.

Let $a_1, \ldots, a_R, A_1, \ldots, A_R$ be the sequences defined by $\mathcal{P}_A$ relative to input X and the received messages $(\beta_1, \ldots, \beta_R)$ (which need not be $\mathcal{P}_B$-faithful). We claim, and show by induction on i, that $(\tilde{a}_i, \tilde{A}_i) = (a_i, A_i)$ for each $i \in [R]$. It will follow from condition 1 of our assumptions that (α, β) is $\mathcal{P}_A$-faithful for X, proving the lemma.

For $i = 1$, by inspection we have $\tilde{A}_1 = \emptyset = A_1$ and $\tilde{a}_1 =$ "$<$" $= a_1$ (using condition 2 of our assumptions). Now for $i > 1$, assume $(\tilde{a}_1, \ldots, \tilde{a}_{i-1}) = (a_1, \ldots, a_{i-1})$. Then, $\tilde{A}_i = A_i$ is immediate from the definition of $\tilde{A}_i$.

We have $v(A_i) = v(A_i \cup X)$ iff $v(A_i)$ has no outgoing edges in $\mathcal{X}$. This holds iff $v(\tilde{A}_i)$ has no such edges either, and by condition 3, this occurs iff $\theta_i = \perp$. Thus, case 1 from $\mathcal{P}_A$ applies in defining a_i iff case 1 applies in defining $\tilde{a}_i$, and $a_i = \tilde{a}_i =$ "$\emptyset$" in this outcome.

If $v(A_i) \neq v(A_i \cup X)$, then by construction of X, $\theta_i \in \{0, 1\}$ identifies the outgoing edge e in X from $v(\tilde{A}_i) = v(A_i)$; that is, $\theta_i = \boldsymbol{\theta}(v(A_i), X)$. Thus in this case,

$$\tilde{a}_i = \mathbf{a}_i(\tilde{A}_i, \tilde{a}_1, \ldots, \tilde{a}_{i-1}, \theta(v(A_i), X)) = \mathbf{a}_i(A_i, a_1, \ldots, a_{i-1}, \theta(v(A_i), X)) = a_i.$$

This extends the induction to i, completing the proof. □

Lemma 21. *There exists a uniform debate circuit Ξ_R, that takes as its start-variables sequences*

$$r \in \mathbb{F}_p^R, \quad \beta \in \mathbb{F}_p^R, \quad \tilde{a} \in \Gamma^R, \quad \theta \in \{0, 1, \perp\}^R,$$

and a value $i \in [R]$. We have $\mathrm{Val}(\mathcal{G}_{\Xi_R}[r, \beta, \tilde{a}, \theta, i]) = 1$ *iff*

$$\tilde{a}_i = \mathbf{a}_i\left(\tilde{A}_i, \tilde{a}_1, \ldots, \tilde{a}_{i-1}, \theta_i\right),$$

where as in Lemma 20, $\tilde{A}_i := \mathcal{A}_i(\tilde{a}_1, \ldots, \tilde{a}_{i-1}, \beta_1, \ldots, \beta_{i-1})$, and where $\mathcal{P}_{BR,\varepsilon}$ is defined relative to $C_{(r)}$. The debate circuit Ξ_R has $O(1)$ debate-blocks and is of total size $O(R)$.

Proof. If $\theta_i = \perp$, the task is trivial: Ξ_R simply needs to check that $a_i =$ "$\emptyset$". So assume $\theta_i \in \{0, 1\}$. Similarly, the task is trivial if $i = 1$, so assume $i > 1$. First, Player 1 is asked to provide a sequence $u \in \Gamma^{i-1}$ for which he claims

$$u = D_{(r)}^{lex}(\beta_1, \ldots, \beta_{i-1}).$$

This claim can be debated using the debate circuit Λ_R from Lemma 14. We let Ξ_R output 0 if Player 1 loses this debate; thus, we may assume in what follows that Player 1 supplies the true value of $D^{lex}_{(r)}(\beta_1, \ldots, \beta_{i-1})$.

Next, the circuit Ξ_R asks Player 1 to provide a description of a vertex v, in the form of a path $h \in \{0,1\}^{\leq T}$ such that $v = vertex(\emptyset; h)$. The implicit claim by Player 1 is that $v = v(\tilde{A}_i)$. Player 0 is then given an opportunity to dispute the value v provided. There are two ways v could fail to equal $v(\tilde{A}_i)$: either some edge e along the path h does not lie in $\tilde{A}_i$, *or* some child-edge e of v lies in $\tilde{A}_i$. Player 0 claims one of these two options hold; in either case, he specifies the relevant edge $e = edge(\emptyset; h')$ in the form of a path h' that is either a subpath of h, or extends h by a single step. (That h has this form can easily be checked by an $O(R)$-sized circuit.) Using the definition

