Two Query PCP with Sub-Constant Error

Dana Moshkovitz [*]   Ran Raz [†]

July 28, 2008

Abstract

We show that the $\mathcal{NP}$-Complete language 3SAT has a PCP verifier that makes two queries to a proof of almost-linear size and achieves sub-constant probability of error $o(1)$. The verifier performs only projection tests, meaning that the answer to the first query determines at most one accepting answer to the second query. Previously, by the parallel repetition theorem, there were PCP Theorems with two-query projection tests, but only (arbitrarily small) *constant* error and *polynomial* size [29]. There were also PCP Theorems with *sub-constant* error and *almost-linear* size, but a constant number of queries that is *larger* than 2 [26].

As a corollary, we obtain a host of new results. In particular, our theorem improves many of the hardness of approximation results that are proved using the parallel repetition theorem. A partial list includes the following:

1. 3SAT cannot be efficiently approximated to within a factor of $\frac{7}{8} + o(1)$, unless $\mathcal{P} = \mathcal{NP}$. This holds even under almost-linear reductions. Previously, the best known $\mathcal{NP}$-hardness factor was $\frac{7}{8} + \varepsilon$ for any constant $\varepsilon > 0$, under polynomial reductions (Håstad, [18]).

2. 3LIN cannot be efficiently approximated to within a factor of $\frac{1}{2} + o(1)$, unless $\mathcal{P} = \mathcal{NP}$. This holds even under almost-linear reductions. Previously, the best known $\mathcal{NP}$-hardness factor was $\frac{1}{2} + \varepsilon$ for any constant $\varepsilon > 0$, under polynomial reductions (Håstad, [18]).

3. A PCP Theorem with amortized query complexity $1 + o(1)$ and amortized free bit complexity $o(1)$. Previously, the best known amortized query complexity and free bit complexity were $1 + \varepsilon$ and ε, respectively, for any constant $\varepsilon > 0$ (Samorodnitsky and Trevisan, [32]).

One of the new ideas that we use is a new technique for doing the *composition* step in the (classical) proof of the PCP Theorem, without increasing the number of queries to the proof. We formalize this as a composition of new objects that we call *Locally Decode/Reject Codes* (LDRC). The notion of LDRC was implicit in several previous works, and we make it explicit in this work. We believe that the formulation of LDRCs and their construction are of independent interest.

1 Introduction

1.1 Probabilistic Checking of Proofs

The PCP Theorem [2, 1] states that any mathematical proof can be converted to a form that can be checked *probabilistically* by reading only a *constant* number of places in the proof. Moreover, the check can be performed by an *efficient* verifier. If the mathematical theorem, supposedly being proven, is correct, then there exists a proof in the new form that the verifier always (or almost always) accepts. On the other hand, if the mathematical theorem is false, then no matter which proof is provided, the verifier rejects with at least some constant probability. Note that soundness holds even though the verifier queries the proof only in a constant number of places.

A PCP verifier has several important parameters (ideally, we would like all parameters, except the completeness, to be as small as possible):

1. **Completeness** (c) : The minimal probability that the verifier accepts a correct proof. An almost perfect completeness $c \approx 1$ is usually required. In most cases, a perfect completeness $c = 1$ can be obtained.

2. **Soundness (or, Error)** (ε) : The maximal probability that the verifier accepts a proof for an incorrect theorem. An error of at most $1 - \delta$ for some constant $\delta > 0$ is usually required. In some cases, a sub-constant error $\varepsilon = o(1)$ can be obtained.

3. **Queries** (q) : The number of queries to the proof. A constant number of queries $q = O(1)$ is usually required. In some cases, $q = 2$ can be obtained.

4. **Size** (m) : The size of a proof in the new form, with respect to the size n of the original proof. A polynomial size $m = \mathsf{poly}(n)$ is usually required. In some cases, an almost linear size $m = n^{1+o(1)}$ can be obtained.

5. **Randomness** (r) : The number of random bits used by the verifier. The randomness upper bounds the size by $m \leq 2^r \cdot q$. Thus, $r = O(\log n)$ corresponds to polynomial size and $r = (1 + o(1)) \cdot \log n$ corresponds to almost linear size.

6. **Alphabet** (Σ) : The alphabet used for the proof in the new form. It is sometimes more convenient to consider the **answer-size** $\log |\Sigma|$ (i.e., the number of bits required to represent an alphabet symbol), rather than the alphabet size $|\Sigma|$ itself. An alphabet size of at most $\mathsf{poly}(n)$ (i.e., answer-size of $O(\log n)$) is usually required. In some cases, answer-size of $O(\log \frac{1}{\varepsilon})$ can be obtained.

We denote by $PCP_{c,s}[r, q]_\Sigma$ the class of languages that have a PCP verifier with completeness c, soundness s, randomness r, and q queries to a proof over alphabet Σ. (When we omit Σ, it should be understood that $\Sigma = \{0, 1\}$.)

We think of all the parameters as functions of n.

1.2 Hardness of Approximation and Two-Query Projection Tests

Feige et al [15] discovered that there is a close and simple connection between PCP Theorems and hardness of approximation. The PCP Theorem can be formalized as stating that approximating the number of satisfiable clauses in a 3SAT formula to within some constant factor is $\mathcal{NP}$-hard. This formulation enabled a vast body of hardness of approximation results via further reductions.

A reduction can be viewed as proving a PCP Theorem in which the type of check corresponds to the problem to which one reduces. Consider, for instance, 3LIN, i.e., the problem of computing the maximal number of equations that can be satisfied simultaneously in a system of linear equations, where each equation is on three variables over $GF(2)$. To prove the hardness of approximating 3LIN, it is sufficient to prove a PCP Theorem in which the verifier's tests consist of querying three bits and comparing their XOR to predefined values (0 or 1).

For many optimization problems, this research direction produced hardness results that match (or almost match) the approximation factors obtained by the best existing algorithms. Thus, one is able to explain our lack of success in finding better efficient approximation algorithms. For other problems, tight hardness results are not known, or are known only under assumptions.

A generic type of tests that was discovered to be especially useful as a starting point for further reductions is a *two-query projection test*. In a system of two-query projection tests, the proof consists of two parts A and B. The verifier makes one query to the A part and one query to the B part. Upon seeing the answer to the A query, the verifier either immediately rejects, or it has a uniquely determined answer to the B query on which it accepts[1].

1.3 Existing PCP Theorems

The basic PCP Theorem that was proved by [2, 1] based on previous work [25, 5, 4, 15, 31] is:
Theorem 1 (Basic PCP, [2, 1]). $NP \subseteq PCP_{1,\frac{1}{2}}[O(\log n), O(1)]$.

One can convert Theorem 1 to the following equivalent formulations.
Theorem 2 (Equivalent formulations, [7]). *Theorem 1 is equivalent to each of the following:*

1. ***Low error:** For any $\varepsilon > 0$, $NP \subseteq PCP_{1,\varepsilon}[O(\log n), O(\log \frac{1}{\varepsilon})]$.*

2. ***Two-query projection tests:** There exist a constant $\delta > 0$ and an alphabet Σ of constant size, such that $NP \subseteq PCP_{1,1-\delta}[O(\log n), 2]_\Sigma$. Moreover, the PCP verifier makes two-query projection tests.*

The first item follows from sequential repetition of the basic PCP test (the repetition can be done in a randomness efficient way by using hitters; see, e.g., in [26]). For a constant error ε, the number of queries is $O(1)$, but for sub-constant error ε, the number of queries becomes super-constant. The

---

[1]In contrast, in UNIQUE-GAMES [23], each of the two answers determines uniquely a single satisfying answer to the other.

second item transforms the number of queries to 2 at the cost of enlarging the error to a fraction not much smaller than 1.

Low error can also be obtained while preserving two-query projection tests. This is done via parallel repetition. The parallel repetition transformation increases the randomness considerably, but decreases the error probability exponentially. This was first shown in [29]. (Recently, several improvements and simplifications were obtained by Holenstein [20] and Rao [28]).

Theorem 3 (Parallel repetition PCP, [29]). *There exists an alphabet Σ of constant size, such that for any $\varepsilon > 0$, $NP \subseteq PCP_{1,\varepsilon}[O(\log n \cdot \log \frac{1}{\varepsilon}), 2]_{\Sigma^{O(\log \frac{1}{\varepsilon})}}$. Moreover, the PCP verifier makes two-query projection tests.*

For a constant error ε, the randomness is $O(\log n)$, and the size is $\text{poly}(n)$. For sub-constant error ε, however, the randomness becomes super-logarithmic, and the size becomes super-polynomial. Interestingly, by a result of Feige et al [16], when applying parallel repetition to "natural" verifiers in order to decrease the error from $\frac{1}{2}$ to a small constant error ε, it is *necessary* to use $c \cdot \log n$ random bits, where $c > 1$ depends on ε.

Sub-constant error PCP Theorems are also known. In these theorems, the error is decreased below a constant while preserving polynomial size [30, 3, 13]. The state of the art in terms of the probability of error was proved in [13].

Theorem 4 (Sub-constant error PCP, [30, 3, 13]). *For any constant $\alpha > 0$, there exist $\varepsilon \le 2^{-(\log n)^{1-\alpha}}$ and alphabet Σ of size $|\Sigma| \le \frac{2}{\varepsilon}$, such that $NP \subseteq PCP_{1,\varepsilon}[O(\log n), O(1)]_{\Sigma}$.*

Theorem 4 gives a low error PCP Theorem with constant number of queries. However, the number of queries is strictly larger than 2.

We note also that one can use known algebraic techniques to obtain very low error with two-query projection tests. However, the alphabet size of this construction is always super-polynomial. The following theorem is folklore and follows from low degree testing theorems with sub-constant error [30, 3, 27].

Theorem 5 (Two-query projection tests PCP, [30, 3, 27]). *Fix any constant $\beta > 0$. Then, for every $\varepsilon \le \frac{1}{(\log n)^{\beta}}$ there exists an alphabet Σ of size $|\Sigma| \le 2^{\text{poly}(\frac{1}{\varepsilon})}$, such that $\mathcal{NP} \subseteq PCP_{1,\varepsilon}[O(\log n), 2]_{\Sigma}$. Moreover, the PCP verifier makes two-query projection tests.*

Note that the alphabet size is super-polynomial in this construction, no matter what the error is.

The randomness complexity of the verifier in the basic PCP Theorem can be improved, yielding a PCP Theorem with almost-linear size. Various papers achieved that [17, 10, 8, 12]. The state of the art is by Dinur [12], based on a result by Ben-Sasson and Sudan [9]:

Theorem 6 (Almost-linear size PCP, [9, 12]). $3\text{SAT} \in PCP_{1,\frac{1}{2}}[\log n + O(\log \log n), O(1)]$.

Note that the result is phrased for a specific $\mathcal{NP}$-Complete language 3SAT, rather than for all $\mathcal{NP}$. The reason is that the reduction from an arbitrary $\mathcal{NP}$ language to 3SAT may not preserve almost-linear size.

The transformations from Theorem 2 can be adapted to preserve almost-linear size:

Theorem 7 (Equivalent formulations, almost-linear size). *Theorem 6 is equivalent to each of the following:*

4

1. ***Low error:*** *For any $\varepsilon > 0$, $3\text{SAT} \in PCP_{1,\varepsilon}[\log n + O(\log\log n) + O(\log\frac{1}{\varepsilon}), O(\log\frac{1}{\varepsilon})]$.*

2. ***Two query projection tests:*** *There exist a constant $\delta > 0$ and an alphabet Σ of constant size, such that $3\text{SAT} \in PCP_{1,1-\delta}[\log n + O(\log\log n), 2]_\Sigma$. Moreover, the PCP verifier makes two-query projection tests.*

The first item follows from randomness efficient sequential repetition via hitters (see, e.g., in [26]). The second item is along the same lines as the second item in Theorem 2.

Recently, sub-constant error was achieved simultaneously with almost-linear size:

Theorem 8 (Sub-constant error PCP of almost-linear size, [27, 26]). *There exists a constant $\alpha > 0$, as well as $\varepsilon \leq 2^{-(\log n)^\alpha}$ and an alphabet Σ of size $|\Sigma| \leq 2^{(\log n)^{1-\alpha}}$, such that $3\text{SAT} \in PCP_{1,\varepsilon}[\log n + O((\log n)^{1-\alpha}), O(1)]_\Sigma$.*

Like Theorem 4, Theorem 8 gives a constant number of queries that is strictly larger than 2.

In light of the results described above, the following questions arise (see, e.g., [3]): Are there PCP Theorems with *two* queries and sub-constant error ? How about two-query projection tests and sub-constant error ? Are there such PCPs with almost-linear size ?

1.4 Our Results

We prove a PCP Theorem with two-query projection tests, sub-constant error and almost-linear size. More precisely, for any error $\varepsilon > 0$ (that can be any function of n), we obtain a construction with soundness ε, answer-size $\mathsf{poly}(\frac{1}{\varepsilon})$ and size $n^{1+o(1)} \cdot \mathsf{poly}(\frac{1}{\varepsilon})$. Our main theorem is as follows.

Theorem 9 (Main theorem). *For every $\varepsilon > 0$, there exists an alphabet Σ with $\log|\Sigma| \leq \mathsf{poly}(\frac{1}{\varepsilon})$, such that $3\text{SAT} \in PCP_{1,\varepsilon}[(1+o(1)) \cdot \log n + O(\log\frac{1}{\varepsilon}), 2]_\Sigma$. Moreover, the PCP verifier makes two-query projection tests.*

In particular, if $\varepsilon \geq \frac{1}{(\log n)^\beta}$, where β is a sufficiently small constant, the answer-size is logarithmic and the size is almost-linear. We note that for error $\varepsilon \leq \frac{1}{(\log n)^\beta}$, Theorem 9 follows from Theorem 5 (the PCP Theorem that is based on Low Degree Testing), but this is exactly the less interesting case where the alphabet size is super-polynomial. The new part is the construction for $\varepsilon \geq \frac{1}{(\log n)^\beta}$.

The previous work that is most related to Theorem 9 is Theorem 3 (the PCP Theorem obtained from the Parallel Repetition Theorem). Theorem 9 is incomparable to Theorem 3. While Theorem 9 obtains two-query projection tests with sub-constant error, and polynomial (even almost-linear) size, which cannot be obtained by Theorem 3, the answer-size in Theorem 9 is $\mathsf{poly}(\frac{1}{\varepsilon})$, rather than $O(\log\frac{1}{\varepsilon})$ in Theorem 3. Note that for $\epsilon = O(1)$, $\mathsf{poly}(\frac{1}{\varepsilon}) = O(\log\frac{1}{\varepsilon}) = O(1)$ and hence in this range Theorem 9 gives the same answer-size as the one in Theorem 3 (up to a constant), but with better size parameter (almost-linear size, rather than polynomial size).

1.5 Hardness of Label-Cover

We can also formalize our main result in terms of the optimization problem LABEL-COVER. The problem captures two-query projection tests and serves as the starting point for many of the existing hardness of approximation results.

Definition 1.1 (Label-Cover). *An instance of* LABEL-COVER *contains a regular bipartite multi-graph $G = (A, B, E)$ and two finite sets Σ_A and Σ_B, where $|\Sigma_A| \geq |\Sigma_B|$. Every vertex in A is supposed to get a label in Σ_A, and every vertex in B is supposed to get a label in Σ_B. For each edge $e \in E$ there is a projection $\pi_e : \Sigma_A \rightarrow \Sigma_B$ which is a partial function.*

Given a labeling to the vertices of the graph, i.e., functions $\varphi_A : A \rightarrow \Sigma_A$ and $\varphi_B : B \rightarrow \Sigma_B$, an edge $e = (a, b) \in E$ is said to be "satisfied" if $\pi_e(\varphi_A(a)) = \varphi_B(b)$ (it might be that $\pi_e(\varphi_A(a))$ is undefined; in which case $\pi_e(\varphi_A(a)) \neq \varphi_B(b)$).

The goal is to find a labeling that maximizes the number of satisfied edges. We say that γ fraction of the edges are satisfiable if there exists a labeling that satisfies γ fraction of the edges.

In the LABEL-COVER notation, the *size* corresponds to the number of vertices $|A| + |B|$. The *alphabet* corresponds to the (larger) set of labels Σ_A. The *randomness* is $\log |E|$.

Sometimes LABEL-COVER is defined with projections π_e that are functions, rather than partial functions. However, the more general definition of partial functions is convenient for us, and works just as well for the applications. In the literature one can find a variety of different problems that are named LABEL-COVER and are incomparable to Definition 1.1. However, today, the name LABEL-COVER usually refers to the problem defined in Definition 1.1.

Our main theorem can be restated as follows.

Theorem 10 (Main theorem). *For every n, and every $\varepsilon > 0$ (that can be any function of n) the following holds. Solving 3SAT on inputs of size n can be reduced to distinguishing between the case that a* LABEL-COVER *instance of size $n^{1+o(1)} \cdot \mathsf{poly}(\frac{1}{\varepsilon})$ and parameters $|\Sigma_A|$, $|\Sigma_B|$ s.t. $\log |\Sigma_A| \leq \mathsf{poly}(\frac{1}{\varepsilon})$ and $\log |\Sigma_B| \leq O(\log \frac{1}{\varepsilon})$, is completely satisfiable and the case that at most ε fraction of its edges are satisfiable.*

1.6 Some Implications of Our Main Theorem

In this section we demonstrate some of the prominent implications of Theorem 10. The presentation follows Khot's survey [24], and the reader is referred to this survey for more details.

The following scheme is used to prove hardness of approximation results:

1. Start with a two-query projection tests PCP Theorem with low error (the PCP based on parallel repetition, given in Theorem 3).

2. Apply Long Code [6, 18] and other techniques to convert the test performed by the verifier to the desired form.

This scheme has been successful in proving hardness of approximation results for many optimization problems. A prominent example is the work of Håstad [18] proving, among other results, the hardness of approximating 3SAT and 3LIN.

Theorem 10 can many times replace Theorem 3 in step 1. When one is interested in $\mathcal{NP}$-hardness results, this will usually give better results than what parallel repetition gives: almost-linear reductions, rather than polynomial or super-polynomial reductions, and sub-constant error, rather than constant error. This is true as long as step 2 does not use specific properties of parallel repetition other than two-query projection tests, and as long as the number of repetitions used is relatively small. (We note that when one is interested in hardness results under stronger assumptions such as $\mathcal{NP} \nsubseteq DTIME(2^{\mathsf{poly}\log n})$, one usually obtains better results using parallel repetition, i.e., using Theorem 3 rather than Theorem 10). In this section we demonstrate a few cases in which Theorem 10 can indeed be used to give better results in this scheme.

3SAT:

Håstad followed the above mentioned scheme to prove hardness results for many optimization problems, including the classical 3SAT [18]. Note that any $3CNF$ formula has an assignment satisfying at least $\frac{7}{8}$ fraction of its clauses. Thus, the best we can hope for is to show that it is $\mathcal{NP}$-hard to distinguish between the case that the formula is satisfiable and the case that only $\frac{7}{8}$ fraction of the clauses are satisfiable.

Corollary 11 (3SAT hardness). *Solving 3SAT on inputs of size n can be reduced to distinguishing between the case that a $3CNF$ formula of size $n^{1+o(1)}$ is satisfiable and the case that only $\frac{7}{8} + o(1)$ fraction of its clauses are satisfiable.*

This corollary improves over Håstad's result in two respects: First, it shows a hardness result based on an almost-linear size reduction, rather than a polynomial size reduction. Second, it shows that approximating the number of satisfiable clauses to within a factor of $\frac{7}{8} + o(1)$ is $\mathcal{NP}$-hard, and not only that approximating that number to within a factor of $\frac{7}{8} + \varepsilon$ for any constant $\varepsilon > 0$ is $\mathcal{NP}$-hard (as in Håstad's original result). The $o(1)$ term is roughly $(\log\log n)^{-\Omega(1)}$ because of Håstad's test construction that is based on the Long Code. We note that if we could have achieved in Theorem 10 the alphabet/error tradeoff of Theorem 3, $|\Sigma_A| \le \mathsf{poly}(\frac{1}{\varepsilon})$, the $o(1)$ term would have been $(\log n)^{-\Omega(1)}$. We note also that Håstad does obtain hardness of approximation results to within a factor of $\frac{7}{8} + o(1)$ where the $o(1)$ term is $(\log n)^{-\Omega(1)}$ (using the above mentioned scheme), but these are not $\mathcal{NP}$-hardness results and are based on stronger assumptions.

3LIN:

To obtain an optimal completeness/soundness gap for 3 query bits, we can follow Håstad [18] and consider the problem of solving linear equations on 3 variables over $GF(2)$. For this problem, one can efficiently check whether all the equations can be satisfied simultaneously by Gauss elimination. Hence, we give up perfect completeness.

Corollary 12 (3LIN hardness). *Solving 3SAT on inputs of size n can be reduced to distinguishing*

between the case that in a set of $n^{1+o(1)}$ linear equations, each depending on 3 variables over $GF(2)$, a fraction of $1 - o(1)$ of the equations can be satisfied, and the case that only $\frac{1}{2} + o(1)$ fraction of the equations are satisfiable.

Again, this corollary improves over Håstad's result in two respects: First, it shows a hardness result based on an almost-linear size reduction, rather than a polynomial size reduction. Second, it shows that approximating the number of satisfiable equations to within a factor of $\frac{1}{2} + o(1)$ is $\mathcal{NP}$-hard, and not only that approximating that number to within a factor of $\frac{1}{2} + \varepsilon$ for any constant $\varepsilon > 0$ is $\mathcal{NP}$-hard (as in Håstad's original result). The $o(1)$ term is the same as the one for 3SAT. Once again, we note also that Håstad does obtain hardness of approximation results to within a factor of $\frac{1}{2} + o(1)$ (where the $o(1)$ term is better than the one we obtain here), but these are not $\mathcal{NP}$-hardness results and are based on stronger assumptions.