$$\tilde{A}_i = \left(E'\left(\tilde{a}_1, \ldots, \tilde{a}_{i-1}\right)\right) \cap \mathcal{X}) \cup \left(E'(D^{lex}_{(r)}(\beta_1, \ldots, \beta_{i-1})) \cap \mathcal{Y}\right),$$

and the assumed fact that Player 1 has provided the true value of $D^{lex}_{(r)}(\beta_1, \ldots, \beta_{i-1})$, the two players may debate whether $edge(\emptyset; h') \in \tilde{A}_i$ using the debate circuit Ψ_R from Lemma 15. (If h' is a subpath of h, Player 0 must argue $e = edge(\emptyset; h') \notin \tilde{A}_i$; if h' extends h, Player 0 must argue $e \in \tilde{A}_i$. Depending on which case we are in, we assign debater roles in Ψ_R accordingly.)

If Player 0 successfully defends the claim that $v \neq v(\tilde{A}_i)$, then we let Ξ_R output 0. Thus we may assume in what follows that Player 1 provides the true value $v = v(\tilde{A}_i)$, and successfully defends his claim.

Recall that $\theta_i \in \{0,1\}$, so Player 1 must argue that case 2 or 3 in the definition of $\mathbf{a}_i$ holds when we evaluate $\mathbf{a}_i\left(\tilde{A}_i, \tilde{a}_1, \ldots, \tilde{a}_{i-1}, \theta_i\right)$. If $\tilde{a}_i \neq$ "$<$", Player 1 must argue that case 2 holds; otherwise Player 1 must argue that case 3 holds, and Player 0 is given an opportunity to argue that, on the contrary, case 2 holds. In either case, one of the two players must argue that case 2 holds, so we will just describe the alternative where Player 1 must argue that case 2 holds (the other alternative is handled similarly with roles reversed).

To argue that case 2 holds in the definition of $\mathbf{a}_i$, and that $\mathbf{a}_i = \tilde{a}_i$, Player 1 is asked to provide the strings $w, w' \in \{0,1\}^*$ from case 2. These strings can be restricted to have length $O(\log R)$, since the descriptions produced by *desc* are of length $O(\log R)$, so the claim that w is a prefix of w' can be checked directly by Ξ_R, and the claim that $\tilde{a}_1, \ldots, \tilde{a}_{i-1}$ is of form $(u, <, w)$ can be easily debated. Ξ_R outputs 0 if Player 1 loses either of these tests, so we may assume the Player 1 provides w honestly and a w' extending w.

Next, Ξ_R checks whether $w' = desc((\ell, s))$ for some $(\ell, s) \in \Pi$, and if so stores the values ℓ, s. This can be done efficiently since w' is of length $O(\log R)$ and we're using a "reasonable" encoding *desc*. So we may assume the w' Player 1 provides is such a valid description, if this is possible.

Now Player 1 must argue that condition (b) in case 2 of the definition of $\mathbf{a}_i$ holds for w, w'. In the case where $v(\tilde{A}_i)$ is the root of $\mathcal{T}$, it is easy to decide whether condition (b) holds. Otherwise, Player 1 must argue that e, the θ_i-outgoing edge of $v(\tilde{A}_i)$, is a descendant of $e_{j-\ell} = e_{j-\ell}(\tilde{a}_1, \ldots, \tilde{a}_{i-1})$, with $e_{j-\ell} \in \mathcal{X}$ and $e = edge(e_{j-\ell}; s)$. This can be debated using our circuit Ψ_R from Lemma 15. If Player 1 loses this debate, Ξ_R will immediately output 0; thus, Player 1 can pass this part of the debate iff case 2 of the definition of $\mathbf{a}_i$ holds, for his

chosen pair w, w'. We may assume that if case 2 holds, Player 1 chooses w, w' correctly and defends his choice successfully.

Finally, we let Ξ_R output 1 exactly if $\tilde{a}_i$ extends w according to w' (or, in the case $w = w'$, Ξ_R outputs 1 iff $\tilde{a}_i =$ ">"). Thus, in the event that Player 1 must argue (truthfully) that case 2 in the definition of $\mathbf{a}_i$ holds, then Player 1 can win iff $\tilde{a}_i = \mathbf{a}_i\left(\tilde{A}_i, \tilde{a}_1, \ldots, \tilde{a}_{i-1}, \theta_i\right)$. On the other hand, Player 1 cannot win if he claims falsely that case 2 in the definition of $\mathbf{a}_i$ holds. As mentioned earlier, the case where Player 0 is arguing that case 2 holds is handled similarly. This proves correctness of Ξ_R. Also, by construction, Ξ_R is of size $O(R)$ and has $O(1)$ debate-blocks. □

6.5 Proof of Lemma 11

We are finally ready to prove our main technical lemma from Section 5.1, Lemma 11.

Proof of Lemma 11. The debate circuit $\Phi^{BR} = \Phi_T^{BR,\varepsilon}$ operates in three phases, as follows:

Phase 1: In this phase, the players use the debate circuits to debate whether $C_{(r)} : \Gamma^{\leq R} \to \mathbb{F}_p$ is a tree code of distance $\geq 1-\varepsilon$. For this, Player 0 is asked to supply a pair u, v of some equal length $k \leq R$. The implicit claim is that u, v form a counterexample to the tree code distance property (defined in Section 4.6).

Player 0 is also asked to supply the index $i \leq k$ on which he claims u, v first differ. This claim can easily be debated (by a circuit of size $O(R)$) using Lemma 12, part (i). If Player 0 lies and is caught, Φ^{BR} will immediately output 1; thus, we may assume Player 0 truthfully supplies the correct value i. From this, the circuit can easily compute the quantity $\ell(u, v)$ from Section 4.6.

Next, Player 1 supplies w, w' for which he claims $\overline{C}(u) = w, \overline{C}(v) = w'$. The players use the debate circuit $\Phi = \Phi_R$ from Lemma 13 to debate this claim. If Player 1 loses this debate, Φ^{BR} will immediately output 0, so we may assume the values w, w' are correct and defended successfully by Player 1. Φ^{BR} then uses Lemma 12, part (iii) to compute $\Delta(w, w')$. If $\Delta(w, w') < (1-\varepsilon)\ell(u, v)$, then Φ^{BR} immediately outputs 0; otherwise we move on to Phase 2. (Φ^{BR} can store the dyadic rational value ε for this use.) By construction, Player 1 can avoid losing in Phase 1 iff $C_{(r)}$ is a tree code of distance $1-\varepsilon$.

Phase 2: In this phase, the players debate whether (α, β) is $\mathcal{P}_A = \mathcal{P}_A[r]$-faithful (we suppress the dependence on r in our notation). To this end, Player 1 first provides sequences $\tilde{a} = (\tilde{a}_1, \ldots, \tilde{a}_R) \in \Gamma^R$, $\theta = (\theta_1, \ldots, \theta_R) \in \Gamma^R$. His first implicit claim, to be debated in Phase 2, is that conditions 1-4 in Lemma 20 hold for $\tilde{a}, \theta$ and for all $i \in [R]$. His second claim, to be debated in Phase 3, is that for the input X guaranteed to exist for $\tilde{a}$ by Lemma 20, we have

$$Eval_A(X, \alpha, \beta) = edge(\emptyset; h).$$

First, Player 0 is given the opportunity to give an $i_0 \in [R]$ and a $c \in \{1, 2, 3, 4\}$ for which he claims condition c in Lemma 20 fails for $i = i_0$.

If $c = 1$, Player 0's claim can be debated using the circuit Φ_R from Lemma 13. If $c = 2$, the claim can be debated using the circuit Ξ_R from Lemma 21. If $c \in \{3, 4\}$, the claim is

easily debated once we can establish the value of $v(\tilde{A}_i)$. We have already described (in the proof of Lemma 21) how the players may debate to determine $v(\tilde{A}_i)$.

If Player 0 successfully defends his claim specified by (i_0, c), Φ^{BR} immediately outputs 0; otherwise we move on to Phase 3. Thus under optimal play, Player 1 can pass through Phase 2 without losing iff there is a pair $\tilde{a}, \theta$ for which conditions 1-4 of Lemma 20 hold for all i. By that Lemma, such a pair exists iff (α, β) is $\mathcal{P}_A$-faithful with respect to some input X for which $a = \tilde{a}$.

Phase 3: In this phase, the players debate whether $Eval_A(X, \alpha, \beta) = vertex(\emptyset; h)$ for the X guaranteed to exist for $\tilde{a}$ by Lemma 20. Recall that $Eval_A(X, \alpha, \beta) = v(A_R)$, where A_R is as defined by $(a_1, \ldots, a_R, \beta_1, \ldots, \beta_R) = (\tilde{a}, \beta)$ in Eq. (1). We showed in the proof of Lemma 21 how the players can debate to determine $v(A_R)$. We reapply that debate; Player 1 is named the winner by Φ^{BR} iff Player 1 successfully defends the claim that $v(A_R) = vertex(\emptyset; h)$.

Thus, Player 1 can win the debate Φ^{BR} iff $C_{(r)}$ is a tree code of distance $\geq 1 - \varepsilon$ and (α, β) is $\mathcal{P}_A$-faithful with respect to some input X for which $Eval_A(X, \alpha, \beta) = vertex(\emptyset; h)$, as desired. Finally, Φ^{BR} has $O(1)$ debate-blocks by construction, and is of size $O(R)$. □

7 Conclusion

There are several possible directions for future work:

1. Can our probabilistically checkable debate systems be made even more efficient? Developing good error-correcting codes decodable by *linear*-sized circuits (improving on the $O(n \log n)$-sized circuits from [Spi96]) would shave a log factor from our debate-string lengths. If we could do this, then our PCDSs could be made to essentially match the efficiency of the best PCPPs.