Amortized query complexity and free bit complexity:

Assume for simplicity that the alphabet is $\Sigma = \{0, 1\}$. Roughly speaking, *amortized query complexity* is the ratio between $\log$ of the soundness and the number of queries. There are several similar and essentially equivalent definitions. Here, we refer by amortized query complexity to $\frac{q + \log(1/c)}{\log(1/s)}$, following [24]. "Free" queries are queries to which the answer can be arbitrary. The satisfying answers to the other queries are determined uniquely by the answers to the free queries. Roughly speaking, *amortized free bit complexity* is the ratio between $\log$ of the soundness and the number of free queries. Formally, we refer by amortized free bit complexity to $\frac{f + \log(1/c)}{\log(1/s)}$, where f is the number of free queries.

Samorodnitsky and Trevisan used the 3LIN test (and the above mentioned scheme) to obtain PCP Theorems in which the amortized query complexity is $1 + \varepsilon$ and the amortized free bit complexity is ε (for any constant $\varepsilon > 0$) [32]. Using a similar approach, our results imply a similar PCP Theorem with amortized query complexity $1 + o(1)$ and free bit complexity $o(1)$. Moreover, this is done by an *almost-linear* size reduction from 3SAT, rather than a polynomial size reduction in [32].
Corollary 13 (Nearly-optimal amortized query complexity and free bit complexity). *There exists a function $k_{max}(n) \geq \omega(1)$, such that, for any natural number $\omega(1) \leq k \leq k_{max}(n)$, 3SAT on inputs of size n has a verifier that uses $(1 + o(1)) \cdot \log n$ random bits to pick $q = k^2 + 2k$ queries to a binary proof, such that only $f = 2k$ of the queries are free. The verifier has completeness $c = 1 - o(1)$ and soundness at most $s = 2^{-k^2+1}$, implying amortized query complexity $1 + o(1)$ and amortized free bit complexity $o(1)$.*

1.7 Techniques

In the literature there are two main approaches for proving the PCP Theorem. The classical approach [2, 1] and Dinur's approach [12]. The classical approach starts with a PCP verifier with small error but very large alphabet, and gradually reduces the alphabet size. Dinur's approach starts with a PCP verifier with small alphabet but very large error, and gradually reduces the error. Our proof is more related to the first approach.

One of the main new ideas that we use can be viewed as a new technique for doing the composition step in the PCP Theorem, without increasing the number of queries to the proof.

Proof composition is what enabled the PCP theorem [2, 1], and is an important part of all PCP constructions since. In the literature, there are several different (but very related) ways to do the composition step (see for example [2, 34, 14, 8]). However, all these methods are either restricted to constant error, or require a large number of queries. For example, each application of the standard way of doing the composition step in the classical proof of the PCP Theorem, increases the number of queries to the proof by 1.

To formalize our techniques, we define the notion of *Locally Decode/Reject Code* (LDRC). Very roughly speaking, LDRCs are codes such that there exists a decoding algorithm that performs a local test on a codeword and based on the test either rejects or outputs the value of several positions in the encoded message. The decoding algorithm should satisfy the following two properties: 1) If it is given as an input a correct codeword then it always accepts and always returns the correct values of the encoded message. 2) Given any word as an input (not necessarily a correct codeword), with very high probability, if the algorithm does accept then the returned values agree with one of a small list of codewords (a list decoding of the word that is given as an input).

The notion of LDRC and variants of it were implicit, or even semi-explicit, in many previous works (e.g., [3, 30, 33, 13, 26]). We believe that the explicit formulation of LDRCs and their construction are of independent interest.

Our new composition technique is a composition of LDRCs, rather than a composition of verifiers. The difference between a verifier and an LDRC is that a verifier checks a predicate, while an LDRC checks a predicate and – provided that the predicate is satisfied – returns values. By using LDRCs, rather than verifiers, we deviate from the path taken in works such as [2, 1, 3, 10, 9, 8], and proceed in the path taken in [13, 26].

Our entire proof is presented as a construction of LDRCs with certain properties, rather than a construction of PCPs. We then use the new LDRCs to reduce the number of queries in existing constructions of PCPs. Thus, our proof can be viewed as a reduction that reduces the number of queries in PCP constructions. More precisely, the reduction converts a PCP with a large number of queries into a PCP with two-query projection tests, while not increasing the error by much.

We note however that for the construction of our LDRCs, we do use many of the techniques that were developed for constructions of PCPs, and our proof contains several steps that are similar to corresponding steps in the classical approach for proving the PCP Theorem. For example, we use the Reed-Muller and Hadamard codes and their local testing and self-correction properties. We do have several new techniques that we need in order to achieve a construction with two queries and sub-constant error.

As in the classical approach for proving the PCP Theorem, our construction starts with an LDRC with low error but large alphabet, and gradually reduces the alphabet size. The construction is by performing various transformations, including compositions of the Reed-Muller and Hadamard constructions, and other transformations. Our proof relies on algebraic constructions, yet the construction involves several combinatorial steps that are quite generic and may find other applications. The

9

combinatorial steps use expanders with a very large spectral gap. The use is different from the use of expanders of constant spectral gap in the work of Dinur [12] (although it bears some similarity to it). More details appear in Section 3 where we outline our construction.

The formal definition of LDRC, as well as more details and applications, appear in Section 2. We wish to emphasize that LDRCs are different from existing notions such as Relaxed Locally Decodable Codes (RLDCs) [8]. A comparison and a construction of RLDCs and locally testable codes from LDRCs appears in Section 2.3. The use of LDRCs to query reduction for PCPs appears in Section 2.2.

2 Locally Decode/Reject Codes for k-Tuples

A Locally Decode/Reject Code is an encoding $E : \{0,1\}^n \to \Sigma^m$ that maps messages $x \in \{0,1\}^n$ to codewords $E(x) \in \Sigma^m$. Σ is the alphabet of the code and m is its length. Underlying an LDRC there is a list of k-tuples of positions in $[n]$:

$$\langle i_{1,1}, \ldots, i_{1,k} \rangle, \ldots, \langle i_{N,1}, \ldots, i_{N,k} \rangle \in [n]^k$$

The code is associated with a local testing/decoding algorithm $\mathcal{A}$. The purpose of the algorithm is to decode a random k-tuple from the list. The algorithm is probabilistic and may only query a constant number of positions in Σ^m. Based on the answers to the queries it should either reject, or return a k-tuple from the list together with a decoding of k bits for it (see Figure 1). Note that the alphabet Σ needs to be large enough to allow that. In our setting, the algorithm makes two queries and performs a projection test. If the test passes, then based on the answer to the first query (that also gives the satisfying answer to the second query), the algorithm should decode a k-tuple.

Let $x \in \{0,1\}^n$ and fix some randomness for the algorithm $\mathcal{A}$. This fixing determines a k-tuple $\langle i_1, \ldots, i_k \rangle \in [n]^k$ in the hard-wired list. Let us say that the algorithm $\mathcal{A}$ on the fixed randomness *decodes* x if the algorithm $\mathcal{A}$ does not reject and does output $b_1 = x(i_1), \ldots, b_k = x(i_k)$, where $x(i_j)$ for $j \in [k]$ denotes the i_j'th symbol in x.

In Definition 2.1 below we state the properties of the local tester/decoder. Given access to a codeword $y = E(x)$, the algorithm must always decode x. The requirement from the algorithm when given as input a non-codeword is more subtle. In existing definitions of local decoders, the input y is assumed to be at least close (in Hamming distance) to some codeword $E(x)$, and the requirement is to decode x. In the definition of LDRCs, we will not assume that y is close to a codeword. That is, we allow y to be an arbitrary string in Σ^m. In this case, y may be far from all codewords. Hence, we allow the algorithm to reject if it cannot decode. Nonetheless, the list of codewords that are somewhat close to y cannot be large (when E defines a code with good distance). We require that with high probability, if $\mathcal{A}$ does not reject, $\mathcal{A}$ decodes one of a short list of messages $x_1, \ldots, x_l \in \{0,1\}^n$. It does not matter which x_i the algorithm decodes, but all k bits must be consistent with the same x_i (note that this is non-trivial when $l \ll 2^k$).

Definition 2.1 (Locally decode/reject code for k-tuples). *Let $0 < \delta_{min} < 1$. Let $l_{max} : (0,1) \to \mathbb{R}^+$ be a decreasing function. An encoding $E : \{0,1\}^n \to \Sigma^m$ together with a testing/decoding*

Local Tester/Decoder

Hard-wired: A collection of size N of k-tuples of positions in $[n]$:

$$\langle i_{1,1}, \ldots, i_{1,k} \rangle, \ldots, \langle i_{N,1}, \ldots, i_{N,k} \rangle \in [n]^k$$

Input: $y \in \Sigma^m$.

Goal: Test whether y locally agrees with a codeword, and, if so, decode k positions in the message, where the positions are chosen uniformly at random from the hard-wired list.

Output: Either *reject*, or a k-tuple of indices $\langle i_1, \ldots, i_k \rangle$ that is uniformly distributed in the hard-wired list, as well as k bits $b_1, \ldots, b_k \in \{0, 1\}$.

Process:

1. Pick in some randomized manner a constant number of queries to y (in our case, two queries), as well as a k-tuple $\langle i_1, \ldots, i_k \rangle$ that is uniformly distributed in the hard-wired list.

2. Perform some test on the queried positions in y (in our case, a projection test). If the test rejects, *reject*.

3. Use the queried positions to compute k bits $b_1, \ldots, b_k \in \{0, 1\}$.

Figure 1: Local tester/decoder

*algorithm $\mathcal{A}$ as in Figure 1 is called a (δ_{min}, l_{max})-*locally decode/reject code *for the hard-wired k-tuples, if the following holds:*

1. **Completeness:** *For every $x \in \{0, 1\}^n$, on input $E(x)$, the algorithm always decodes x.*

2. **Soundness:** *For every $y \in \Sigma^m$, for every real δ such that $\delta_{min} \leq \delta < 1$, there exist $l \leq l_{max}(\delta)$ messages $x_1, \ldots, x_l \in \{0, 1\}^n$, such that the following holds: the probability that the algorithm does not reject, yet does not decode any of $x_1, \ldots, x_l$, is at most $O(\delta)$.*

The parameter δ_{min} lower bounds the error of the LDRC, i.e., the probability that the tester/decoder accepts although it should not. The parameter l_{max} gives the list size as a function of the error we are willing to tolerate. Typically, $l_{max}(\delta) \leq \delta^{-O(1)}$.

2.1 Bipartite Locally Decode/Reject Codes

For our setting, let us also explicitly define the LABEL-COVER version of LDRCs. The notion of *bipartite LDRCs* imposes the two query projection tests structure on the local tester/decoder. The

notion is stronger than the notion in Definition 2.1. The encoding consists of two parts A and B. The list-decoding is determined solely by the B part.

Definition 2.2 (Bipartite locally decode/reject code for k-tuples). *Consider a list of k-tuples*

$$\langle i_{1,1}, \ldots, i_{1,k} \rangle, \ldots, \langle i_{N,1}, \ldots, i_{N,k} \rangle \in [n]^k$$

A Bipartite LDRC for the k-tuples is $\mathcal{G} = \langle G = (A, B, E), \Sigma_A, \Sigma_B, \{\pi_e\}_{e \in E}, \{\tau_e\}_{e \in E}, \{\rho_e\}_{e \in E} \rangle$, where $\mathcal{G}' = \langle G = (A, B, E), \Sigma_A, \Sigma_B, \{\pi_e\}_{e \in E} \rangle$ is an instance of LABEL-COVER*, and every edge $e \in E$ carries a k-tuple τ_e from the list and an evaluation function $\rho_e : \Sigma_A \to \{0, 1\}^k$. For each $j \in [N]$, the tuple $\langle i_{j,1}, \ldots, i_{j,k} \rangle$ appears on the same number of edges.*

Given a labeling to the vertices of the graph, i.e., functions $C_A : A \to \Sigma_A$ and $C_B : B \to \Sigma_B$, an edge $e = (a, b) \in E$ is said to be "satisfied" if it is satisfied in $\mathcal{G}'$. For a message $x \in \{0, 1\}^n$, the edge e is said to "decode" x if $\rho_e(C_A(a)) = \langle x_{i_1}, \ldots, x_{i_k} \rangle$ where $\tau_e = \langle i_1, \ldots, i_k \rangle$ is the tuple associated with e.

Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function. We say that the LDRC is a (δ_{min}, l_{max})-bipartite LDRC if it satisfies the following conditions:

1. **Completeness:** *For every $x \in \{0, 1\}^n$, one can efficiently compute assignments $C_A : A \to \Sigma_A$ and $C_B : B \to \Sigma_B$, such that all edges $e \in E$ are satisfied and decode x.*

2. **Soundness:** *For every $C_B : B \to \Sigma_B$, for every real δ such that $\delta_{min} \leq \delta < 1$, there exist $l \leq l_{max}(\delta)$ messages $x_1, \ldots, x_l \in \{0, 1\}^n$, such that the following holds for any $C_A : A \to \Sigma_A$: when picking uniformly at random an edge $e \in E$, the probability that e is satisfied but does not decode any one of $x_1, \ldots, x_l$, is at most $O(\delta)$.*

Note that for decoding to be possible, the alphabet must satisfy $\log |\Sigma_A| \geq k$.

In the LABEL-COVER notation, the length of the code corresponds to the number of vertices $|A| + |B|$, and the alphabet of the code corresponds to the (larger) set of labels Σ_A. The randomness of the local tester/decoder is $\log |E|$. For any interesting list of k-tuples (where we refrain from defining "interesting" explicitly; an "uninteresting" list may be one that does not even contain most possible indices in $[n]$), the length must be at least $\Omega(N + n)$. We refer to the size of the graph G, $|G| = |A| + |B| + |E|$, as the *size* of the bipartite LDRC. The size measures both the length of the LDRC and the number of possible tests of its local tester/decoder. We say that the construction is of *almost-linear* size, if the size is $(N + n) \cdot n^{o(1)}$.

We show a construction of an almost-linear size bipartite LDRCs as follows:

Theorem 14 (Construction of bipartite LDRC). *There exists a constant $0 < \alpha < \frac{1}{2}$ such that the following holds. Let k be such that $k \leq (\log n)^\alpha$. Let $\frac{1}{(\log n)^\alpha} \leq \varepsilon < 1$. Then, there is an efficient algorithm that given a collection of size N of k-tuples, outputs a $(\delta_{min} = \varepsilon, l_{max}(\delta) = \delta^{-O(1)})$-bipartite LDRC for these tuples. Its size is almost linear $(N + n) \cdot n^{o(1)}$, and its alphabets satisfy $\log |\Sigma_A| = k \cdot \mathsf{poly}(\frac{1}{\varepsilon})$ and $\log |\Sigma_B| = O(\log \frac{1}{\varepsilon})$. The degree of the A vertices is $(\frac{1}{\varepsilon})^{O(k)}$, and the degree of the B vertices is $(\frac{1}{\varepsilon})^{O(1)}$.*

12

2.2 Query Reduction For PCPs Via LDRCs

Bipartite LDRCs allow us to convert a PCP with a low error and a large number of queries, e.g., the one appearing in Theorem 7 (item 1), to a PCP with low error and two query projection tests. Using the bipartite LDRC construction of Theorem 14, we get our PCP theorem, Theorem 10.

Definition 2.3 (Construction algorithm). *A (k_{max}, δ_{min})-construction algorithm for bipartite LDRCs with parameters $\langle \text{size}, \text{block}_A, \text{block}_B \rangle$ is an efficient algorithm that given a collection of k-tuples, where $k \leq k_{max}$, outputs a (δ_{min}, l_{max})-bipartite LDRC for the tuples, where $l_{max}(\delta) \leq \delta^{-O(1)}$. The size of the output is size, the alphabet size of the A vertices is 2^{block_A} and the alphabet size of the B vertices is 2^{block_B}.*

Theorem 15 (Query reduction). *If there is a (q, ε)-construction algorithm for bipartite LDRCs with parameters $\langle \text{size} \leq (N + n) \cdot n^{o(1)}, \text{block}_A, \text{block}_B \rangle$, then for some $\varepsilon_0 \geq \varepsilon^{O(1)}$,*

$$PCP_{1,\varepsilon_0}[(1 + o(1)) \cdot \log n, q] \subseteq PCP_{1,O(\varepsilon)}[(1 + o(1)) \cdot \log n, 2]_{\{0,1\}^{\text{block}_A}}$$

Moreover, the PCP verifier performs two query projection tests, and the answer to the second query consists of block_B bits.

Proof. Denote by $\mathcal{C}$ a (q, ε)-construction algorithm for bipartite LDRCs with parameters $\langle \text{size} \leq (N + n) \cdot n^{o(1)}, \text{block}_A, \text{block}_B \rangle$. Assume that $\mathcal{C}$ outputs (ε, l_{max})-bipartite LDRCs. Let us choose $\varepsilon_0 \doteq \varepsilon / l_{max}(\varepsilon) \geq \varepsilon^{O(1)}$.

Let $L \in PCP_{1,\varepsilon_0}[(1 + o(1)) \cdot \log n, q]$. Denote the implied PCP verifier by V_1. Denote the set of randomness strings the verifier V_1 uses by R, where $|R| \leq n^{1+o(1)}$. On randomness $r \in R$, the verifier V_1 performs q queries to a binary proof of size $m \leq n^{1+o(1)}$; denote the q-tuple of queries that V_1 performs by $V_1(r) \in [m]^q$.

Invoke the construction algorithm $\mathcal{C}$ on the collection of size $n^{1+o(1)}$ of q-tuples $\{V_1(r)\}_{r \in R}$ to obtain a bipartite LDRC:

$$\mathcal{G} = \langle G = (A, B, E), \Sigma_A, \Sigma_B, \{\pi_e\}_{e \in E}, \{\tau_e\}_{e \in E}, \{\rho_e\}_{e \in E} \rangle$$

Identify Σ_A with $\{0,1\}^{\text{block}_A}$ and Σ_B with $\{0,1\}^{\text{block}_B}$. Note that the size of G is $n^{1+o(1)}$.

Consider the following PCP verifier V_2 for L. Assume that the verifier V_2 is given input x. The verifier V_2 has oracle access to a proof which it interprets as labels $\langle C_A, C_B \rangle$, where $C_A : A \to \{0,1\}^{\text{block}_A}$ and $C_B : B \to \{0,1\}^{\text{block}_B}$. Supposedly, C_A and C_B encode a proof that would have convinced the verifier V_1 that $x \in L$. The verifier V_2 proceeds as follows:

1. Pick uniformly at random an edge $e = (a, b) \in E$. Let $V_1(r)$ for a uniformly distributed $r \in R$ be such that $\tau_e = V_1(r)$.

2. If $\pi_e(C_A(a)) \neq C_B(b)$, *reject*.

3. Otherwise, accept or reject, depending on V_1's verdict on input x, randomness r and answers $\rho_e(C_A(a))$ to its queries.

Note that V_2 is efficient, uses only $(1+o(1)) \cdot \log n$ random bits to make two queries to a proof, where the answer to the first query consists of block_A bits, and the answer to the second query consists of block_B bits, and performs a projection test on the answers.

Let us argue completeness and soundness.

Completeness. Assume that $x \in L$. By the completeness of V_1, there exists a proof $\pi \in \{0,1\}^m$ that V_1 always accepts. Let $C_A : A \to \{0,1\}^{\mathsf{block}_A}$ and $C_B : B \to \{0,1\}^{\mathsf{block}_B}$ be labels for which all edges are satisfied and decode π. For these labels, the verifier V_2 always accepts.

Soundness. Assume that $x \notin L$. Consider labels $C_A : A \to \{0,1\}^{\mathsf{block}_A}$ and $C_B : B \to \{0,1\}^{\mathsf{block}_B}$. Let $\pi_1, \ldots, \pi_l \in \{0,1\}^m$ be the $l \leq l_{max}(\varepsilon)$ strings that follow from the definition of the LDRC $\mathcal{G}$ for the assignments C_A, C_B and the parameter ε.

Let us show that the probability that V_2 accepts on input x and proof $\langle C_A, C_B \rangle$ is at most $O(\varepsilon)$:

By the soundness of the LDRC $\mathcal{G}$, the probability that the edge e is satisfied in $\mathcal{G}$, but does not decode any of $\pi_1, \ldots, \pi_l$, is at most $O(\varepsilon)$.

For every $i \in [l]$, when V_1 is given input x and proof π_i, the probability over the randomness of V_1 that V_1 accepts is at most ε_0. Thus, the probability that given input x, the verifier V_1 accepts when given as proof one of $\pi_1, \ldots, \pi_l$, is at most $l \cdot \varepsilon_0 \leq O(\varepsilon)$. $\qquad \square$

Corollary 16. *Theorem 10 holds.*

Proof. Let $\varepsilon > 0$. Let us assume that for some constant $\beta > 0$ it holds that $\varepsilon \geq \frac{1}{(\log n)^\beta}$. Otherwise, the conclusion of Theorem 10 follows from Theorem 5. We will choose $\varepsilon' \geq \varepsilon^{O(1)}$ shortly. By Theorem 7 (item 1),

$$3\mathrm{SAT} \in PCP_{1,\varepsilon'}[(1 + o(1)) \cdot \log n, O(\log \frac{1}{\varepsilon})]$$

Apply the query reduction theorem (Theorem 15) using the LDRC construction algorithm given in Theorem 14. Deduce that for some $\varepsilon_0 \geq \varepsilon^{O(1)}$,

$$PCP_{1,\varepsilon_0}[(1 + o(1)) \cdot \log n, O(\log \frac{1}{\varepsilon})] \subseteq PCP_{1,\varepsilon}[(1 + o(1)) \cdot \log n, 2]_{\{0,1\}^{\mathsf{poly}(\frac{1}{\varepsilon})}}$$

Moreover, the PCP verifier performs two query projection tests, and the answer to the second query consists of $O(\log \frac{1}{\varepsilon})$ bits. Let $\varepsilon' \doteq \varepsilon_0$. Therefore, solving 3SAT on inputs of size n can be reduced to distinguishing between the case that a LABEL-COVER instance of size $n^{1+o(1)} \cdot \mathsf{poly}(\frac{1}{\varepsilon})$ and parameters $|\Sigma_A|, |\Sigma_B|$ s.t. $\log |\Sigma_A| \leq \mathsf{poly}(\frac{1}{\varepsilon})$ and $\log |\Sigma_B| \leq O(\log \frac{1}{\varepsilon})$, is completely satisfiable and the case that at most ε fraction of its edges are satisfiable. $\qquad \square$

2.3 Relaxed Locally Decodable Codes

In this section we recite the notion of Relaxed Locally Decodable Codes (RLDC) [8]. RLDCs are codes with local testing and decoding algorithm that are different from LDRCs. The notions are

incomparable: in one sense, the requirement of an RLDC is *stronger* than the requirement for an LDRC, while in another sense, the requirement of an RLDC is *weaker* than the requirement for an LDRC. In the sequel we compare the two notions.

To motivate Relaxed Locally Decodable Codes (RLDC), we revisit the definition of Locally Decodable Codes (LDCs) [21]. LDCs are encodings $C : \{0, 1\}^n \to \Sigma^m$ that are associated with a local decoding algorithm $\mathcal{A}$. The algorithm gets as input an index $i \in [n]$ and has oracle access to a word $y \in \Sigma^m$ that is *close* (in Hamming distance) to some encoding, i.e., there exists $x \in \{0, 1\}^n$ such that $\Delta(y, C(x)) \leq \delta$, where δ is a small constant. The purpose of $\mathcal{A}$ is to decode x_i. The algorithm is probabilistic and is allowed to query a constant number of positions in y. When $y = C(x)$, the algorithm should always output x_i. The soundness requirement is that *for any $i \in [n]$* and any y such that $\Delta(y, C(x)) \leq \delta$, the algorithm $\mathcal{A}$ decodes x_i with probability at least $\approx 1 - \delta$. The probability is only taken over the randomness of the algorithm $\mathcal{A}$, and not over the choice of $i \in [n]$.

The Hadamard code is locally decodable with two queries, but its length is exponential $m = 2^n$. The best constructions known today (under the assumption that there are infinitely many *Mersenne primes*) are slightly sub-exponential $2^{n^{o(1)}}$ and obtain a local decoder that queries 3 bits [35]. For two queries, an exponential lower bound is known [22]. For more queries, a super-linear lower bound is known [21].

Motivated by this state of affairs, Ben-Sasson et al [8] relaxed the notion of LDCs as to enable succinct constructions. Their idea was to allow the decoder *not* to decode a position. In this case, the decoder should declare that it cannot decode and *reject*. Of course, the decoder must not use this privilege too often: it may declare it cannot decode only few positions, depending on the distance of the received word from the code. The key point is that, *for every position $i \in [n]$*, the algorithm $\mathcal{A}$ may err, i.e., not reject yet return a wrong value, with only a small probability over its random coin tosses. There must not be even one $i \in [n]$ for which algorithm $\mathcal{A}$ err with large probability (unlike for LDRCs).

Definition 2.4 (Relaxed locally decodable code). *Let $0 < \delta < 1$ and let $0 < \rho < 1$. An encoding $E : \{0, 1\}^n \to \Sigma^m$ together with a decoding algorithm $\mathcal{A}$ as in Figure 2 is called a (δ, ρ)-relaxed locally decodable code, if the following holds:*

1. **Completeness:** *For every $x \in \{0, 1\}^n$ and every $i \in [n]$, on input $E(x)$ and i, the algorithm $\mathcal{A}$ outputs x_i.*

2. **Soundness:** *Given a word $y \in \Sigma^m$ with $\Delta(y, E(x)) \leq \delta$,*

   (a) *For every position $i \in [n]$, the probability that $\mathcal{A}$ does not reject, yet outputs $b \neq x_i$, is at most $\frac{2}{3}$.*

   (b) *For at least ρ fraction of the positions $i \in [n]$, the probability that $\mathcal{A}$ does not reject, and does output x_i, is at least $\frac{2}{3}$.*

The main differences between RLDCs and LDRCs are as follows:

- *List decoding vs. unique decoding.* In RLDCs, the guarantee is that the word is very close to

15

Relaxed Local Decoder

Input: $y \in \Sigma^m$ and $i \in [n]$.

Goal: Find x_i for $x \in \{0,1\}^n$ such that $E(x)$ is the closest codeword (in Hamming distance) to y.

Output: Either *reject*, or a bit $b \in \{0,1\}$.

Process:

1. Pick in some randomized manner a constant number of queries to y.

2. Perform some test on the queried positions in y. If the test rejects, *reject*.

3. Use the queried positions to compute the bit $b \in \{0,1\}$.

Figure 2: Relaxed local decoder

a (unique) codeword. In LDRCs there is no such guarantee. The local decoder has to perform well in the list decoding region.

- *Average case vs. worst case.* In RLDCs, the local decoder has to perform well for *all* indices with high probability over its randomness. In LDRCs, the local decoder has to perform well for *almost all* indices with high probability over its randomness. There might be very few indices, on which the algorithm always returns an incorrect value.

- k-*tuple vs. one position.* In RLDCs (as in LDCs), the requirement is to decode one position. In LDRCs, the requirement is to decode k positions.

3 Construction Outline

In this section we outline our LDRC construction, i.e., the proof of Theorem 14. We present a simplified construction, taking the liberty of ignoring several issues. In particular, in this outline we ignore the almost-linear size guarantee. The reason is that the ideas involved in handling almost-linear size appear in previous works [27, 26] and introduce many technical difficulties. A full account of issues we ignore in this outline appears in Section 3.12. Recall that obtaining a PCP Theorem with two query projection tests and sub-constant error, even with polynomial size, was unknown prior to our work.

3.1 Encoding codewords

The first conceptual step is as follows: Instead of LDRCs encoding binary strings as in Definition 2.2, we will construct LDRCs encoding codewords in some code $C \subseteq \Gamma^n$. The formal definition appears in Section 3.2. The differences from Definition 2.1 are as follows:

- Given a codeword $x \in C$, we would like assignments C_A, C_B "encoding" x (the encoding does not need to work for all possible strings, only for codewords in C; this is a relaxation of Definition 2.1).

- Given assignments C_A, C_B we would like a "list decoding" of codewords $x_1, \ldots, x_l \in C$ (the list decoding cannot use any string in the decoding, only codewords in C; this is a strengthening of Definition 2.1).

Note that constructing LDRCs for an infinite family of efficiently encodable *linear* codes $C = \{C_n\}$ yields, in particular, LDRCs as in Definition 2.1. The reason is that given a binary string $x \in \{0, 1\}^n$ we can first encode it using a code $C_{n'}$ for a sufficiently large n', and obtain a codeword $x' \in C_{n'}$ such that x is a prefix of x' (by linearity, we can assume, without loss of generality, that the code is *systematic*, i.e., an encoding contains the message bits and a sequence of linear functions of these bits). Then, we can use an LDRC for the code $C_{n'}$. The positions we wish to decode in x also appear in x'.

For the final construction of Theorem 14, we use as our linear code C the concatenation of Reed-Muller and Hadamard. The reason is that both for the Reed-Muller code and for the Hadamard code, we can *locally decode/reject*. For the Reed-Muller code – by *low degree testing* and using curves. For the Hadamard code – by *linearity testing* and using linear subspaces.

Nonetheless, the Reed-Muller code and the Hadamard code have apparent caveats. For the Reed-Muller code – while the length can be made almost-linear, the alphabet size is too large. For the Hadamard code – while the alphabet size is small, the length is exponential.

Our methods allow us to gain from the advantages of the two codes, while not losing much from their shortcomings. By composing the respective LDRCs, we obtain locally decode/reject codes with almost-linear length and small alphabet.

3.2 New Notion of LDRC and Its Parameters

Next we formulate the new notion of LDRCs we use. In this new notion we make an additional "technical" change.

We associate satisfiability constraints with the A vertices, rather than with the edges. Recall that the projections on the edges are partial functions. That is, on some values of the A endpoint, an edge may never be satisfied. Instead, we define the projections to be functions and define sets $\{\chi_a\}_{a \in A}$, where for every $a \in A$ we have $\chi_a \subseteq \Sigma_A$ is the set of satisfying values for a. A label to the vertex a projects on all the neighbors of a. The satisfiability constraint on a may check consistency between the different projections.

Definition 3.1 (Bipartite locally decode/reject code for k-tuples; new notion). *Fix a code $C \subseteq \Sigma^n$. Consider a list of k-tuples*

$$\langle i_{1,1}, \ldots, i_{1,k} \rangle, \ldots, \langle i_{N,1}, \ldots, i_{N,k} \rangle \in [n]^k$$

A Bipartite LDRC for the k-tuples is $\langle G = (A, B, E), \Sigma_A, \Sigma_B, \{\chi_a\}_{a \in A}, \{\pi_e\}_{e \in E}, \{\tau_e\}_{e \in E}, \{\rho_e\}_{e \in E} \rangle$, where $\langle G = (A, B, E), \Sigma_A, \Sigma_B, \{\pi_e\}_{e \in E} \rangle$ is an instance of LABEL-COVER, *and every edge $e \in E$ carries a tuple τ_e from the list and an evaluation function $\rho_e : \Sigma_A \to \Sigma^k$. For each $j \in [N]$, the tuple $\langle i_{j,1}, \ldots, i_{j,k} \rangle$ appears on the same number of edges.*

Given a labeling to the vertices of the graph, i.e., functions $C_A : A \to \Sigma_A$ and $C_B : B \to \Sigma_B$, a vertex $a \in A$ is said to be "satisfied" if $C_A(a) \in \chi_a$. An edge $e = (a, b) \in E$ is said to be "satisfied" if a is satisfied and $\pi_e(C_A(a)) = C_B(b)$. Given a message $x \in C$, an edge $e = (a, b) \in E$ is said to "decode" x, if $\rho_e(C_A(a)) = \langle x_{i_1}, \ldots, x_{i_k} \rangle$ where $\tau_e = \langle i_1, \ldots, i_k \rangle$ is the tuple associated with e.

Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function. We say that the LDRC is a (δ_{min}, l_{max})-bipartite LDRC if it satisfies the following conditions:

1. **Completeness:** *For every $x \in C$, there are assignments $C_A : A \to \Sigma_A$ and $C_B : B \to \Sigma_B$, such that all edges $e \in E$ are satisfied and decode x.*

2. **Soundness:** *For every $C_B : B \to \Sigma_B$, for every real δ such that $\delta_{min} \leq \delta < 1$, there exist $l \leq l_{max}(\delta)$ messages $x_1, \ldots, x_l \in C$, such that the following holds for any $C_A : A \to \Sigma_A$: when picking uniformly at random an edge $e \in E$, the probability that e is satisfied but does not decode any one of $x_1, \ldots, x_l$, is at most $O(\delta)$.*

We will be interested in various properties of an LDRC. One of them is the form of the satisfiability constraints of the A vertices. Another is the alphabets Σ_A and Σ_B. The alphabets will usually be codes themselves.

We will also be interested in the following parameters:

1. **Size.** The size of the LDRC, i.e., $|A| + |B| + |E|$. The size combines the length of the code $|A| + |B|$ and the randomness of the tester/decoder $\log |E|$. As we mentioned earlier, in this outline we will focus on polynomial size.

2. **Block length.** The block length of the A vertices is $\log |\Sigma_A|$. The block length of the B vertices is $\log |\Sigma_B|$.

3. **Left degree.** In this outline we focus on graphs that are left regular, i.e., all the A vertices have the same degree. The left degree is this degree.

4. **Right degree.** In this outline we focus on graphs that are right regular, i.e., all the B vertices have the same degree. The right degree is this degree.

3.3 Locally Decode/Reject Code for Reed-Muller

In this section we describe a locally decode/reject code for the Reed-Muller code. The construction is (a variant of) the folklore construction that yields Theorem 5.

Let the parameters of the Reed-Muller code be: a finite field $\mathbb{F}$, a dimension m and a degree d. The code consists of all m-variate polynomials of degree at most d over the field $\mathbb{F}$. It will be convenient to identify positions in the code with points in $\mathbb{F}^m$. Thus, the k-tuples we wish to decode are given as tuples of points:

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

The size of the LDRC is polynomial in $|\mathbb{F}^m|$. The left degree is $|\mathbb{F}|^{O(1)}$, and the right degree is polynomial in $|\mathbb{F}^m|$. The alphabet of the A vertices is a Reed-Muller code itself, but of much reduced parameters: the dimension and the degree are $O(\log kd)$ (independent of the initial dimension m). Still, the block length is large $\text{poly}(k, d) \cdot \log |\mathbb{F}|$ (compare it to the lower bound $k \cdot \log |\mathbb{F}|$; note the dependence on the degree d), and this is the main disadvantage of this construction. To see how severe this disadvantage is, recall that the degree d must be taken to be large if we want a good rate: The number of codewords in the Reed-Muller code is $|\mathbb{F}|^{\binom{m+d}{m}}$. Thus, the rate is $\binom{m+d}{m}/n$, where $n = |\mathbb{F}^m|$. To get large distance and polynomial length, we need to take d such that $m \cdot |\mathbb{F}|^{\Omega(1)} \leq d \ll |\mathbb{F}|$. Hence, the alphabet is at best super-polynomial $2^{\text{poly} \log n}$. If we wish the length to be almost linear, the alphabet becomes even larger, slightly sub-exponential.

What allows local testing/decoding for the Reed-Muller code is a principle that we loosely state as follows:

Low degree testing principle: Fix a function $f : \mathbb{F}^m \to \mathbb{F}$. There are a few low degree polynomials $q_1, \ldots, q_l : \mathbb{F}^m \to \mathbb{F}$ as follows. Pick uniformly at random a line ℓ in $\mathbb{F}^m$. "Almost surely, when f agrees with some low degree polynomial on a non-negligible fraction of the points on ℓ (*local test*), it agrees on these points with one of $q_1, \ldots, q_l$ (*global conclusion*)".

Here *non-negligible* means a sufficiently large fraction $m^{O(1)} \cdot \left(\frac{d}{|\mathbb{F}|}\right)^{\Omega(1)}$.

Proofs of variants of the above principle appear in [3, 30, 13, 27]. They rely on the distance property of the Reed-Muller code and on its recursive structure: any low degree subset of $\mathbb{F}^m$ defines a Reed-Muller sub-code. The "line" in the above principle can be replaced by any low degree curve or manifold in $\mathbb{F}^m$. In particular, we replace the "line" with a low degree manifold that goes through a k-tuple we wish to decode.

We will refer to our LDRC as the Manifold vs. Point construction, and define it as follows:

- The A vertices are 4-dimensional manifolds of degree at most $k+1$ in $\mathbb{F}^m$: there is a manifold for every k-tuple we wish to decode and three dimensional subspace in $\mathbb{F}^m$ (where we use three-dimensional subspaces because we apply the low degree testing of [27]). The manifold contains the k points and the subspace.

- The B vertices are the points in $\mathbb{F}^m$.

- Every manifold is connected to all the points on it, except for a few points that are removed to ensure right regularity.

- A label to a B vertex is a field element.

- A label to an A vertex is a 4-variate polynomial of degree at most $(k+1) \cdot d$. The projections $\{\pi_e\}_{e \in E}$ are such that for an edge $e = (a,b) \in E$, the projection $\pi_e(\sigma_a)$ is the value of the polynomial σ_a on the point corresponding to b on a.

- To encode a Reed-Muller codeword, we view it as a labeling of the B vertices (i.e., the points in $\mathbb{F}^m$) with field elements. We label the A vertices with the restrictions of the codeword to the manifolds on the A side.

- There are no satisfiability constraints, i.e., for every vertex $a \in A$, we have that χ_a is the set of all possible labels.

- For an edge $e = (a,b) \in E$, the tuple τ_e is the k-tuple contained in a. The evaluation $\rho_e(\sigma_a)$ is the value of the polynomial σ_a on the k points corresponding to τ_e.

Note that the labels to the A vertices are Reed-Muller codewords, but not Reed-Muller codewords with the parameters we declared. Yet, we can represent every polynomial with constant dimension and degree $O(kd)$ as a polynomial with dimension and degree $O(\log kd)$.

3.4 Why Composition is Hard – The Two-Prover Game Perspective

It will be useful to think of the LDRC construction above in terms of a game between a verifier and two provers: prover A and prover B. The verifier asks prover A about a manifold in $\mathbb{F}^m$. The verifier asks prover B about a point on the manifold. Prover A knows that prover B is asked about one of the points on the manifold that prover A got, but prover A does not know which point. Prover B knows that prover A is asked about one of the manifolds that contain the point that prover B got, but prover B does not know which manifold. The low degree testing principle assures us that this missing information is sufficient to force the provers to adhere to the same low degree polynomials.

Unfortunately, the alphabet of this Manifold vs. Point construction is large, because prover A needs to describe a polynomial for its manifold. A natural solution is to use *composition*. Prover A needs to provide a polynomial for the entire manifold, but, in fact, the verifier is only interested in the value of this polynomial on $k+1$ points: the k-tuple it decodes and the point that prover B got (on which the verifier compares the answers). The idea is to view the verifier's task as decoding $k+1$ symbols from a Reed-Muller code, and use an LDRC for this purpose.

Let us demonstrate the idea by taking the Manifold vs. Point construction of Section 3.3 as a concrete instance of an LDRC. Instead of prover A, we will have two provers: prover $A.A$ and prover $A.B$. Provers $A.A$ and $A.B$ will convince the verifier in $k+1$ values that are all evaluations

of one low degree polynomial for the manifold on the relevant points. The verifier will send prover $A.A$ a sub-manifold within the manifold for A that goes through the $k + 1$ points. The verifier will send prover $A.B$ a point in the sub-manifold. The provers $A.A$ and $A.B$ are expected to reply with consistent evaluations.

The composition we described increases the number of provers/queries from two: A and B, to three: $A.A$, $A.B$ and B. When composition is applied several times, the number of queries increases even further. Indeed, this is what happened in previous works that applied composition for similar needs [3, 30, 13, 26].

A-priori, it seems that we could insist on using only two provers by, in addition to B's original role, letting each of provers A, B simulate one of provers $A.A$, $A.B$. However, this fails, no matter how we attempt to split $A.A$ and $A.B$ between A and B. The reason is that the questions in the inner protocol reveal information on the questions of the outer protocol, in a way that some prover will always gain information about the outer question of the other prover. To see this, let us check the two splitting alternatives:

1. **First alternative: $\mathbf{A}.\mathbf{A} \to \mathbf{A} \mid \mathbf{A}.\mathbf{B} \to \mathbf{B}$.** Prover A gets the sub-manifold. Prover B gets the outer point (on the manifold) and the inner point (on the sub-manifold). In this case, prover A gains information about the point of B from knowing the sub-manifold that contains it.

2. **Second alternative: $\mathbf{A}.\mathbf{A} \to \mathbf{B} \mid \mathbf{A}.\mathbf{B} \to \mathbf{A}$.** Prover A gets the inner point (on the sub-manifold). Prover B gets the outer point (on the manifold) and the sub-manifold. In this case, prover B gains information about the manifold from knowing the sub-manifold.

The question that arises is whether one can devise a composition protocol for two provers in which the questions of the inner protocol do not give (enough) information on the questions of the outer protocol.

3.5 The Key Idea: Confusing the Provers

The problem with implementing composition is that we cannot afford that *any* of the provers will learn the sub-manifold. We saw that once any of the provers learns the sub-manifold, that prover gets information that may allow the provers to fool the outer verifier.

What we do instead is let *both* provers learn the sub-manifold. The key idea is that prover A will also get many other sub-manifolds to confuse it. Prover A will not know which of the sub-manifolds that it got is the one that prover B got. Prover B will not know which of the possible questions to prover A prover A actually got (where each question to prover A is a collection of manifolds containing the manifold that prover B got).

In the next few sections we will show that the two prover game can indeed be transformed into a form in which the question to prover A consists of a few manifolds, and the question to prover B consists of a single manifold. Since we wish the alphabet to be as small as possible, the question to prover A should consist of as few manifolds as possible.

3.6 Reducing Right Degree

For the presentation, we will go back to viewing the Manifold vs. Point LDRC of Section 3.3 as a bipartite graph. Our first step is to decrease the right degree of the graph to some small D.

For right degree reduction, we use a regular expander graph $H = (V_H, E_H)$ with number of vertices that equals the original right degree of the graph, degree D and second eigenvalue D^α for a constant $\frac{1}{2} \le \alpha < 1$.

The construction is as follows (see Figure 3):

- The A vertices are as before. The labels to the A vertices are as before.