2. In this paper we did not attempt to minimize the increase in the number of *turns* in our debate transformation. Our approach gives no improvement over [CFLS95] in this respect: both approaches increase the number of turns by a constant factor, *if* the starting debate has strictly-alternating turns, each consisting of a single bit. If not, the number of turns may increase by a much larger amount.

   To our knowledge, there also has been no in-depth study of the number of rounds required for error-resilient communication, when the communication protocol to be simulated lasts a bounded number of rounds (with several bits transmitted per round). Can we make communication error-resilient, while increasing each of the number of rounds and the total communication by only a constant (or at least slowly-growing) factor? The challenging case seems to be when rounds are of variable bitlength. Understanding this question would clarify the situation for PCDSs, and would be of interest in its own right.

3. Recall that our PCDSs for L are of bitlength nearly-linear in $s(n)$, the *circuit size* of the verifier for an ordinary debate system for L. We left open whether the PCDS

bitlength could be polynomial in the *bitlength* of the original debate. Can we derive unlikely consequences of this? To explore this, one might try to work by analogy with Fortnow and Santhanam's results on infeasibility of succinct PCPs for SAT [FS11] (see Section 1.4).

4. As mentioned earlier, the same authors who first built PCDSs in [CFLS95] also showed in [CFLS97] that *interactive proofs* (i.e., debates between a maximizing Player 1 and a completely random Player 0) can also be made probabilistically checkable. It would be very interesting to know whether this reduction can be carried out with efficiency comparable to our reduction for ordinary debate systems in this paper. This would yield improved conditional hardness statements for the complexity of approximating stochastic sequential optimization problems on CSPs. The difficulty in applying our methods is that we appear to have no effective way to make the random player "follow" the Braverman-Rao protocol.

References

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[BFL90] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. In *FOCS*, pages 16–25, 1990.

[BR10] Mark Braverman and Anup Rao. Towards coding for maximum errors in interactive communication. *Electronic Colloquium on Computational Complexity (ECCC)*, TR10-166, 2010. To appear in STOC 2011.

[Bra11] Mark Braverman. Towards deterministic tree code constructions. *Electronic Colloquium on Computational Complexity (ECCC)*, TR11-064, 2011.

[BSGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

[BSS06] Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Struct. Algorithms*, 28(4):387–402, 2006.

[CFLS95] Anne Condon, Joan Feigenbaum, Carsten Lund, and Peter W. Shor. Probabilistically checkable debate systems and nonapproximability of PSPACE-hard functions. *Chicago J. Theor. Comput. Sci.*, 1995, 1995.

[CFLS97] Anne Condon, Joan Feigenbaum, Carsten Lund, and Peter W. Shor. Random debaters and the hardness of approximating stochastic functions. *SIAM J. Comput.*, 26(2):369–400, 1997.

[CKS81] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[CS76] Ashok K. Chandra and Larry J. Stockmeyer. Alternation. In *FOCS*, pages 98–108, 1976.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[DR06] Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM J. Comput.*, 36(4):975–1024, 2006.

[Dru10] Andrew Drucker. A PCP characterization of AM. *Electronic Colloquium on Computational Complexity (ECCC)*, TR10-019, 2010. To appear in ICALP 2011.

[For05] Lance Fortnow. Beyond NP: the work and legacy of Larry Stockmeyer. In *STOC*, pages 120–127, 2005.

[FS11] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011.

[GS11] Ran Gelles and Amit Sahai. Potent tree codes and their applications: Coding for interactive communication, revisited. *ArXiv e-prints*, April 2011, 1104.0739.

[KL94] Ker-I Ko and Chih-Long Lin. Non-approximability in the polynomial-time hierarchy. Technical Report 94-2, Dept. of Computer Science, SUNY at Stony Brook, 1994.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *FOCS*, pages 2–10, 1990.

[Moi11] Ankur Moitra. Efficiently coding for interactive communication. *Electronic Colloquium on Computational Complexity (ECCC)*, TR11-042, 2011.

[Sch96] Leonard J. Schulman. Coding for interactive communication. *IEEE Trans. Inf. Theory*, 42(6):1745–1756, 1996.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time(preliminary report). In *STOC*, pages 1–9, 1973.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inf. Theory*, 42(6):1723–1731, 1996.

[Sto76] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976.

[Wil08] Ryan Williams. Non-linear time lower bound for (succinct) quantified Boolean formulas. *Electronic Colloquium on Computational Complexity (ECCC)*, TR08-076, 2008.

[Wra76] Celia Wrathall. Complete sets and the polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):23–33, 1976.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de