- For every vertex $b \in B$ and every expander vertex $v \in V_H$, create a copy $\langle b, v \rangle$. The labels to the copies are as the labels to the original B vertices.

- For every edge $(a, b) \in E$ in the original graph, where (a, b) is the u'th edge coming into b, and every expander edge $(u, v) \in E_H$, create an edge $(a, \langle b, v \rangle)$. The projection π, the tuple τ and the evaluation function ρ that this edge carries are the same as the projection, tuple and evaluation of (a, b).
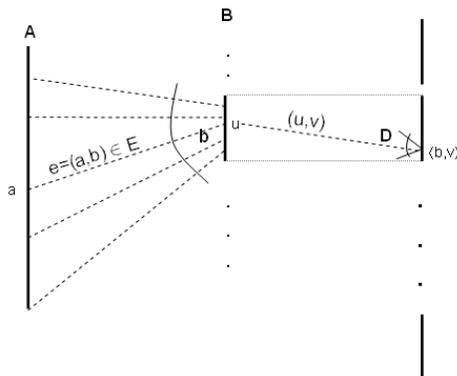

Figure 3: Right degree reduction.

It turns out that choosing $D = \lceil \frac{1}{\varepsilon} \rceil$, where ε is the error we aim for, suffices for soundness. Note that the left degree, as well as (essentially) the size, are multiplied by a factor of D.

The construction uses in an essential way the fact that we only decrease the right degree, and not the left degree. It works because each left vertex determines a labeling for all the copies of each of its neighbors. More than that, it is unreasonable to expect any construction of this kind to reduce left degree: Revealing to prover A a short list of points in it, among them the point that prover B got, allows prover A to choose a polynomial that agrees with prover B on all these points. This can be done even when prover B's answers do not correspond to any low degree polynomial.

3.7 The Sunflowers Construction

To transform our LDRC into the form described in Section 3.5, we switch the roles of prover A and prover B. When the right degree is small, we can do that while preserving the projection property: we can ask prover B to answer about a vertex $a \in A$, and ask prover A to answer about all the neighbors of a vertex $b \in B$.

Here is where we pay in the alphabet size. Recall that in Theorem 14, and hence in our PCP construction (Theorem 10), the block length depends polynomially on $\frac{1}{\varepsilon}$, rather than on $\log \frac{1}{\varepsilon}$. The reason is that the block length depends linearly on the degree, rather than on the logarithm of the degree: we keep separate information about each neighbor.

The construction is as follows:

- The new A vertices are the old B vertices, and the new B vertices are the old A vertices. The edges are flipped, but otherwise remain the same.

- The alphabet of the new B vertices is the alphabet of the old A vertices. A label of a new A vertex consists of D labels for the new B vertices, one per neighbor.

- For every old vertex $b \in B$, for every $i \in [D]$, assuming that $e = (a, b)$ is the i'th edge touching b, for a label $\vec{p} = \langle p_1, \ldots, p_D \rangle$ to b in the new graph, $\pi_e(\vec{p}) = p_i$.

- Each edge e is associated with the same tuple τ_e as before. The evaluation function that e carries follows from the previous evaluation function: Assume that the input to the new evaluation function is $\vec{p} = \langle p_1, \ldots, p_D \rangle$. Then, the evaluation is obtained using the old evaluation ρ_e on p_i where e is the i'th edge according to the ordering we defined.

- The new A vertices have a satisfiability constraint: for every old b vertex, which corresponds to a point in $\mathbb{F}^m$, all the D labels must agree on the point.

It is instructive to think of a new A vertex a as a sunflower, composed of its neighboring B vertices as petals (as in Figure 4). The neighboring B vertices intersect on a point given by a. A satisfying label to a is composed of labels to the neighboring B vertices that are consistent on the intersection.

3.8 Right Degree Reduction on the Sunflowers Construction

The Sunflowers construction gives a graph with small left degree and large right degree, rather than a graph with small right degree and large left degree. We can apply right degree reduction on this graph, and get a graph with small left and right degrees.

In the two prover game terminology, prover A gets D different manifolds that have a common "center" in their intersection. Prover B gets one of these manifolds. Prover A does not know which manifold prover B got. Prover B does not know which sunflower containing its manifold, among
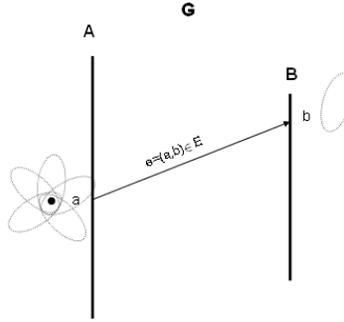
Figure 4: The Sunflowers construction.

D possible sunflowers, prover A actually got. Yet, although each of the provers only has a small amount of uncertainty regarding the question that the other prover was asked, both provers should prefer to adhere to the prescribed strategy.

In addition, the satisfiability constraints on prover A's answer can be checked by querying its manifolds on a few points. To see that, note that the satisfiability constraints in the Sunflowers construction in fact check: (i) agreement on "centers"; (ii) identity between copies of the same manifold. The (ii) checks come from the right degree reduction, and can be done by comparison on a random point.

The new structure of the Sunflowers construction is what allows composition in the next section.

3.9   Composition

In this section we start with the Sunflowers LDRC we constructed in Section 3.8, and show how to perform composition of this LDRC with inner LDRCs of the same type. The purpose of composition is to obtain an LDRC with lower alphabet. The block length of the composed LDRC is proportional to the left degree of the outer construction and the block length of the inner construction (and independent of the block length of the outer construction). The composition preserves the structure of Section 3.8.

Prover A in the outer LDRC needs to provide for each manifold that is a petal in its sunflower, a polynomial for the entire manifold. However, in reality, we are only interested in the value of this polynomial on $k + 1$ points[2]: the k-tuple we wish to decode and the point in the center of the sunflower (on which we compare the answers from all petals). Note that, this time, prover B already knows the manifold that prover A got, because prover B also got the exact same manifold.

The idea is to use an inner Sunflowers LDRC to decode the $k + 1$ positions in the Reed-Muller codeword corresponding to a polynomial on the manifold. Instead of prover B, we will have two provers: prover $B.A$ and prover $B.B$. Provers $B.A$ and $B.B$ can convince the verifier in $k + 1$

---

[2]To simplify the presentation, we ignore the random point needed for the (ii) checks. In Section 3.12 we remark how this matter is solved.

values that are all evaluations of one low degree polynomial for the manifold on the relevant points. We insist on using only two provers by letting prover A simulate prover $B.A$ and prover B simulate provers $B.B$. We have to make sure that the questions in the inner protocols reveal no information on the questions of the outer protocol.

Prover A does not get information, because prover A simulates prover $B.A$ for every petal in its sunflower. Hence, prover A does not know which petal is the one that prover B was asked about. (The picture is completely symmetric from prover A's point of view).

However, prover B does gain information about the outer question of prover A, because the $k+1$ points that the inner LDRC decodes, reveal the outer center, and hence give information about the outer question that A got. For that reason we change the protocol a little bit. In order to confuse prover B, each inner LDRC on an outer edge (a, b) decodes not only the $k + 1$ points that need to be decoded but also the $k + 1$ points that every other neighbor of b needs to decode. Since the right degree of the outer LDRC is small, this is possible. Now the picture is completely symmetric from the point of view of prover B and hence prover B gets no information about the outer question of prover A.

In the composed two prover game, the verification is as follows:

1. *Outer sunflower:* Pick at random a sunflower containing D manifolds, as well as one of these manifolds.

2. *Inner sunflower:* For every manifold, pick at random a sub-sunflower containing D sub-manifolds. For the manifold that was picked, pick a sub-manifold in it.

3. Ask prover A about *all* the D sub-sunflowers (one for each manifold). Ask prover B about the sub-manifold. Check the consistency between their answers.

Why does this protocol work? Prover A does not know which of the D^2 sub-manifolds is the one that prover B was asked about. Prover B does not know which of the sunflowers (that contain the sub-manifold that prover B got) is the one that prover A was asked about. Hence, both provers would better off adhere to their prescribed strategy.

The composed graph is as shown in Figure 5. There is an A vertex in the composed graph for every pair $\langle a, a_{in} \rangle$ of an outer A vertex a and an inner A vertex a_{in}. It should be thought of as taking the sunflower a_{in} for each of the petals of the sunflower a. There is a B vertex in the composed graph for every pair $\langle b, b_{in} \rangle$ of an outer B vertex b and an inner B vertex b_{in}. It should be thought of as taking the sub-manifold b_{in} inside the manifold b. Every outer edge (a, b) is replaced by an inner graph for decoding all the $(k + 1)$-tuples for neighbors of b.

Composition essentially multiplies the size of the outer construction and the inner construction. The inner construction is typically smaller, and hence the dominant factor is the size of the outer construction. Composition multiplies the outer and inner left degrees, as well as the outer and inner right degrees. Nonetheless, the degrees remain polynomial in $\frac{1}{\varepsilon}$. By a single composition, we can get the block length down from $\mathsf{poly}(k, d) \cdot \log |\mathbb{F}|$ to $\mathsf{poly}(k, \log d, \frac{1}{\varepsilon}) \cdot \log |\mathbb{F}|$. This block length is
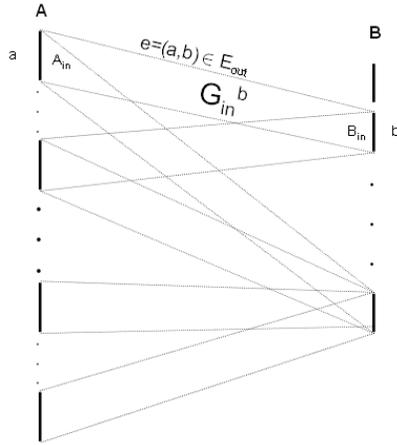
Figure 5: Composed graph.

small, but not as small as we want (recall that we wish to eliminate the dependence on d and $|\mathbb{F}|$). We solve this in the next section.

3.10 Locally Decode/Reject Code for Hadamard and The Concatenation of Reed-Muller and Hadamard

We solve the still-too-large-alphabet problem the same way as all PCP constructions since [1]: compose our construction with a construction for the Hadamard code. This results in arbitrarily small alphabet at the cost of a larger size. First, let us describe the Hadamard construction. In the next section we describe the composition with it.

We let the Hadamard code be over a small finite field $\mathbb{L}$. The field $\mathbb{L}$ may be $GF(2)$, but for low error we use larger fields $|\mathbb{L}| = (\frac{1}{\varepsilon})^{O(1)}$. We use the letter $\mathbb{L}$ to distinguish the field from the field $\mathbb{F}$ we used for the Reed-Muller code. It will be convenient to take $\mathbb{L}$ to be a subfield of $\mathbb{F}$. Let w be such that we can identify the B alphabet of the construction of Section 3.9 with $\mathbb{L}^w$. This w will be the dimension of the Hadamard code we take. The length of the Hadamard code is $|\mathbb{L}^w|$, which is exponential in w, but since w is relatively small, this is tolerable.

It will be convenient to identify positions in the Hadamard code with points in $\mathbb{L}^w$. Thus, a list of k-tuples we wish to decode can be thought of as a list of k-tuples of points:

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{L}^w)^k$$

We can construct an LDRC for the Hadamard code similarly to the way we constructed LDRCs for the Reed-Muller code in Section 3.3. The difference from the Reed-Muller LDRC is that we consider $(k+2)$-dimensional linear subspaces in $\mathbb{L}^w$ instead of degree-$(k+1)$ manifolds. In addition, the correctness of the construction now follows from linearity testing theorems (e.g., of [19]), rather than from the more difficult low degree testing theorems.

26

Importantly, the Hadamard construction also gives an LDRC for the concatenation of Reed-Muller and Hadamard: Not only that we can locally decode/reject the w symbols of a label (where the label corresponds to a Reed-Muller codeword), but we can also locally decode/reject any $\mathbb{L}$-*linear function* of the w symbols. In particular, we can locally decode/reject any symbol in the concatenation of the Reed-Muller codeword with Hadamard.

3.11 Composition of Locally Decode/Reject Code for Reed-Muller with Locally Decode/Reject Code for Concatenation of Reed-Muller and Hadamard

To design an LDRC of reasonable rate for the concatenation of Reed-Muller and Hadamard, we compose the LDRC for Reed-Muller obtained in Section 3.9 with the low rate LDRCs for the concatenation of Reed-Muller and Hadamard obtained in Section 3.10. The composition is along the same lines as the composition of LDRCs for Reed-Muller described in Section 3.9.

The major difference is as follows. We cannot ask the inner LDRCs to decode the evaluation of a manifold on the center of a sunflower as we could earlier. This is because the inner LDRCs can return symbols in $\mathbb{L}$, and not symbols in (the much too large) $\mathbb{F}$. However, the inner LDRCs may return symbols of the Hadamard encoding of the evaluation. We show that this is sufficient to ensure consistency on the centers.

In the composed construction of Section 3.9, the satisfiability constraints of the A vertices take the form of a *tree* of comparisons: each petal in an outer sunflower introduces an inner sunflower of its own. All the sub-manifolds of this inner sunflower intersect on the center of the outer sunflower, as well as on a new center. This can be described by a tree in which the D^2 sub-manifolds are leaves and the inner nodes correspond to centers. Each inner node has D children in the tree: one for each petal that intersects on the inner node's center (see Figure 6). Any two sub-manifolds have to be consistent on all the centers that are common ancestors.
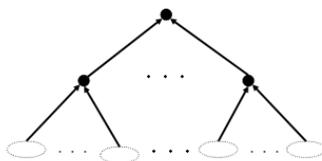
Figure 6: Comparisons tree.

Since the satisfiability constraints of the construction of Section 3.9 are given in the form of a comparisons tree, the analysis of the composition of the construction of Section 3.9 with the construction of Section 3.10 boils down to analyzing the following two-prover game:

The Tree-Path Game. Underlying a tree-path game there is a fixed tree. Each node in the tree may be labeled by a value in $\mathbb{F}$. $\mathbb{L}$ is a subfield of $\mathbb{F}$. The purpose of the verifier is to check whether

two provers agree on a labeling of the nodes. The *tree prover* gets an index i and replies, for each of the nodes in the tree, with the i'th symbol of the Hadamard encoding over $\mathbb{L}$ of a label to the node. The *path prover* gets a leaf in the tree and replies, for each of the nodes on the path from the leaf to the root, with a label in $\mathbb{F}$ to the node. The verifier checks the consistency of the answers it got from the two provers.

3.12 Some Technical Difficulties and Non-technical Subtleties [Or: Why Is The Formal Proof So Long?]

In this section we list some complications that arise in the construction.

Codes and domains. The algebraic construction in Section 3.3 is such that the list decoding may contain polynomials of a slightly larger degree than the Reed-Muller code permits (degree $(k+1) \cdot d$ rather than d). Yet, the set of polynomials of degree at most $(k+1) \cdot d$ is also a Reed-Muller code. To allow the construction to go through, we consider LDRCs that have *two codes* underlying them, instead of one. To facilitate that, we introduce the notion of a *domain*. A domain is composed of two codes of the same length and over the same alphabet: the *encoded code* and the *decoded code*. The decoded code contains the encoded code as a subset. The LDRC has to encode codewords of the encoded code. The LDRC is allowed to use in its list decoding codewords from the decoded code. When we compose, the outer LDRC has to work just as well with labels over the decoded code of the inner LDRC, as it would with labels over the encoded code of the inner LDRC. Hence, our definition of LDRCs needs to be extended to alphabets that are themselves domains.

Consistency between copies. The presentation of composition in Section 3.9 ignored the issue of checking consistency between copies of the same petal. While for different copies we wish to decode different tuples, their labels should be the same. This issue arises from the right degree reduction in Section 3.8. It complicates the construction and its analysis considerably:

1. We change the inner Sunflowers LDRC construction so that the centers of the sunflowers are uniformly distributed in $\mathbb{F}^m$, independently of the tuples being decoded. More on this is in *Regularity and uniformity* below.

2. We change the composition so that copies are compared on the inner centers (which are uniformly distributed on the manifolds, independently of the tuples being decoded).

3. In the analysis of the composition, we show that the comparisons on the inner centers suffice for the consistency check of the outer test.

Regularity and uniformity. The adaptation of the Manifold vs. Point LDRC to almost-linear size results in LDRCs that are just *almost*-right regular, rather then right regular. Crucially, the right degrees depend on the tuples we wish to decode. As a result, when performing right degree

28

reduction as in Section 3.6 and then when obtaining the Sunflowers construction in Section 3.7, the distribution of the centers of the sunflowers, *depends on the tuples we wish to decode*. This dependence is problematic in an inner construction of a composition.

Hence, for the inner construction we consider a uniform right-regular, but polynomial size (rather than almost-linear size), Manifold vs. Point LDRC construction. The inefficiency is tolerable in the context of an inner construction. We go through the construction steps in Sections 3.3, 3.6, 3.7 and 3.8 to prove that uniformity and independence are preserved.

4 Organization of the Construction

We start with some preliminaries about expanders and codes. In Section 6 we define several variants of locally decode/reject codes that are needed for our construction and prove some useful lemmas about them. In Section 7 we define our building blocks, which are locally decode/reject codes with special structure for specific codes (e.g., Reed-Muller, Hadamard). These building blocks lend themselves to various manipulations as described in Section 8. Those manipulations, put in the right order, allow us to construct the locally decode/reject codes we are after in Section 9. The rest of the paper is devoted to implementing and analyzing the different manipulations.

5 Preliminaries

The set of real numbers is $\mathbb{R}$. The set of positive real numbers is $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x > 0\}$. The set of natural numbers is $\mathbb{N} = \{0, 1, 2, \ldots\}$. The set of positive natural numbers is $\mathbb{N}^+ = \{1, 2, \ldots\}$. For a natural number n, we denote $[n] = \{1, \ldots, n\}$. For a string x or a vector $\vec{x}$ of length n and an index $i \in [n]$, we let x_i denote the i'th coordinate of x. All the logarithms in this work are base 2.

5.1 Bipartite Graphs

In this work we refer to bipartite (multi-)graphs $G = (A, B, E)$. The *size* of G is $|G| = |A| + |B| + |E|$ (where $|E|$ is counted with multiplicities). The *left degree* of G is the maximal degree of A vertices, and the *right degree* of G is the maximal degree of B vertices. If all the degrees of the A vertices are equal to the left degree, we say that the graph is *left regular*. If all the degrees of the B vertices are equal to the right degree, we say that the graph is *right regular*. If the graph is both left and right regular with the same degree Δ, we say that it is Δ-regular. We use the notation $\Delta_G(v)$ to denote the degree of a vertex $v \in A \cup B$. It will be convenient to think of the edges touching a vertex as being ordered. For a vertex $v \in A \cup B$ and an index $i \in [\Delta_G(v)]$, let $e_G(v, i) \in E$ denote the i'th edge touching v. For two sets $X \subseteq A$ and $Y \subseteq B$, let $E(X, Y) \doteq \{(x, y) \in E \mid x \in X, y \in Y\}$.

5.2 Expanders

For an undirected (multi-)graph $G = (V, E)$ and two sets of vertices $X, Y \subseteq V$, let $E(X, Y) \doteq \{(x, y) \in E \mid x \in X, y \in Y\}$ (we use the convention that an edge $(x, y) \in E$ with both $x, y \in X, Y$ is taken to the multi-set twice). Roughly speaking, an *expander* is a Δ-regular undirected (multi-)graph $G = (V, E)$ in which the number of edges (with multiplicities) between any two sets of vertices $X, Y \subseteq V$, i.e., $|E(X, Y)|$, is approximately the expected number in a *random* Δ-regular undirected (multi-)graph.

It can be shown that any Δ-regular undirected (multi-)graph whose adjacency matrix has low second largest eigenvalue (in absolute value) λ satisfies this property:

Lemma 5.1 (Expander mixing lemma). *Let $G = (V, E)$ be a Δ-regular undirected (multi-)graph, whose adjacency matrix has second largest eigenvalue (in absolute value) λ. Then, for any two sets $X, Y \subseteq V$,*

$$\left| \frac{|E(X, Y)|}{\Delta |V|} - \frac{|X|}{|V|} \cdot \frac{|Y|}{|V|} \right| \leq \frac{\lambda}{\Delta} \cdot \sqrt{\frac{|X|}{|V|} \cdot \frac{|Y|}{|V|}}$$

We will use the explicit constructions guaranteed by the following lemma (for a proof see Appendix A):

Lemma 5.2 (Explicit construction of expanders). *There is a constant $\alpha < 1$ and a function $T : \mathbb{N} \to \mathbb{N}^+$ with $T(\Delta) = \Theta(\Delta)$, such that given two natural numbers n and Δ, one can find in time polynomial in n and in Δ an undirected (multi-)graph $G = (V, E)$ with $|V| = n$, which is $T(\Delta)$-regular and whose adjacency matrix has second largest eigenvalue (in absolute value) $\lambda \leq (T(\Delta))^\alpha$.*

In this work we will refer to the bipartite versions of expander graphs. Given a (multi-)graph $G = (V, E)$, we define its bipartite version, or its *double cover*, $G' = (V \times \{out\}, V \times \{in\}, E')$ by taking for every edge $e = (x, y) \in E$ the edges $(\langle x, out \rangle, \langle y, in \rangle) \in E'$ and $(\langle y, out \rangle, \langle x, in \rangle) \in E'$. Note that if G is Δ-regular, then so is G'. Combining Lemma 5.1 and Lemma 5.2 we have the following:

Lemma 5.3 (Bipartite expanders). *There is a constant $\alpha < 1$ and a function $T : \mathbb{N} \to \mathbb{N}^+$ with $T(\Delta) = \Theta(\Delta)$, such that given two natural numbers n and Δ, one can find in time polynomial in n and in Δ a $T(\Delta)$-regular undirected bipartite (multi-)graph $G = (V_1, V_2, E)$ with $|V_1| = |V_2| = n$ as follows. For every two sets $X \subseteq V_1$ and $Y \subseteq V_2$,*

$$\left| \frac{|E(X, Y)|}{T(\Delta) \cdot |V_1|} - \frac{|X|}{|V_1|} \cdot \frac{|Y|}{|V_2|} \right| \leq \frac{1}{T(\Delta)^{1-\alpha}} \cdot \sqrt{\frac{|X|}{|V_1|} \cdot \frac{|Y|}{|V_2|}}$$

5.3 Codes

Let Σ be a finite alphabet and let n be a natural number. The *(relative) Hamming distance* between two strings $x, y \in \Sigma^n$ is the fraction of positions on which x and y differ, i.e., $\Pr_{i \in [n]}[x_i \neq y_i]$. Let $0 < \epsilon < 1$. A *code* with (relative) distance $1 - \epsilon$ is a set $C \subseteq \Sigma^n$ such that every two elements

$x, y \in C$ have (relative) Hamming distance at least $1 - \epsilon$. In other words, every two elements $x, y \in C$ *agree* on at most ϵ fraction of the positions. The number n is called the *length* of the code. The elements of C are called the *codewords* of the code. For a set M of size $|M| = |C|$, an *encoding* of M via C is a one-to-one function $E_C : M \to \Sigma^n$ that takes messages from M to codewords in C. When Σ is a field and M is a linear space over Σ, we say that the encoding is *linear* if it is a linear transformation from M to the linear space Σ^n.

Given a string $x \in \Sigma^n$ and a real number $0 < \delta \leq 1$, the δ-*list decoding* of x with respect to C is the set of all codewords $c \in C$ that agree with x on at least a δ fraction of the positions, i.e., $\Pr_{i \in [n]} [c_i = x_i] \geq \delta$. We have the following useful proposition:

Proposition 5.4 (List decoding). *Let $C \subseteq \Sigma^n$ be a code with (relative) distance $1 - \epsilon$. Assume $\delta \geq 2\sqrt{\epsilon}$. Then, for every $x \in \Sigma^n$, the δ-list decoding of x with respect to C contains at most $\frac{2}{\delta}$ codewords.*

Proof. Let $x \in \Sigma^n$. Assume towards a contradiction that the δ-list decoding of x with respect to C contains $l \doteq \lfloor \frac{2}{\delta} \rfloor + 1$ different codewords. Then, when picking a position $i \in [n]$ uniformly at random, the probability that x agrees with one of the codewords on the i'th position is at least $\delta l - \binom{l}{2} \cdot \epsilon > 1$. Contradiction! $\qquad\square$

5.3.1 Some Specific Codes

Some examples of codes that we will use are:

The Reed-Muller Code.   The *Reed-Muller Code* (RM) is defined by a finite field $\mathbb{F}$ and two natural numbers m and d. The code is of length $n = |\mathbb{F}^m|$ over alphabet $\mathbb{F}$. Let us identify the positions $1, \dots, n$ with the points in $\mathbb{F}^m$. Then, for every m-variate polynomial Q of degree at most d over $\mathbb{F}$ we have a codeword $c_Q \in \mathbb{F}^n$. The symbol in the position corresponding to $\vec{x} \in \mathbb{F}^m$ in c_Q is $Q(\vec{x})$. This code is of distance $1 - \frac{d}{|\mathbb{F}|}$.

We identify the following encoding with the Reed-Muller code: let $W \doteq \binom{m+d}{m}$ be the number of monomials in an m-variate polynomial of degree at most d over $\mathbb{F}$. Let $E_{RM} : \mathbb{F}^W \to \mathbb{F}^n$ be the transformation taking a vector in $\mathbb{F}^W$, representing coefficients for the W monomials, to the evaluations of the induced polynomial on the points in $\mathbb{F}^m$. Note that this encoding is linear over $\mathbb{F}$.

The Hadamard Code.   The *Hadamard Code* (Had) is defined by a finite field $\mathbb{F}$ and a natural number m. The code is of length $n = |\mathbb{F}^m|$ over alphabet $\mathbb{F}$. Let us identify the positions $1, \dots, n$ with the points in $\mathbb{F}^m$. Then, for every linear function $L : \mathbb{F}^m \to \mathbb{F}$, i.e., a function of the form $L(x_1, \dots, x_m) = \sum_{i=1}^m a_i x_i$ for some coefficients vector $\vec{a} = (a_1, \dots, a_m) \in \mathbb{F}^m$, we have a codeword $c_L \in \mathbb{F}^n$. The symbol in the position corresponding to $\vec{x} \in \mathbb{F}^m$ in c_L is $L(\vec{x})$. This code is of distance $1 - \frac{1}{|\mathbb{F}|}$.

We identify the following encoding with the Hadamard code: Let $E_{Had} : \mathbb{F}^m \to \mathbb{F}^n$ be the transformation taking a vector in $\mathbb{F}^m$, which is the coefficients vector of a linear function, to the

evaluations of the linear function on the points in $\mathbb{F}^m$. Note that this encoding is linear over $\mathbb{F}$.

5.3.2 Transformation on Codes

We can produce new codes from existing ones by using transformations such as repetition or concatenation:

Repetition. Assume that $C \subseteq \Sigma^n$ is a code. Let l be a natural number. Then, the *l-repetition* of C is the code $C^l \subseteq \Sigma^{n \cdot l}$ defined by taking l copies of each symbol in a codeword. In other words, for every codeword $c = c_1 \cdots c_n \in C$ we have the codeword

$$\underbrace{c_1 \cdots c_1}_{l} \cdots \underbrace{c_n \cdots c_n}_{l} \in C^l$$

If C has (relative) distance $1 - \epsilon$, then so does C^l.

Concatenation. Assume that $C_1 \subseteq \Sigma^n$ is a code and that $C_2 \subseteq \Gamma^k$ is a code that has $|C_2| = |\Sigma|$ codewords. Assume that C_1 is associated with an encoding $E_{C_1} : M \to \Sigma^n$ and that C_2 is associated with an encoding $E_{C_2} : \Sigma \to \Gamma^k$. The concatenation of C_1 and C_2, denoted $C_1 \diamond C_2 \subseteq \Gamma^{n \cdot k}$, is the code obtained when encoding each symbol of a C_1 codeword by E_{C_2}, i.e., we define $E_{C_2}^n : \Sigma^n \to \Gamma^{n \cdot k}$ such that $E_{C_2}^n(\sigma_1 \cdots \sigma_n) \doteq E_{C_2}(\sigma_1) \cdots E_{C_2}(\sigma_n)$. The encoding associated with the concatenation $C_1 \diamond C_2$ is $E_{C_2}^n \circ E_{C_1} : M \to \Gamma^{k \cdot n}$. C_1 is called *the outer code*, and C_2 is called *the inner code*. If C_1 has (relative) distance $1 - \epsilon_1$ and C_2 has (relative) distance $1 - \epsilon_2$, then $C_1 \diamond C_2$ has (relative) distance $1 - (\epsilon_1 + \epsilon_2 - \epsilon_1 \epsilon_2)$. Code concatenation is used to reduce the alphabet of a code C_1 (from Σ to the typically much smaller Γ) at the cost of slightly enlarging the length (from n to $n \cdot k$) and damaging the distance.

If E_{C_1} is linear and E_{C_2} is linear, then we can view M as a linear space over Γ (by defining for $\lambda \in \Gamma$ and $v \in M$, the scalar multiplication to be $\lambda \cdot v \doteq (\lambda \cdot 1) \cdot v$ where $1 \in \Sigma$ (recall that Σ is a field). Note that $\lambda \cdot 1 \in \Sigma$ is well-defined, and so is $(\lambda \cdot 1) \cdot v \in M$). Using this perspective, the encoding $E_{C_2}^n \circ E_{C_1}$ corresponding to the concatenation $C_1 \diamond C_2$ becomes linear over Γ.

6 Graph Theoretic Formulation

6.1 Bipartite Constraint Graphs

We formalize two query constraint graphs with projection property as bipartite graphs $G = (A, B, E)$, where vertices in A determine values for vertices in B. Vertices in A are assigned values from an alphabet Σ_A, while vertices in B are assigned values from an alphabet Σ_B. Every edge $e = (a, b) \in E$ is associated with an element ξ from some set Ω, where ξ is called the *label* of the edge. Every assignment $\sigma_a \in \Sigma_A$ for a determines a single assignment for b given by $proj(a, \sigma_a, \xi) \in \Sigma_B$. The test

associated with the edge e consists of comparing whether the assignment for b equals $proj(a, \sigma_a, \xi)$, as well as of a satisfiability check $sat(a, \sigma_a)$ that checks the validity of σ_a. In particular, $sat(a, \sigma_a)$ may check some consistency condition between $proj(a, \sigma_a, \xi)$ for different ξ's.

Definition 6.1 (Bipartite constraint graph). $\mathcal{G} = \langle G, \Omega, \Sigma_A, \Sigma_B, sat, label, proj \rangle$ *is called a* bipartite constraint graph, *if*

1. $G = (A, B, E)$ *is a bipartite (multi-)graph,* Ω *is a finite set and* Σ_A, Σ_B *are finite sets.*

2. $sat : A \times \Sigma_A \to \{true, false\}$ *is a function (sat determines for each vertex $a \in A$ whether it is satisfied under the given assignment).*

3. $label : E \to \Omega$ *is a function (label assigns every edge some element in Ω).*

4. $proj : A \times \Sigma_A \times \Omega \to \Sigma_B$ *is a function (for every vertex $a \in A$ and assignment for it $\sigma_a \in \Sigma_A$, proj gives, for every $\xi \in \Omega$, the "projection" of σ_a to ξ).*

We say that an edge $e = (a, b) \in E$ is satisfied *in $\mathcal{G}$ under assignments $\sigma_a \in \Sigma_A$ and $\sigma_b \in \Sigma_B$, if $sat(a, \sigma_a) = true$ and $proj(a, \sigma_a, label(e)) = \sigma_b$.*

We say that an edge $e = (a, b) \in E$ is satisfied *in $\mathcal{G}$ under assignments $C_A : A \to \Sigma_A$ and $C_B : B \to \Sigma_B$, if e is satisfied in $\mathcal{G}$ under the assignments $C_A(a)$ and $C_B(b)$.*

When there are several constraint graphs involved we sometimes use subscripts to distinguish between functions corresponding to different graphs. E.g., we write $sat_{\mathcal{G}}$ to refer to sat of $\mathcal{G}$.

6.2 Bipartite Locally Decode/Reject Codes

We formulate locally decode/reject codes that make two queries with a projection property as bipartite graphs. But, before we do that, let us define *domains*. Domains capture the sets of messages we encode and decode. It is not a standard notion, but it will be very convenient for us in the sequel.

6.2.1 Domains

The messages we encode will sometimes be strings in Σ^n for a finite alphabet Σ and a length n, and sometimes be codewords themselves, e.g., Reed-Muller codewords or Hadamard codewords. We let $\mathcal{D}_{enc}$ denote the set of messages we encode.

For the decoding we allow to use messages from a set that is possibly larger than the set of messages we encode. For example, suppose we encode Reed-Muller codewords corresponding to polynomials of degree at most d for some natural number d. Then, we may consider as an appropriate decoding a codeword that corresponds to a polynomial of a slightly larger degree d'. In general, we let $\mathcal{D}_{dec}$ be the set of possible decodings, where $\mathcal{D}_{dec} \supseteq \mathcal{D}_{enc}$. Equality $\mathcal{D}_{enc} = \mathcal{D}_{dec}$ may hold, but is not required.

For notational convenience we refer to messages as functions, rather than strings. Strings in Σ^n can be thought of as functions $[n] \to \Sigma$, while the codes we use naturally give rise to functions. For

instance, the Reed-Muller code gives rise to low degree polynomials, and the Hadamard code gives rise to linear functions.

Formally, a *domain* is defined as follows:

Definition 6.2 (Domain). *A domain $\mathcal{D}$ is a tuple $\langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec}\rangle$, where D and R are finite sets, and $\mathcal{D}_{enc} \subseteq \mathcal{D}_{dec}$ are sets of functions $D \to R$. $\mathcal{D}_{enc}$ is called* the encoded domain*, and $\mathcal{D}_{dec}$ is called* the decoded domain.

Some particular domains of interest in this work are:

Σ. With any finite set Σ we associate a domain $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec}\rangle$ that corresponds to encoding and decoding symbols in Σ. The domain consists of $D = \{1\}$, $R = \Sigma$ and $\mathcal{D}_{enc} = \mathcal{D}_{dec} = \{f \mid f : \{1\} \to \Sigma\}$.

$\Sigma^{\mathbf{n}}$. With any set Σ^n for a finite set Σ and a natural number n, we associate a domain $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec}\rangle$ that corresponds to encoding and decoding of strings in Σ^n. The domain consists of $D = [n]$, $R = \Sigma$ and $\mathcal{D}_{enc} = \mathcal{D}_{dec} = \{f \mid f : [n] \to \Sigma\}$ (the previous item is a special case for $n = 1$).

Reed-Muller Code. A Reed-Muller domain $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec}\rangle$ corresponds to encoding and decoding of Reed-Muller codewords, where the code we use for the decoding contains the code used for the encoding. The domain is defined by a finite field $\mathbb{F}$, a natural number m and two natural numbers $0 < d \leq d' < |\mathbb{F}|$. We take $D = \mathbb{F}^m$ and $R = \mathbb{F}$. $\mathcal{D}_{enc}$ is the set of all m-variate polynomials of degree at most d over $\mathbb{F}$, while $\mathcal{D}_{dec}$ is the set of all m-variate polynomials of degree at most d' over $\mathbb{F}$. The number m is called the *dimension*. The degree d is called the *encoding degree* and the degree d' is called the *decoding degree*.

Hadamard Code. A Hadamard domain $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec}\rangle$ corresponds to encoding and decoding of Hadamard codewords. The domain is defined by a finite field $\mathbb{F}$ and a natural number m. We take $D = \mathbb{F}^m$ and $R = \mathbb{F}$. The encoded and decoded domains are the same $\mathcal{D}_{enc} = \mathcal{D}_{dec}$ and equal to the set of all linear functions $L : \mathbb{F}^m \to \mathbb{F}$. The number m is called the *dimension*.

Concatenation of Reed-Muller and Hadamard Codes. A $RM \diamond Had$ domain $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec}\rangle$ corresponds to encoding and decoding codewords in a code which is a concatenation of a Reed-Muller code and a Hadamard code. The domain is defined by a finite field $\mathbb{F}$, a subfield $\mathbb{L}$ of $\mathbb{F}$, a natural number m and two natural numbers $0 < d \leq d' < |\mathbb{F}|$. Denote the extension degree of $\mathbb{F}$ over $\mathbb{L}$ by $\tau = [\mathbb{F} : \mathbb{L}]$. Let ϕ be a linear bijection from $\mathbb{F}$ (viewed as a linear space over the field $\mathbb{L}$) to the linear space of linear functions $\mathbb{L}^\tau \to \mathbb{L}$ (note that there are $|\mathbb{L}^\tau| = |\mathbb{F}|$ linear functions $\mathbb{L}^\tau \to \mathbb{L}$). The domain will correspond to encoding each symbol in a Reed-Muller codeword by ϕ. We take $D = \mathbb{F}^m \times \mathbb{L}^\tau$ and $R = \mathbb{L}$.

- The encoded domain $\mathcal{D}_{enc}$ is the set of all functions $f : \mathbb{F}^m \times \mathbb{L}^\tau \to \mathbb{L}$ of the form $f(\vec{x}_1, \vec{x}_2) = \phi(Q(\vec{x}_1))(\vec{x}_2)$ for some polynomial $Q : \mathbb{F}^m \to \mathbb{F}$ of degree at most d.

- The decoded domain $\mathcal{D}_{dec}$ is the set of all functions $f : \mathbb{F}^m \times \mathbb{L}^\tau \to \mathbb{L}$ of the form $f(\vec{x}_1, \vec{x}_2) = \phi(Q(\vec{x}_1))(\vec{x}_2)$ for some polynomial $Q : \mathbb{F}^m \to \mathbb{F}$ of degree at most d'.

6.2.2 Bipartite Evaluation Graphs

Fix a domain $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ that defines the messages we encode and decode. Let k and N be natural numbers. Fix a collection of k-tuples of positions in a message we wish to decode:

$$\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$$

We formulate a bipartite evaluation graph as a bipartite constraint graph $G = (A, B, E)$, in which tuples $\langle x_{i,1}, \ldots, x_{i,k} \rangle$ are associated with vertices that are "responsible" for evaluating f on them. Either the A vertices or the B vertices get associated with tuples. The set of vertices that are associated with tuples is denoted $V \in \{A, B\}$, and we refer to them as *the evaluating vertices*. The tuple that is associated with a vertex $v \in V$ is denoted $tup(v)$. For an evaluating vertex $v \in V$ and an assignment $\sigma_v \in \Sigma_V$ to v, an evaluation on the tuple $tup(v)$, namely, k values in R, is given by a function $eval(v, \sigma_v)$. All tuples should be associated with the same number of evaluating vertices, and all evaluating vertices must have the same degree in the graph.

Formally we define bipartite evaluation graphs as follows:

Definition 6.3 (Bipartite evaluation graph). *Let $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a domain. Let k and N be natural numbers. Assume a collection of k-tuples:*

$$\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$$

$\mathcal{G} = \langle G = (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat, label, proj, tup, eval \rangle$ *is called a* bipartite evaluation graph *for the k-tuples, if*

1. *$\mathcal{G}' = \langle G, \Omega, \Sigma_A, \Sigma_B, sat, label, proj \rangle$ is a bipartite constraint graph.*

2. *$V \in \{A, B\}$ is a set of* evaluating vertices. *All the V vertices have the same degree in the graph G.*

3. *$tup : V \to D^k$ is a function mapping each evaluating vertex to a k-tuple $\langle x_{i,1}, \ldots, x_{i,k} \rangle$ for $i \in [N]$. Each $i \in [N]$ must have the same (positive) number of vertices $v \in V$ mapped to $\langle x_{i,1}, \ldots, x_{i,k} \rangle$.*

4. *$eval : V \times \Sigma_V \to R^k$ is a function, mapping each evaluating vertex $v \in V$ with an assignment for it to assignments for the elements of $tup(v)$.*

We say that a vertex $v \in V$ with $tup(v) = \langle x_{i,1}, \ldots, x_{i,k} \rangle \in D^k$ reads $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under an assignment $\sigma_v \in \Sigma_V$, if $eval(v, \sigma_v) = \langle f(x_{i,1}), \ldots, f(x_{i,k}) \rangle$. We say that an edge $e = (a, b) \in E$

reads $f \in \mathcal{D}_{dec}$ *in $\mathcal{G}$ under assignments $\sigma_a \in \Sigma_A$ and $\sigma_b \in \Sigma_B$, if the evaluating vertex it touches* $v \in \{a, b\} \cap V$ *reads f in $\mathcal{G}$ under the assignment σ_v. We say that an edge $e = (a, b) \in E$ reads* $f \in \mathcal{D}_{dec}$ *in $\mathcal{G}$ under assignments $C_A : A \rightarrow \Sigma_A$ and $C_B : B \rightarrow \Sigma_B$, if e reads f in $\mathcal{G}$ under the assignments $C_A(a)$ and $C_B(b)$.*

We say that an edge $e \in E$ is satisfied *in $\mathcal{G}$ under assignments $C_A : A \rightarrow \Sigma_A$ and $C_B : B \rightarrow \Sigma_B$, if e is satisfied in $\mathcal{G}'$ under C_A and C_B.*

If $V = A$, we say that $\mathcal{G}$ is a left evaluator. *Otherwise, we say that it is a* right evaluator.

6.2.3   Bipartite Locally Decode/Reject Codes for k-Tuples

Fix a bipartite evaluation graph as in Definition 6.3. An encoding of a message is given by assignments to the vertices $C_A : A \rightarrow \Sigma_A$ and $C_B : B \rightarrow \Sigma_B$. Given assignments C_A and C_B, local decode/reject is done as follows:

1. Pick an edge $e = (a, b) \in E$ uniformly at random. Let $v \in \{a, b\} \cap V$ be the evaluating vertex that e touches.

2. Check the satisfiability constraint on the edge. If e is not satisfied under C_A and C_B, *reject*.

3. Otherwise, return "the evaluation of $tup(v)$ is $eval(v, C_V(v))$".

Note that $tup(v)$ is $\langle x_{i,1}, \ldots, x_{i,k} \rangle$ for a uniformly distributed $i \in [N]$.

For every message $f \in \mathcal{D}_{enc}$ one should be able to efficiently compute assignments to the A and B vertices such that the local decode/reject procedure never rejects and always evaluates f. That is, all edges are satisfied and read f.

Given an assignment to the B vertices, there should be a short list decoding $f_1, \ldots, f_l \in \mathcal{D}_{dec}$, such that, for every assignment to the A vertices, with high probability, the local decode/reject procedure either rejects or evaluates one of $f_1, \ldots, f_l$. That is, for almost all edges, either the edge is not satisfied, or the edge reads one of $f_1, \ldots, f_l$.

Note that we require the list decoding $f_1, \ldots, f_l \in \mathcal{D}_{dec}$ to depend only on the assignment to the B vertices, and not on the assignment to the A vertices. That is, given the assignment to the B vertices, there is a single list decoding $f_1, \ldots, f_l$ that works *for all* assignments to the A vertices. Conceptually, one can give this requirement the following meaning: The assignment to the B vertices alone is already an encoding of the message (and thus sufficient for decoding), while the assignment to the A vertices provides additional information that is needed for the purpose of *local* decode/reject.

Two parameters determine the quality of list decoding in the codes. One is δ_{min} which lower bounds the error of the decoding, namely, the probability that the decoding procedure does not reject, yet its evaluation does not correspond to the elements of the list decoding. The other is l_{max} which upper bounds the size of the list decoding. This size is a (decreasing) function of the error $\delta \geq \delta_{min}$ we are willing to settle for.

The formal definition is as follows:

Definition 6.4 (Bipartite locally decode/reject code). *Let* $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ *be a domain. Let* k *and* N *be natural numbers. Assume a collection of* k*-tuples:*

$$\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$$

Let $0 < \delta_{min} < 1$. *Let* $l_{max} : (0,1) \to \mathbb{R}^+$ *be a decreasing function. A bipartite evaluation graph* $\mathcal{G} = \langle G = (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat, label, proj, tup, eval \rangle$ *for the* k*-tuples is called a* (δ_{min}, l_{max})*-bipartite locally decode/reject code for the* k*-tuples, if the following holds:*

1. ***Encoding:*** *There is an efficient algorithm that given a message* $f \in \mathcal{D}_{enc}$, *computes assignments* $C_A : A \to \Sigma_A$ *and* $C_B : B \to \Sigma_B$, *such that every edge* $e = (a, b) \in E$ *is satisfied and reads* f *in* $\mathcal{G}$ *under* C_A, C_B.

2. ***List Decoding:*** *For every assignment* $C_B : B \to \Sigma_B$, *for every real* δ *such that* $\delta_{min} \leq \delta < 1$, *there exist* $l \leq l_{max}(\delta)$ *elements* $f_1, \ldots, f_l \in \mathcal{D}_{dec}$, *such that for every assignment* $C_A : A \to \Sigma_A$ *the following holds: when picking uniformly at random an edge* $e = (a, b) \in E$, *the probability that in* $\mathcal{G}$ *under* C_A, C_B, *the edge* e *is satisfied, although* e *does not read any of* $f_1, \ldots, f_l$, *is at most* $O(\delta)$.

6.2.4 Composable Bipartite Locally Decode/Reject Codes

We strengthen the definition of bipartite locally decode/reject codes (Definition 6.4) with the intent of allowing composition of codes. In the strengthened definition the alphabets are domains themselves,

$$\Sigma_A = \langle D_A, R_A, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle \quad \Sigma_B = \langle D_B, R_B, \Sigma_{B,enc}, \Sigma_{B,dec} \rangle$$

Encoding of a message comprises assignments over the encoded domains of the alphabets $\Sigma_{A,enc}$ and $\Sigma_{B,enc}$. On the other hand, the list decoding property should hold even given assignments over the decoded domains of the alphabets $\Sigma_{A,dec}$ and $\Sigma_{B,dec}$. The previous definition can be seen as a special case in which the encoded and decoded domains of the alphabets are equal, namely $\Sigma_{A,enc} = \Sigma_{A,dec}$ and $\Sigma_{B,enc} = \Sigma_{B,dec}$.

Adding elements to the decoded domains of the alphabets makes the task of the decoder harder, since it needs to succeed on more assignments. In contrast, adding elements to the decoded domain $\mathcal{D}_{dec}$ makes the task of the decoder easier, since it can use these elements in the decoding as well. Usually there will be a correspondence between the decoded domains of the alphabets and the decoded domain $\mathcal{D}_{dec}$, so whenever the decoder needs to succeed in decoding more assignments, it has more elements it can use in the decoding.

Definition 6.5 (Composable bipartite locally decode/reject code). *Let* $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ *be a domain. Let* k *and* N *be natural numbers. Assume a collection of* k*-tuples:*

$$\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$$

Let $0 < \delta_{min} < 1$. *Let* $l_{max} : (0,1) \to \mathbb{R}^+$ *be a decreasing function.*

$\mathcal{G} = \langle G = (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat, label, proj, tup, eval \rangle$ *is called a* (δ_{min}, l_{max})-composable bipartite locally decode/reject code *for the k-tuples, if:*

- $\Sigma_A = \langle D_A, R_A, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$ *and* $\Sigma_B = \langle D_B, R_B, \Sigma_{B,enc}, \Sigma_{B,dec} \rangle$ *are domains.*

- $\mathcal{G}' = \langle G = (A, B, E), V, \Omega, \Sigma_{A,dec}, \Sigma_{B,dec}, sat, label, proj, tup, eval \rangle$ *is a bipartite evaluation graph for the k-tuples. We say that an edge* $e \in E$ *is satisfied in* $\mathcal{G}$ *under assignments* $C_A : A \to \Sigma_{A,dec}$ *and* $C_B : B \to \Sigma_{B,dec}$, *if* e *is satisfied in* $\mathcal{G}'$ *under the assignments* C_A *and* C_B. *We say that an edge* $e \in E$ *reads* $f \in \mathcal{D}_{dec}$ *in* $\mathcal{G}$ *under assignments* $C_A : A \to \Sigma_{A,dec}$ *and* $C_B : B \to \Sigma_{B,dec}$, *if* e *reads* $f \in \mathcal{D}_{dec}$ *in* $\mathcal{G}'$ *under the assignments* C_A *and* C_B.

- *The following holds:*

  1. **Encoding:** *There is an efficient algorithm that given a message* $f \in \mathcal{D}_{enc}$, *computes assignments* $C_A : A \to \Sigma_{A,enc}$ *and* $C_B : B \to \Sigma_{B,enc}$, *such that every edge* $e = (a, b) \in E$ *is satisfied and reads* f *in* $\mathcal{G}$ *under* C_A, C_B.

  2. **List Decoding:** *For every assignment* $C_B : B \to \Sigma_{B,dec}$, *for every real* δ *such that* $\delta_{min} \leq \delta < 1$, *there exist* $l \leq l_{max}(\delta)$ *elements* $f_1, \ldots, f_l \in \mathcal{D}_{dec}$, *such that for any assignment* $C_A : A \to \Sigma_{A,dec}$, *the following holds: when picking uniformly at random an edge* $e = (a, b) \in E$, *the probability that, in* $\mathcal{G}$ *under* C_A, C_B, *the edge* e *is satisfied, although* e *does not read any of* $f_1, \ldots, f_l$, *is at most* $O(\delta)$.

We sometimes omit the specification of δ_{min} and l_{max}, when we do not wish to relate to them.

Properties. We consider the following properties of $\mathcal{G}$:

Size. The *size* of $\mathcal{G}$ is the size of G.

Alphabet size and block length. The *alphabet size* of the A vertices is $|\Sigma_{A,enc}|$. The *alphabet size* of the B vertices is $|\Sigma_{B,enc}|$. The *block length* of the A vertices is $\log |\Sigma_{A,enc}|$. The *block length* of the B vertices is $\log |\Sigma_{B,enc}|$. The *alphabet size* of $\mathcal{G}$ is the maximum between the alphabet size of the A vertices and the alphabet size of the B vertices (which is typically the alphabet size of the A vertices). The *block length* of $\mathcal{G}$ is the maximum between the block length of the A vertices and the block length of the B vertices (which is typically the block length of the A vertices).

Graph degrees. The *left degree* of $\mathcal{G}$ is the left degree of G. The *right degree* of $\mathcal{G}$ is the right degree of G. $\mathcal{G}$ is *left regular* if G is. $\mathcal{G}$ is *right regular* if G is.

6.2.5 Construction Algorithms for Composable Bipartite Locally Decode/Reject Codes

Let $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a domain. Let k and N be natural numbers. A $(\mathcal{D}, k, N)$-*construction algorithm* for composable bipartite locally decode/reject codes is a procedure that given as input a collection of size N of k-tuples of points

$$\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$$

outputs a composable bipartite locally decode/reject code for those tuples. The construction algorithm is said to be *efficient*, if its running time is polynomial in $|D|$, $|R|$, k and N.

We define several *uniformity* properties that construction algorithms may or may not have. By *uniformity* we refer to properties that are common to all outputs of a construction algorithm, independently of the k-tuples given as input to the algorithm.

Uniform in structure. Fix finite sets A, B and $V \in \{A, B\}$. Fix a finite set Ω. Fix domains Σ_A and Σ_B. A $(\mathcal{D}, k, N)$-construction algorithm is said to be *uniform in structure* $\langle A, B, V, \Omega, \Sigma_A, \Sigma_B \rangle$, if on all inputs its output has the same vertex set A, the same vertex set B, the same evaluating vertices V, the same label set Ω and the same alphabet domains Σ_A and Σ_B.

When the identity of A, B, V, Ω, Σ_A and Σ_B is inessential, we simply say that the algorithm is *uniform in structure* without specifying them.

Uniform in tuple association. Fix finite sets A, B and $V \in \{A, B\}$. Fix a finite set Ω. Fix domains Σ_A and Σ_B. Let tupi $: V \to [N]$ be a function (called a *uniform tuple associator*). The function tupi assigns every evaluating vertex an index of an input tuple.

A $(\mathcal{D}, k, N)$-construction algorithm is said to be *uniform in the tuple association* tupi, if it is uniform in structure $\langle A, B, V, \Omega, \Sigma_A, \Sigma_B \rangle$ and on all input tuples, the tup function of its output is as follows: for every vertex $v \in V$, the tuple $tup(v)$ is the i'th input tuple for index $i = $ tupi(v).

When the identity of the uniform tuple associator tupi is inessential, we simply say that the algorithm is *uniform in the tuple association*, without specifying tupi.

Uniform in encoding and list decoding. Fix finite sets A, B and $V \in \{A, B\}$. Fix a finite set Ω. Fix domains Σ_A and Σ_B. Assume the following:

- $\mathcal{E}$: an efficient algorithm (called a *uniform encoder*) that given a message $f \in \mathcal{D}_{enc}$ computes an assignment $C_B : B \to \Sigma_{B,enc}$.

- $\mathcal{L}$: an algorithm (called a *uniform list decoder*) that given an assignment $C_B : B \to \Sigma_{B,dec}$ and a real parameter δ computes a sequence of messages $f_1, \ldots, f_l \in \mathcal{D}_{dec}$.

Notably, both $\mathcal{E}$ and $\mathcal{L}$ are independent of the tuples on which we evaluate.

A $(\mathcal{D}, k, N)$-construction algorithm is said to be *uniform in the encoding $\mathcal{E}$ and in the list decoding $\mathcal{L}$*, if:

1. It is uniform in structure $\langle A, B, V, \Omega, \Sigma_A, \Sigma_B \rangle$.

2. There is an efficient algorithm $\mathcal{E}'$ that given a message $f \in \mathcal{D}_{enc}$ and a collection of k-tuples of points $\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$, computes an assignment $C_A : A \to \Sigma_{A,enc}$.

3. Given as input a collection of k-tuples of points $\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$, the output $\mathcal{G}$ of the construction algorithm for those tuples is a (δ_{min}, l_{max})-composable bipartite locally decode/reject code that satisfies the following encoding and list decoding properties:

   Uniform Encoding: Let $f \in \mathcal{D}_{enc}$. Invoke $\mathcal{E}'$ on f and $\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle$ to compute an assignment $C_A : A \to \Sigma_{A,enc}$. Invoke $\mathcal{E}$ on f to compute an assignment $C_B : B \to \Sigma_{B,enc}$ (independent of the input tuples). Then, every edge $e = (a, b) \in E$ is satisfied and reads f under C_A and C_B.

   Uniform List Decoding: Let $C_B : B \to \Sigma_{B,dec}$ and let δ be a real parameter such that $\delta_{min} \leq \delta < 1$. Invoke $\mathcal{L}$ on C_B and δ to compute $l \leq l_{max}(\delta)$ messages $f_1, \ldots, f_l \in \mathcal{D}_{dec}$ (independent of the input tuples). Then, for every assignment $C_A : A \to \Sigma_{A,dec}$ the following holds: When picking uniformly at random an edge $e = (a, b) \in E$, the probability that, in $\mathcal{G}$ under C_A, C_B, the edge e is satisfied, although e does not read any of $f_1, \ldots, f_l$, is at most $O(\delta)$.

When the identities of the uniform encoder $\mathcal{E}$ and the uniform list decoder $\mathcal{L}$ are inessential, we simply say that the algorithm is *uniform in encoding and list decoding*, without specifying $\mathcal{E}$ and $\mathcal{L}$.

6.2.6 Point Variant of Composable Bipartite Locally Decode/Reject Codes

We define a variant of composable bipartite locally decode/reject codes (see Definition 6.5). In this variant, not only the V vertices are meant to evaluate a function on k-tuples of points in D, but also the vertices on the other side (which will be denoted $W \in \{A, B\}$) are meant to evaluate the same function on points in D. Each vertex $v \in W$ has a point $pnt(v) \in D$ associated with it. For an assignment σ_v to v, the evaluation of v on the point is given by $evalp(v, \sigma_v) \in R$. All the points in D have the same number of W vertices associated with them. We require nothing about the joint distribution of the tuples and the points.

Each edge touches one V vertex and one W vertex. We strengthen the definition of reading, so for an edge to read a function $f \in \mathcal{D}_{dec}$, the V vertex must evaluate f on its tuple and the W vertex must evaluate f on its point. We require all the vertices in W to be of the same degree, so a uniformly distributed edge touches a vertex associated with a uniformly distributed point $p \in D$.

Analogous to a bipartite evaluation graph we define:

Definition 6.6 (Bipartite tuple-point evaluation graph). *Let $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a domain. Let k and N be natural numbers. Assume a collection of k-tuples:*

$$\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$$

$\mathcal{G} = \langle G = (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat, label, proj, tup, eval, pnt, evalp \rangle$ *is called a* bipartite tuple-point evaluation graph *for the k-tuples, if*

1. $\mathcal{G}' = \langle G, V, \Omega, \Sigma_A, \Sigma_B, sat, label, proj, tup, eval \rangle$ *is a bipartite evaluation graph for the k-tuples.*

2. *Denote the vertices on the other side of V by $W \doteq (A \cup B) \setminus V \in \{A, B\}$. All the W vertices must have the same degree in G.*

3. $pnt : W \to D$ *is a function mapping each vertex in W to a point in D. Each point $p \in D$ must have the same (positive) number of vertices $v \in W$ mapped to it.*

4. $evalp : W \times \Sigma_W \to R$ *is a function, mapping each vertex $v \in W$ with an assignment for it to an assignment for $pnt(v)$.*

We say that a vertex $v \in W$ with $pnt(v) = p \in D$ reads $f \in \mathcal{D}_{dec}$ *in $\mathcal{G}$ under an assignment $\sigma_v \in \Sigma_W$, if $evalp(v, \sigma_v) = f(p)$. We say that an edge $e = (a, b) \in E$* reads $f \in \mathcal{D}_{dec}$ *in $\mathcal{G}$ under assignments $\sigma_a \in \Sigma_A$ and $\sigma_b \in \Sigma_B$, if e reads f in $\mathcal{G}'$ under the assignments σ_a and σ_b (as in Definition 6.3) and $v \in \{a, b\} \cap W$ reads f in $\mathcal{G}$ under the assignment σ_v. We say that an edge $e = (a, b) \in E$* reads $f \in \mathcal{D}_{dec}$ *in $\mathcal{G}$ under assignments $C_A : A \to \Sigma_A$ and $C_B : B \to \Sigma_B$, if e reads f in $\mathcal{G}$ under the assignments $C_A(a)$ and $C_B(b)$.*

We say that an edge $e \in E$ is satisfied *in $\mathcal{G}$ under assignments $C_A : A \to \Sigma_A$ and $C_B : B \to \Sigma_B$, if e is satisfied in $\mathcal{G}'$ under C_A and C_B.*

The point variant of composable bipartite locally decode/reject codes is defined similarly to Definition 6.5, with the bipartite tuple-point evaluation graph underlying it.

Definition 6.7 (Composable bipartite locally decode/reject code (point variant)). *Let $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a domain. Let k and N be natural numbers. Assume a collection of k-tuples:*

$$\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$$

Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function.

$\mathcal{G} = \langle G = (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat, label, proj, tup, eval, pnt, evalp \rangle$ *is called a (δ_{min}, l_{max})-composable bipartite locally decode/reject code (point variant) for the k-tuples, if we have that:*

- $\Sigma_A = \langle D_A, R_A, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$ *and $\Sigma_B = \langle D_B, R_B, \Sigma_{B,enc}, \Sigma_{B,dec} \rangle$ are domains.*

- $\mathcal{G}' = \langle G, V, \Omega, \Sigma_{A,dec}, \Sigma_{B,dec}, sat, label, proj, tup, eval, pnt, evalp \rangle$ *is a bipartite tuple-point evaluation graph for the k-tuples. We say that an edge $e \in E$ is satisfied in $\mathcal{G}$ under assignments $C_A : A \to \Sigma_{A,dec}$ and $C_B : B \to \Sigma_{B,dec}$, if e is satisfied in $\mathcal{G}'$ under the assignments C_A and C_B. We say that an edge $e \in E$ reads $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under assignments $C_A : A \to \Sigma_{A,dec}$ and $C_B : B \to \Sigma_{B,dec}$, if e reads $f \in \mathcal{D}_{dec}$ in $\mathcal{G}'$ under the assignments C_A and C_B.*

- *The following holds:*

41

1. **Encoding:** *There is an efficient algorithm that given a message $f \in \mathcal{D}_{enc}$, computes assignments $C_A : A \to \Sigma_{A,enc}$ and $C_B : B \to \Sigma_{B,enc}$, such that every edge $e = (a, b) \in E$ is satisfied and reads f in $\mathcal{G}$ under C_A, C_B.*

2. **List Decoding:** *For every assignment $C_B : B \to \Sigma_{B,dec}$, for every real δ such that $\delta_{min} \leq \delta < 1$, there exist $l \leq l_{max}(\delta)$ elements $f_1, \ldots, f_l \in \mathcal{D}_{dec}$, such that for every assignment $C_A : A \to \Sigma_{A,dec}$ the following holds: when picking uniformly at random an edge $e = (a, b) \in E$, the probability that, in $\mathcal{G}$ under C_A, C_B, the edge e is satisfied, although e does not read any of $f_1, \ldots, f_l$, is at most $O(\delta)$.*

Note that $\mathcal{G}^- = \langle G, V, \Omega, \Sigma_A, \Sigma_B, sat, label, proj, tup, eval \rangle$ is a composable bipartite locally decode/reject code. *We refer to it as the composable bipartite locally decode/reject code induced by $\mathcal{G}$.*

6.2.7 Construction Algorithms for The Point Variant of Composable Bipartite Locally Decode/Reject Codes

Let $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a domain. Let k and N be natural numbers. A $(\mathcal{D}, k, N)$-*construction algorithm* for composable bipartite locally decode/reject codes (point variant) is a procedure that given as input a collection of size N of k-tuples of points

$$\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$$

outputs a composable bipartite locally decode/reject code (point variant) for those tuples. Efficiency and uniformity in structure of such algorithms are as for construction algorithms for composable bipartite locally decode/reject codes.

For the point variant we are interested in an additional uniformity property that construction algorithms may or may not have:

Uniform in point association. Fix finite sets A, B and $V \in \{A, B\}$. Fix a finite set Ω. Let $W = (A \cup B) \setminus V$. Fix domains Σ_A and Σ_B. Let $pnt : W \to D$ be a function (called a *uniform point associator*).

A $(\mathcal{D}, k, N)$-construction algorithm for bipartite locally decode/reject codes (point variant) is said to be *uniform in the point association* pnt, if it is uniform in structure $\langle A, B, V, \Omega, \Sigma_A, \Sigma_B \rangle$ and on all input tuples, the output of the algorithm has as its pnt function the uniform point associator pnt.

When the identity of the uniform point associator pnt is inessential, we simply say that the algorithm is *uniform in the point association*, without specifying pnt.

6.2.8 List Decoding Based on Point Evaluations

Recall that $\mathcal{D}_{dec}$ defines a code as follows: for every $f \in \mathcal{D}_{dec}$ there is a codeword with $|D|$ co-ordinates, where the symbol in position $x \in D$ is $f(x)$. Provided that $\mathcal{D}_{dec}$ defines a code with

large (relative) distance, a list decoding can be computed based solely on the point evaluations. The following lemma shows that and relates the list decoding to the list decoding guaranteed in the definition of the codes:

Lemma 6.8 (List decoding for point evaluations). *Let $0 < \epsilon < 1$. Fix a domain $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$. Suppose that $\mathcal{D}_{dec}$ defines a code with (relative) distance $1 - \epsilon$. Let k and N be natural numbers. Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function. Let $\mathcal{A}$ be a $(\mathcal{D}, k, N)$-construction algorithm that outputs (δ_{min}, l_{max})-composable bipartite locally decode/reject codes (point variant). Assume that $\mathcal{A}$ is uniform in structure $\langle A, B, V, \Omega, \Sigma_A, \Sigma_B \rangle$ and in the point association $\mathsf{pnt} : W \to D$, where $W \doteq (A \cup B) \setminus V$. Denote $\Sigma_A = \langle D_A, R_A, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$ and $\Sigma_B = \langle D_B, R_B, \Sigma_{B,enc}, \Sigma_{B,dec} \rangle$.*

Assume that there is $0 < \delta'_{min} < 1$, such that for every real δ satisfying $\delta'_{min} \leq \delta < 1$ it holds that $\frac{\delta}{l_{max}(\delta)} \geq 2\sqrt{\epsilon}$.

Then, for every assignment $pe : W \to R$ and every real δ such that $\max \{\delta_{min}, \delta'_{min}\} \leq \delta < 1$, there exist $l' \leq \frac{2}{\delta} \cdot l_{max}(\delta)$ elements $g_1, \ldots, g_{l'} \in \mathcal{D}_{dec}$, for which the following holds.

Assume the algorithm $\mathcal{A}$ is invoked on some input k-tuples. Denote the output by

$$\mathcal{G} = \langle G = (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat, label, proj, tup, eval, pnt \equiv \mathsf{pnt}, evalp \rangle$$

Let $C_B : B \to \Sigma_{B,dec}$ be an assignment, and let $f_1, \ldots, f_l \in \mathcal{D}_{dec}$ be the $l \leq l_{max}(\delta)$ elements guaranteed by the list decoding property of $\mathcal{G}$ for C_B and δ. Let $C_A : A \to \Sigma_{A,dec}$ be an assignment.

When picking uniformly at random an edge $e = (a, b) \in E$, the probability that, in $\mathcal{G}$ under C_A, C_B, (i) the edge e is satisfied, (ii) for the vertex $v \in W \cap \{a, b\}$ it holds $evalp(v, C_W(v)) = pe(v)$, and (iii) e does not read an element from $\{f_1, \ldots, f_l\} \cap \{g_1, \ldots, g_{l'}\}$, is at most $O(\delta)$.

Proof. Define a code $C \subseteq R^{|W|}$ as follows: for every $g \in \mathcal{D}_{dec}$ define a codeword with $|W|$ coordinates by letting the symbol in the position $v \in W$ be $g(\mathsf{pnt}(v))$. Note that by the definition of bipartite tuple-point evaluation graphs, this code is a repetition of the code defined by $\mathcal{D}_{dec}$. Hence, its relative distance is $1 - \epsilon$ as well.

Fix an assignment $pe : W \to R$. Fix a real δ such that $\max \{\delta_{min}, \delta'_{min}\} \leq \delta < 1$. Set $\delta' \doteq \frac{\delta}{l_{max}(\delta)}$, and note that $\delta' \geq 2\sqrt{\epsilon}$. Let $g_1, \ldots, g_{l'} \in \mathcal{D}_{dec}$ be all functions $g \in \mathcal{D}_{dec}$ that agree with the point evaluation pe on at least δ' fraction of the vertices, that is, $|\{v \in W \mid g(\mathsf{pnt}(v)) = pe(v)\}| \geq \delta' \cdot |W|$. Note that there are at most $\frac{2}{\delta'} = \frac{2}{\delta} \cdot l_{max}(\delta)$ such functions by applying Proposition 5.4 on C.

Assume the algorithm $\mathcal{A}$ is invoked on some input k-tuples. Denote the output by $\mathcal{G}$ as above. Let $C_B : B \to \Sigma_{B,dec}$ be an assignment, and let $f_1, \ldots, f_l \in \mathcal{D}_{dec}$ be the $l \leq l_{max}(\delta)$ elements guaranteed by the list decoding property of $\mathcal{G}$ for C_B and δ. Let $C_A : A \to \Sigma_{A,dec}$ be an assignment.

Pick an edge $e = (a, b) \in E$ uniformly at random. Denote the W vertex touching e by $v \in \{a, b\} \cap W$, and note that v is uniformly distributed in W. We will bound the probability that the following *bad* events happen by $O(\delta)$ and be done:

43

- **BAD$_1$**: In $\mathcal{G}$ under the assignments C_A and C_B, the edge e is satisfied but does not read any of $f_1, \ldots, f_l$.

- **BAD$_2$**: In $\mathcal{G}$ under the assignments C_A and C_B, for some $j \in [l]$, the edge e reads f_j, $evalp(v, C_W(v)) = pe(v)$, yet $f_j \notin \{g_1, \ldots, g_{l'}\}$.

The bound on BAD_1 follows directly from Definition 6.7. Let us bound BAD_2: Fix $j \in [l]$. Whenever e reads f_j and $evalp(v, C_W(v)) = pe(v)$, it holds that $f_j(\mathsf{pnt}(v)) = pe(v)$. When $f_j \notin \{g_1, \ldots, g_{l'}\}$, this can happen with probability less than δ'. The probability that this happens for some $j \in [l]$ is at most $l \cdot O(\delta') = O(\delta)$. $\qquad\square$

6.2.9 Generic Framework for Composable Bipartite Locally Decode/Reject Codes

We define a generic framework for handling both composable bipartite locally decode/reject codes and their point variants.

Definition 6.9 (Generic bipartite evaluation graph). *Let $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a domain. $\mathcal{G} = \langle G = (A, B, E), \Omega, \Sigma_A, \Sigma_B, sat, label, proj, read \rangle$ is called a generic bipartite evaluation graph, if $\mathcal{G} = \langle G = (A, B, E), \Omega, \Sigma_A, \Sigma_B, sat, label, proj \rangle$ is a bipartite constraint graph and $read \subseteq E \times \Sigma_A \times \Sigma_B \times \mathcal{D}_{dec}$ is a relation. We say that an edge $e = (a, b) \in E$ reads some $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under assignments $C_A : A \to \Sigma_A$ and $C_B : B \to \Sigma_B$, if $(e, C_A(a), C_B(b), f) \in read$.*

Definition 6.10 (Generic bipartite locally decode/reject code). *Let $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a domain. Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function.*

$\mathcal{G} = \langle G = (A, B, E), \Omega, \Sigma_A, \Sigma_B, sat, label, proj, read \rangle$ is called a (δ_{min}, l_{max})-generic bipartite locally decode/reject code, if:

- *$\Sigma_A = \langle D_A, R_A, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$ and $\Sigma_B = \langle D_B, R_B, \Sigma_{B,enc}, \Sigma_{B,dec} \rangle$ are domains.*

- *$\mathcal{G}' = \langle G = (A, B, E), \Omega, \Sigma_{A,dec}, \Sigma_{B,dec}, sat, label, proj, read \rangle$ is a generic bipartite evaluation graph. We say that an edge $e \in E$ is satisfied in $\mathcal{G}$ under assignments $C_A : A \to \Sigma_{A,dec}$ and $C_B : B \to \Sigma_{B,dec}$, if e is satisfied in $\mathcal{G}'$ under C_A, C_B. We say that an edge $e \in E$ reads some $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under assignments $C_A : A \to \Sigma_{A,dec}$ and $C_B : B \to \Sigma_{B,dec}$, if e reads f in $\mathcal{G}'$ under C_A, C_B.*

- *The following holds:*

  1. ***Encoding:*** *There is an efficient algorithm that given a message $f \in \mathcal{D}_{enc}$, computes assignments $C_A : A \to \Sigma_{A,enc}$ and $C_B : B \to \Sigma_{B,enc}$, such that every edge $e = (a, b) \in E$ is satisfied and reads f in $\mathcal{G}$ under C_A, C_B.*

  2. ***List Decoding:*** *For every assignment $C_B : B \to \Sigma_{B,dec}$, for every real δ such that $\delta_{min} \leq \delta < 1$, there exist $l \leq l_{max}(\delta)$ elements $f_1, \ldots, f_l \in \mathcal{D}_{dec}$, such that for any assignment $C_A : A \to \Sigma_{A,dec}$, the following holds: when picking uniformly at random an edge $e = (a, b) \in E$, the probability that, in $\mathcal{G}$ under C_A, C_B, the edge e is satisfied, although e does not read any of $f_1, \ldots, f_l$, is at most $O(\delta)$.*

44

In this generic framework, we can prove the following useful proposition, stating that not only that every assignment to the B vertices defines a list decoding, but also every assignment to the A vertices defines a list decoding. The list decoding may be larger than the list decoding defined by the assignment to the B vertices and may incur a larger error.

Proposition 6.11 (List decoding for assignment to A vertices). *Let $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec}\rangle$ be a domain. Let $0 < \delta_{min} < 1$. Let $l_{max} : (0,1) \to \mathbb{R}^+$ be a decreasing function. Let $\mathcal{G} = \langle G = (A, B, E), \Omega, \Sigma_A, \Sigma_B, sat, label, proj, read\rangle$ be a (δ_{min}, l_{max})-generic bipartite locally decode/reject code. Then, for every assignment $C_A : A \to \Sigma_{A,dec}$, for every real δ such that $\sqrt{\delta_{min}} \leq \delta < 1$, there exist $l' \leq \frac{1}{\delta} \cdot l_{max}(\delta^2)$ elements $g_1, \ldots, g_{l'} \in \mathcal{D}_{dec}$ satisfying the following. Let $C_B : B \to \Sigma_{B,dec}$ be an assignment. When picking uniformly at random an edge $e = (a, b) \in E$, the probability that, in $\mathcal{G}$ under C_A, C_B, the edge e is satisfied although e does not read an element from $g_1, \ldots, g_{l'}$, is at most $O(\delta)$.*

Proof. Fix an assignment $C_A : A \to \Sigma_{A,dec}$ and a real δ such that $\sqrt{\delta_{min}} \leq \delta < 1$.

Fix $s \doteq \lfloor \frac{1}{\delta} \rfloor$. Let $b \in B$. Let $\sigma_1(b), \ldots, \sigma_s(b) \in \Sigma_{B,dec}$ be all elements $\sigma \in \Sigma_{B,dec}$ that at least δ fraction of the edges coming into b in G "vote" for them according to C_A, i.e.,

$$|\{i \in [\Delta_G(b)] \mid e_G(b,i) = (a,b) \wedge proj(a, C_A(a), label(e_G(b,i))) = \sigma\}| \geq \delta \cdot \Delta_G(b)$$

Note that indeed there are at most s such elements $\sigma \in \Sigma_{B,dec}$ (there might be less than s elements, in which case we pad the list arbitrarily). Define s assignments for B, $C_{B,1}, \ldots, C_{B,s} : B \to \Sigma_{B,dec}$, by letting, for every $j \in [s]$ and $b \in B$, $C_{B,j}(b) \doteq \sigma_j(b)$.

Fix a confidence parameter $\delta^* \doteq \delta^2 \geq \delta_{min}$. For every $j \in [s]$, let $g_{j,1}, \ldots, g_{j,l^*} \in \mathcal{D}_{dec}$ denote the list decoding of $l^* \leq l_{max}(\delta^*)$ elements corresponding to $C_{B,j}$ for confidence parameter δ^*, as follows from the list decoding property of $\mathcal{G}$. Note that the total number of elements we define (possibly with repetitions) is $l' \doteq s \cdot l^* \leq \frac{1}{\delta} \cdot l_{max}(\delta^2)$.

Let $C_B : B \to \Sigma_{B,dec}$ be an assignment. Pick uniformly at random an edge $e = (a, b) \in E$. We will bound the probability that the following *bad* events happen by $O(\delta)$ and be done:

- **BAD$_1$**: The edge e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, however $C_B(b) \notin \{C_{B,1}(b), \ldots, C_{B,s}(b)\}$.

- **BAD$_2$**: The edge e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, and for some $j \in [s]$, it holds that $C_B(b) = C_{B,j}(b)$, however, e does not read one of $g_{j,1}, \ldots, g_{j,l^*}$.

Bounding BAD_1**.** Let $b \in B$ such that $C_B(b) \notin \{C_{B,1}(b), \ldots, C_{B,s}(b)\}$. Then, for less than δ fraction of the $i \in [\Delta_G(b)]$ we have that $e_G(b,i) = (a,b) \in E$ is satisfied in $\mathcal{G}$ under the assignments C_A and C_B: it cannot hold that $proj(a, C_A(a), label(e_G(b,i))) = C_B(b)$ for δ fraction of the $i \in [\Delta_G(b)]$. The bound on BAD_1 follows.

Bounding BAD_2**.** On the event BAD_2, for $j \in [s]$ it holds that e is satisfied in $\mathcal{G}$ under $C_A, C_{B,j}$, yet e does not read any of $g_{j,1}, \ldots, g_{j,l^*}$. By the list decoding property of $\mathcal{G}$, the probability that this happens is at most $s \cdot O(\delta^*) = O(\delta)$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

6.3 Edge Reading Bipartite Locally Decode/Reject Code

In this section we define another instance of generic bipartite locally decode/reject code. In this instance, the edges evaluate tuples.

Definition 6.12 (Edge reading bipartite locally decode/reject code). *Let* $\mathcal{D} = \langle D, R, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ *be a domain. Let* k *and* N *be natural numbers. Assume a collection of* k*-tuples:*

$$\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$$

Let $0 < \delta_{min} < 1$*. Let* $l_{max} : (0,1) \to \mathbb{R}^+$ *be a decreasing function.*

$\mathcal{G} = \langle G = (A, B, E), \Omega, \Sigma_A, \Sigma_B, sat, label, proj, tup, eval \rangle$ *is called a* (δ_{min}, l_{max})*-edge reading bipartite locally decode/reject code, if:*

- $\Sigma_A = \langle D_A, R_A, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$ *and* $\Sigma_B = \langle D_B, R_B, \Sigma_{B,enc}, \Sigma_{B,dec} \rangle$ *are domains.*

- $tup : E \to D^k$ *is a function mapping each edge to a* k*-tuple* $\langle x_{i,1}, \ldots, x_{i,k} \rangle$ *for* $i \in [N]$*. Each* $i \in [N]$ *must have the same (positive) number of edges* $e \in E$ *that are associated with the* i*'th* k*-tuple, i.e.,* $tup(e) = \langle x_{i,1}, \ldots, x_{i,k} \rangle$*.*

- $eval : E \times \Sigma_{A,dec} \to R^k$ *is a function, mapping each edge* $e \in E$ *with an assignment to the edge's* A *endpoint (which determines an assignment to the edge's* B *endpoint) to assignments for the elements of* $tup(e)$*.*

- *For an edge* $e \in E$*, assignments* $\sigma_a \in \Sigma_{A,dec}$ *and* $\sigma_b \in \Sigma_{B,dec}$ *and* $f \in \mathcal{D}_{dec}$*, let* $(e, \sigma_a, \sigma_b, f) \in$ $read$ *if and only if* $eval(e, \sigma_a) = \langle f(x_{i,1}), \ldots, f(x_{i,k}) \rangle$ *where* $tup(e) = \langle x_{i,1}, \ldots, x_{i,k} \rangle$*. Then,* $\mathcal{G}' = \langle G = (A, B, E), \Omega, \Sigma_A, \Sigma_B, sat, label, proj, read \rangle$ *is a* (δ_{min}, l_{max})*-generic bipartite locally decode/reject code.*

- *We say that* e *is satisfied in* $\mathcal{G}$ *under assignments* $C_A : A \to \Sigma_{A,dec}$ *and* $C_B : B \to \Sigma_{B,dec}$*, if* e *is satisfied in* $\mathcal{G}'$ *under* C_A *and* C_B*. We say that* e *reads* f *in* $\mathcal{G}$ *under* C_A *and* C_B*, if* e *reads* f *in* $\mathcal{G}'$ *under* C_A *and* C_B*.*

A $(\mathcal{D}, k, N)$-construction algorithm for edge reading bipartite locally decode/reject codes is a procedure that given as input a collection of size N of k-tuples of points

$$\langle x_{1,1}, \ldots, x_{1,k} \rangle, \ldots, \langle x_{N,1}, \ldots, x_{N,k} \rangle \in D^k$$

outputs an edge reading bipartite locally decode/reject code for those tuples. Efficiency and uniformity in structure of such algorithms are as for construction algorithms for composable bipartite locally decode/reject codes.

7 Building Blocks

We will devise construction algorithms for various types of composable bipartite locally decode/reject codes and point variants of them. These types will differ in the type of domains they work with (Reed-Muller domain, Hadamard domain, *etc*), in whether they are left or right evaluators and in the specific form of their sat, $label$, $proj$, $eval$ and $evalp$ functions. These specifics would later allow us to transform construction algorithms for one type of codes to construction algorithms for others and to compose construction algorithms. In this section we survey the different types of building blocks we use. In the next section we give a full account of the different manipulations on the building blocks.

7.1 Reed-Muller Left Reader

A Reed-Muller Left Reader (RM-LR) is a composable bipartite locally decode/reject code. It works for a Reed-Muller domain $\mathcal{D}$ defined by some finite field $\mathbb{F}$, a dimension m, an encoding degree d and a decoding degree d'. It is a left evaluator (meaning that the set of evaluating vertices is A), and the alphabet domain for the A vertices $\widetilde{\mathcal{D}}$ is also a Reed-Muller domain with the same finite field $\mathbb{F}$, but with different (hopefully reduced) dimension and degree parameters, denoted w, d_w and d'_w, respectively. No additional satisfiability constraints are imposed on the assignments to the A vertices. Assignments to the B vertices are over the domain associated with the field $\mathbb{F}$.

The edges are labeled by points in $\mathbb{F}^w$, i.e., $\Omega \doteq \mathbb{F}^w$. The projection of a vertex $a \in A$, assigned some polynomial, is given by evaluating the polynomial on the point given as label. A vertex $a \in A$ evaluates its tuple $tup(a)$ by evaluating the polynomial assigned to it on k pre-defined points $\vec{p}_1, \ldots, \vec{p}_k \in \mathbb{F}^w$. This way we reduce the problem of evaluating k-tuples in $\mathcal{D}$ to evaluating k-tuples in $\widetilde{\mathcal{D}}$. It is convenient – and does not restrict us – to have the same points $\vec{p}_1, \ldots, \vec{p}_k$ for all vertices $a \in A$.

Formally,

Definition 7.1 (Reed-Muller Left Reader (RM-LR)). *Assume domains as follows:*

- *Let $\mathcal{D} = \langle \mathbb{F}^m, \mathbb{F}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension m, an encoding degree d and a decoding degree d'.*

- *Let $\widetilde{\mathcal{D}} = \langle \mathbb{F}^w, \mathbb{F}, \widetilde{\mathcal{D}}_{enc}, \widetilde{\mathcal{D}}_{dec} \rangle$ be a Reed-Muller domain defined by the field $\mathbb{F}$, a dimension w, an encoding degree d_w and a decoding degree d'_w.*

- *Let $\widetilde{\mathbb{F}} = \langle \{1\}, \mathbb{F}, \widetilde{\mathbb{F}}_{enc}, \widetilde{\mathbb{F}}_{dec} \rangle$ be the domain associated with $\mathbb{F}$. Recall that we associate a domain with a finite set by taking $\widetilde{\mathbb{F}}_{enc} = \widetilde{\mathbb{F}}_{dec} = \{f \mid f : \{1\} \to \mathbb{F}\}$.*

Let k and N be natural numbers. Assume a collection of k-tuples:

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function.

A (δ_{min}, l_{max})-composable bipartite locally decode/reject code for the k-tuples

$$\mathcal{G} = \langle G = (A, B, E), V = A, \Omega, \widetilde{\mathcal{D}}, \widetilde{\mathbb{F}}, sat, label, proj, tup, eval \rangle$$

is called a (δ_{min}, l_{max})-Reed-Muller Left Reader (RM-LR) reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the k-tuples, if the following holds:

1. **Satisfaction:** *For every vertex $a \in A$ and assignment $\sigma_a \in \widetilde{\mathcal{D}}_{dec}$, it holds that $sat(a, \sigma_a) = true$.*

2. **Projection:** *$\Omega \doteq \mathbb{F}^w$, and for every vertex $a \in A$, assignment $\sigma_a \in \widetilde{\mathcal{D}}_{dec}$ and label $\vec{p} \in \mathbb{F}^w$, we have that $proj(a, \sigma_a, \vec{p})$ is the element in $\widetilde{\mathbb{F}}_{dec}$ corresponding to $\sigma_a(\vec{p})$.*

3. **Tuple Evaluation:** *There are points $\vec{p}_1, \dots, \vec{p}_k \in \mathbb{F}^w$, such that for every vertex $a \in A$ and assignment $\sigma_a \in \widetilde{\mathcal{D}}_{dec}$, we have $eval(a, \sigma_a) = \langle \sigma_a(\vec{p}_1), \dots, \sigma_a(\vec{p}_k) \rangle$.*

A $(\mathcal{D}, k, N)$-RM-LR construction algorithm with structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ is an efficient $(\mathcal{D}, k, N)$-construction algorithm that given a collection of size N of k-tuples, outputs an RM-LR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the k-tuples that has size size, block length block, left degree $\mathsf{degleft}$ and right degree $\mathsf{degright}$.

7.2 Reed-Muller Left+Point Reader

A Reed-Muller Left+Point Reader (RM-LPR) is the point variant of a Reed-Muller Left Reader. Every vertex $b \in B$ is associated with a point $pnt(b)$. An assignment σ_b to b corresponds to a field element, which is also the evaluation on the associated point $evalp(b, \sigma_b)$.

Definition 7.2 (Reed-Muller Left+Point Reader (RM-LPR)). *Let $\mathcal{D}$ and $\widetilde{\mathcal{D}}$ be Reed-Muller domains. Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function.*

A (δ_{min}, l_{max})-composable bipartite locally decode/reject code (point variant) $\mathcal{G}$ for some collection of tuples is called a (δ_{min}, l_{max})-Reed-Muller Left+Point Reader (RM-LPR) reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the tuples, if:

1. *The composable bipartite locally decode/reject code induced by $\mathcal{G}$ is a (δ_{min}, l_{max})-Reed-Muller Left Reader reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the tuples.*

2. *Denote the alphabet domain of the B vertices by $\widetilde{\mathbb{F}} = \langle \{1\}, \mathbb{F}, \widetilde{\mathbb{F}}_{enc}, \widetilde{\mathbb{F}}_{dec} \rangle$. For every vertex $b \in B$ and assignment $\sigma_b \in \widetilde{\mathbb{F}}_{dec}$, it should hold that $evalp(b, \sigma_b)$ is the field element corresponding to σ_b.*

A $(\mathcal{D}, k, N)$-RM-LPR construction algorithm with structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ is an efficient $(\mathcal{D}, k, N)$-construction algorithm that given a collection of size N of k-tuples, outputs an RM-LPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the k-tuples that has size size, block length block, left degree $\mathsf{degleft}$ and right degree $\mathsf{degright}$.

7.3 Hadamard Left Reader

A Hadamard Left Reader (Had-LR) is a composable bipartite locally decode/reject code that works for a Hadamard domain and is a left evaluator. Since we will use Hadamard Left Readers only as inner constructions, we make very few restrictions on their structure.

Definition 7.3 (Hadamard Left Reader (Had-LR)). *Let $\mathbb{F}$ be a finite field and let m be a natural number. Let $\mathcal{D} = \langle \mathbb{F}^m, \mathbb{F}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a Hadamard domain. Let k and N be natural numbers. Assume a collection of k-tuples:*

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Let $0 < \delta_{min} < 1$. Let $l_{max} : (0,1) \to \mathbb{R}^+$ be a decreasing function.

A (δ_{min}, l_{max})-composable bipartite locally decode/reject code for the k-tuples

$$\mathcal{G} = \langle G = (A, B, E), V = A, \Omega, \Sigma_A, \Sigma_B, sat \equiv true, label, proj, tup, eval \rangle$$

is called a (δ_{min}, l_{max})-Hadamard Left Reader (Had-LR) for the k-tuples.

A $(\mathcal{D}, k, N)$-Had-LR construction algorithm with structural parameters (size, block, degleft, degright) *is an efficient $(\mathcal{D}, k, N)$-construction algorithm that given a collection of size N of k-tuples, outputs a Had-LR for the k-tuples that has size* size, *block length* block, *left degree* degleft *and right degree* degright.

7.4 RM ⋄ Had Left Reader

An RM ⋄ Had Left Reader (RM⋄Had-LR) is any composable bipartite locally decode/reject code that works for a RM ⋄ Had domain and is a left evaluator. Since we will use RM ⋄ Had Left Readers only as inner constructions, we make very few restrictions on their structure.

Definition 7.4 (RM ⋄ Had Left Reader (RM ⋄ Had-LR)). *Let $\mathbb{F}$ be a finite field, and let $\mathbb{L}$ be a subfield of $\mathbb{F}$, where the extension degree of $\mathbb{F}$ over $\mathbb{L}$ is $\tau = [\mathbb{F} : \mathbb{L}]$. Let m be a natural number. Let $\mathcal{D} = \langle \mathbb{F}^m \times \mathbb{L}^\tau, \mathbb{L}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a RM⋄Had Domain. Let k and N be natural numbers. Assume a collection of k-tuples:*

$$\langle (\vec{x}_{1,1}, \vec{y}_{1,1}), \ldots, (\vec{x}_{1,k}, \vec{y}_{1,k}) \rangle, \ldots, \langle (\vec{x}_{N,1}, \vec{y}_{N,1}), \ldots, (\vec{x}_{N,k}, \vec{y}_{N,k}) \rangle \in (\mathbb{F}^m \times \mathbb{L}^\tau)^k$$

Let $0 < \delta_{min} < 1$. Let $l_{max} : (0,1) \to \mathbb{R}^+$ be a decreasing function.

A (δ_{min}, l_{max})-composable bipartite locally decode/reject code for the k-tuples

$$\mathcal{G} = \langle G = (A, B, E), V = A, \Omega, \Sigma_A, \Sigma_B, sat \equiv true, label, proj, tup, eval \rangle$$

is called a (δ_{min}, l_{max})-RM⋄Had Left Reader (RM ⋄ Had-LR) for the k-tuples.

A $(\mathcal{D}, k, N)$-RM ⋄ Had-LR construction algorithm with structural parameters (size, block, degleft, degright) *is an efficient $(\mathcal{D}, k, N)$-construction algorithm that given a collection of size N of k-tuples, outputs an RM ⋄ Had-LR for the k-tuples that has size* size, *block length* block, *left degree* degleft *and right degree* degright.

7.5 Reed-Muller Right Reader

A Reed-Muller Right Reader (RM-RR) is a composable bipartite locally decode/reject code. It works for a Reed-Muller domain defined by some finite field $\mathbb{F}$, dimension m, encoding degree d and decoding degree d'. It is a right evaluator (meaning that the set of evaluating vertices is B), and the alphabet domain $\widetilde{\mathcal{D}}$ for the B vertices is also a Reed-Muller domain with the same finite field $\mathbb{F}$, but with different (hopefully reduced) dimension and degree parameters, denoted w, d_w and d'_w, respectively. A vertex $b \in B$ evaluates its tuple $tup(b)$ by evaluating the polynomial assigned to it on k pre-defined points $\vec{p}_1, \ldots, \vec{p}_k \in \mathbb{F}^w$. Assignments to the A vertices contain assignments to the neighboring B vertices, by specifying a polynomial per label $\xi \in \Omega$. Satisfiability constraints on the a vertices compare the evaluations of the polynomials on different points. The constraints are in a tree structure as explained next.

Tree satisfiability constraints. Tree satisfiability constraints for a vertex $a \in A$ are given by a (rooted) tree $T_a = (U_a \cup \Omega, E_a)$ and functions $\{P_{a,\xi}\}_{\xi \in \Omega}$, called *ancestors point specification functions*:

1. The leaves of the tree T_a are the elements in Ω. The set of inner nodes in the tree is U_a.

2. For every depth in the tree, all the nodes in this depth have the same number of children. In particular, all the leaves have the same depth in T_a, denoted $depth(T_a)$.

3. Every leaf $\xi \in \Omega$ specifies a point in $\mathbb{F}^w$ for each of its ancestors. The specification is given by the function $P_{a,\xi} : \{0, \ldots, depth(T_a) - 1\} \to \mathbb{F}^w$, where the ancestors are represented by their depth in the tree.

A polynomial $Q_\xi \in \widetilde{\mathcal{D}}_{dec}$ assigned to a leaf $\xi \in \Omega$ defines an assignment of field elements to the ancestors of ξ in the tree by evaluating the polynomial on the points associated with them $Q_\xi(P_{a,\xi}(i))$ for $i = 0, \ldots, depth(T_a) - 1$. The tree satisfiability constraints are said to be *satisfied* by an assignment $\sigma_a : \Omega \to \widetilde{\mathcal{D}}_{dec}$ of polynomials to the leaves, if there is an assignment of field elements to all the inner nodes of the tree $\sigma : U_a \to \mathbb{F}$ that is consistent with the evaluations of all the leaves. I.e., if $u \in U_a$ is in depth $i \in \{0, \ldots, depth(T_a) - 1\}$ in the tree, the leaf $\xi \in \Omega$ is a descendent of it and $Q_\xi = \sigma_a(\xi)$ is the polynomial assigned to ξ, then it should hold that $\sigma(u) = Q_\xi(P_{a,\xi}(i))$. Intuitively, each leaf $\xi \in \Omega$ "has an opinion" on all the vertices on the path from it to the root. The satisfiability constraints are satisfied if all the leaves agree.

In addition, we require an RM-RR to be left regular (note that since it is a right evaluator, it is necessarily right regular), and require that for every vertex $a \in A$, there would be the same number of edges coming out of a for each label.

Definition 7.5 (Reed-Muller Right Reader (RM-RR)). *Assume domains as follows:*

- *Let $\mathcal{D} = \langle \mathbb{F}^m, \mathbb{F}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension m, an encoding degree d and a decoding degree d'.*

- *Let $\widetilde{\mathcal{D}} = \langle \mathbb{F}^w, \mathbb{F}, \widetilde{\mathcal{D}}_{enc}, \widetilde{\mathcal{D}}_{dec} \rangle$ be a Reed-Muller domain defined by the field $\mathbb{F}$, a dimension w, an encoding degree d_w and a decoding degree d'_w.*

- *Let $\Sigma_A = \langle \Omega, \widetilde{\mathcal{D}}_{dec}, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$, where $\Sigma_{A,enc} = \left\{ f \mid f : \Omega \to \widetilde{\mathcal{D}}_{enc} \right\}$ and $\Sigma_{A,dec} = \left\{ f \mid f : \Omega \to \widetilde{\mathcal{D}}_{dec} \right\}$.*

Let k and N be natural numbers. Assume a collection of k-tuples:

$$\langle \vec{x}_{1,1}, \dots, \vec{x}_{1,k} \rangle, \dots, \langle \vec{x}_{N,1}, \dots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Let $0 < \delta_{min} < 1$. Let $l_{max} : (0,1) \to \mathbb{R}^+$ be a decreasing function.

A (δ_{min}, l_{max})-composable bipartite locally decode/reject code for the k-tuples

$$\mathcal{G} = \langle G = (A, B, E), V = B, \Omega, \Sigma_A, \widetilde{\mathcal{D}}, sat, label, proj, tup, eval \rangle$$

is called a (δ_{min}, l_{max})-Reed-Muller Right Reader (RM-RR) reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the k-tuples, if the following holds:

1. ***Satisfaction:*** *For every vertex $a \in A$ there are tree satisfiability constraints given by a tree T_a and ancestors point specification functions $\{P_{a,\xi}\}_{\xi \in \Omega}$, such that for every assignment $\sigma_a : \Omega \to \widetilde{\mathcal{D}}_{dec}$, it holds that $sat(a, \sigma_a) = true$ if and only if the tree satisfiability constraints are satisfied by σ_a.*

2. ***Labeling:*** *Let $a \in A$ be a vertex. For all labels $\xi \in \Omega$, there is the same number of edges $e \in E$ coming out of a with $label(e) = \xi$.*

3. ***Projection:*** *For every vertex $a \in A$, assignment $\sigma_a : \Omega \to \widetilde{\mathcal{D}}_{dec}$ and label $\xi \in \Omega$, we have that $proj(a, \sigma_a, \xi) = \sigma_a(\xi)$.*

4. ***Tuple Evaluation:*** *There are points $\vec{p}_1, \dots, \vec{p}_k \in \mathbb{F}^w$, such that for every vertex $b \in B$ and assignment $\sigma_b \in \widetilde{\mathcal{D}}_{dec}$, we have $eval(b, \sigma_b) = \langle \sigma_b(\vec{p}_1), \dots, \sigma_b(\vec{p}_k) \rangle$.*

5. ***Regularity:*** *$\mathcal{G}$ is left regular.*

A $(\mathcal{D}, k, N)$-RM-RR construction algorithm with structural parameters (size, block, degleft, degright, depth) reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ is an efficient $(\mathcal{D}, k, N)$-construction algorithm that given a collection of size N of k-tuples, outputs an RM-RR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the k-tuples that has size size, block length block, left degree degleft, right degree degright, and whose A vertices all have tree satisfiability constraints of depth depth.

7.6 Reed-Muller Right+Point Reader

A Reed-Muller Right+Point Reader (RM-RPR) is the point variant of a Reed-Muller Right Reader. Each vertex $a \in A$ is associated with a point $pnt(a)$. We think of this point as associated with the root of the satisfiability tree of a, namely, the point that all the leaves $\xi \in \Omega$ "have an opinion on". Given an assignment σ_a to a, we let the point evaluation function $evalp(a, \sigma_a)$ be the opinion of an arbitrary leaf $\xi_0 \in \Omega$. Recall that if σ_a is satisfying, i.e., $sat(a, \sigma_a) = true$, then all the leaves agree on the assignment to the root.

Definition 7.6 (Reed-Muller Right+Point Reader (RM-RPR)). *Let $\mathcal{D}$ and $\widetilde{\mathcal{D}}$ be Reed-Muller domains. Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \rightarrow \mathbb{R}^+$ be a decreasing function.*

A (δ_{min}, l_{max})-composable bipartite locally decode/reject code (point variant) $\mathcal{G}$ for some collection of tuples is called a (δ_{min}, l_{max})-Reed-Muller Right+Point Reader (RM-RPR) reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the tuples, if:

1. *The composable bipartite locally decode/reject code induced by $\mathcal{G}$ is a (δ_{min}, l_{max})-Reed-Muller Right Reader reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the tuples.*

2. *Denote the alphabet domain of the A vertices by $\Sigma_A = \langle \Omega, \widetilde{\mathcal{D}}_{dec}, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$. Fix some arbitrary $\xi_0 \in \Omega$. For every vertex $a \in A$, for every assignment $\sigma_a : \Omega \rightarrow \widetilde{\mathcal{D}}_{dec}$, we have that $evalp(a, \sigma_a) = \sigma_a(\xi_0)(P_{a,\xi_0}(0))$ (where P_{a,ξ_0} is the ancestors point specification function associated with a's satisfiability tree).*

A $(\mathcal{D}, k, N)$-RM-RPR construction algorithm with structural parameters
$(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright}, \mathsf{depth})$ *reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ is an efficient $(\mathcal{D}, k, N)$-construction algorithm that given a collection of size N of k-tuples, outputs an RM-RPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the k-tuples that has size size, block length block, left degree $\mathsf{degleft}$, right degree $\mathsf{degright}$, and whose A vertices all have tree satisfiability constraints of depth depth.*

8 Manipulations on Building Blocks

In this section we survey the different manipulations we have on building blocks: generation, change of domains, right degree reduction, transformation of left readers into right readers and composition.

8.1 Generation of Left Readers

We will be able to devise construction algorithms for left readers.

8.1.1 Construction of Reed-Muller Left and Left+Point Readers

The first algorithm we show is an RM-LPR construction algorithm that is uniform in the point association. This algorithm works for Reed-Muller domains $\mathcal{D}$ in which the field is sufficiently

large with respect to the dimension m, the decoding degree d' and the number k of points in a tuple we wish to evaluate. It reduces the evaluation in $\mathcal{D}$ to evaluation in a Reed-Muller domain $\widetilde{\mathcal{D}}$ with a *constant* dimension. On the downside, $\widetilde{\mathcal{D}}$ has an encoding degree that – not only is not smaller than the encoding degree d of $\mathcal{D}$ – but is slightly larger. In addition, the size, block, degleft and degright parameters of the algorithm are all very large. Since we will use this algorithm as an inner construction, we can put up with the large size and block length. As to the left and right degrees – subsequent manipulations would allow us to reduce them.

Lemma 8.1 (Construction of RM-LPR). *Set $w \doteq 4$. Let $\mathcal{D}$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension $m > w$, an encoding degree d and a decoding degree d'. Let $k < |\mathbb{F}|$ and N be natural numbers. We assume that the following condition holds:*

- $d' \geq (k+1) \cdot d$.

Let $\widetilde{\mathcal{D}}$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension w, an encoding degree $(k+1) \cdot d$ and a decoding degree d'.

Then, there is a $(\mathcal{D}, k, N)$-RM-LPR construction algorithm with structural parameters (size, block, degleft, degright) *reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for* size $\leq N \cdot |\mathbb{F}|^{O(m)}$, block $\leq \mathsf{poly}(k, d) \cdot \log |\mathbb{F}|$, degleft $\leq |\mathbb{F}|^{O(1)}$ *and* degright $\leq N \cdot |\mathbb{F}|^{O(m)}$. *The algorithm is uniform in the point association and outputs (δ_{min}, l_{max})-RM-LPRs for*

$$\delta_{min} \doteq \max \left\{ \sqrt{\frac{d'(k+1)}{|\mathbb{F}| - k}}, m \left(\sqrt[8]{\frac{1}{|\mathbb{F}|}} + \sqrt[4]{\frac{md'}{|\mathbb{F}|}} \right) \right\}$$

and $l_{max}(\delta) \doteq \frac{2}{\delta}$.

We will also show a construction algorithm for (the weaker) RM-LR with much smaller size parameter. With the right choice of parameters, this size would be almost-linear $(N + |\mathbb{F}^m|)^{1+o(1)}$, rather than polynomial in $|\mathbb{F}^m|$ and N. In particular, to save in the size we need the field $\mathbb{F}$ to have a subfield $\mathbb{K}$ of an appropriate size. The smaller the subfield – the smaller the size. The larger the subfield – the lower δ_{min}. Specifically, for δ_{min} to be small we require $|\mathbb{K}| \geq \Omega(m^8)$, while the influence of $|\mathbb{K}|$ on the size is a factor of $|\mathbb{K}|^{O(m)}$. When the dimension m is sufficiently small, we can get low δ_{min} at a reasonable increase in the size.

The other structural parameters block, degleft and degright would remain large, and subsequent manipulations are required for reducing them.

Lemma 8.2 (Construction of RM-LR). *Set $w \doteq 4$. Let $\mathcal{D}$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension $m > w$, an encoding degree d and a decoding degree d'. Let $\mathbb{K} \subseteq \mathbb{F}$ be a subfield of $\mathbb{F}$. Let $k < |\mathbb{F}|$ and N be natural numbers. We assume that the following condition holds:*

- $d' \geq (k+1) \cdot d$.

Let $\widetilde{\mathcal{D}}$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension w, an encoding degree $(k+1) \cdot d$ and a decoding degree d'.

Then, there is a $(\mathcal{D}, k, N)$-RM-LR construction algorithm with structural parameters (size, block, degleft, degright) *reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for* size $\leq (N + |\mathbb{F}|^m) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m}$, block $\leq$ poly$(k, d) \cdot \log |\mathbb{F}|$, degleft $\leq |\mathbb{F}|^{O(1)}$ *and* degright $\leq (N + |\mathbb{F}|^m) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m}$. *The algorithm outputs* (δ_{min}, l_{max})*-RM-LRs for*

$$\delta_{min} \doteq \max \left\{ \sqrt{\frac{d'(k+1)}{|\mathbb{F}| - k}}, m \left(\sqrt[8]{\frac{1}{|\mathbb{K}|}} + \sqrt[4]{\frac{md'}{|\mathbb{F}|}} \right) \right\}$$

and $l_{max}(\delta) \doteq \frac{2}{\delta}$.

8.1.2 Construction of Hadamard Left Reader

We show a Had-LR construction algorithm that is uniform in the tuple association and in the encoding and list decoding. The size, block length, left and right degrees are all large, but as we use this algorithm only as an inner construction, their influence on the overall construction is minor.

We require that the field underlying the Hadamard domain is prime. In the overall construction, this field will be a small subfield of the field we are using for the outer construction.

Lemma 8.3 (Construction of Had-LR). *Let $\mathcal{D}$ be a Hadamard domain defined by a prime finite field $\mathbb{F}$ and a dimension m. Let $k \leq m - 2$ and N be natural numbers.*

Then, there is a $(\mathcal{D}, k, N)$-Had-LR construction algorithm with structural parameters (size, block, degleft, degright) *for* size $\leq N \cdot |\mathbb{F}|^{O(m)}$, block $\leq O(k) \cdot \log |\mathbb{F}|$, degleft $\leq |\mathbb{F}|^{O(k)}$ *and* degright $\leq N \cdot |\mathbb{F}|^{O(m)}$. *The algorithm is uniform in the tuple association and in the encoding and list decoding. It outputs* (δ_{min}, l_{max})*-Had-LRs for* $\delta_{min} \doteq 2\sqrt[6]{\frac{1}{|\mathbb{F}|}}$ *and* $l_{max}(\delta) \doteq \frac{2}{\delta^3}$. *Moreover, the right degrees of the vertices in the Had-LR do not depend on the input to the algorithm.*

8.2 Power Reduction

Suppose that we have construction algorithms for RM-LRs or RM-LPRs reducing $\mathcal{D} \mapsto \mathcal{D}_1$, where the dimension of the RM domain $\mathcal{D}_1$ is small, but the encoding degree is large (Indeed we have such algorithms by Lemmata 8.1 and 8.2). Then, we can transform these algorithms into construction algorithms reducing $\mathcal{D} \mapsto \mathcal{D}_2$, where $\mathcal{D}_2$ is a domain in which both the dimension and the encoding degree are relatively small. Specifically, the dimension and encoding degree parameters are logarithmic in the encoding degree of $\mathcal{D}_1$. Note that this means that the block length becomes larger (although not by much if the dimension of $\mathcal{D}_1$ is constant).

The only pre-requirement for the transformation is that there is a large enough gap to begin with between the encoding degree of $\mathcal{D}_1$ and its decoding degree.

Lemma 8.4 (Power reduction). *Assume the following:*

- *Let $\mathcal{D} = \langle \mathbb{F}^{m_0}, \mathbb{F}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension m_0, an encoding degree d_0 and a decoding degree d_0'.*

- *Let $\mathcal{D}_1 = \langle \mathbb{F}^{m_1}, \mathbb{F}, \mathcal{D}_{1,enc}, \mathcal{D}_{1,dec} \rangle$ be a Reed-Muller domain defined by the field $\mathbb{F}$, a dimension m_1, an encoding degree d_1 and a decoding degree d_1'.*

  Fix $b_1 \doteq \lceil \log(d_1 + 1) \rceil$, and assume that $d_1' \geq m_1 b_1 d_1$. Let $m_2 = d_2 \doteq m_1 \cdot b_1$. Let $d_2' \doteq \lfloor d_1'/d_1 \rfloor$.

- *Let $\mathcal{D}_2 = \langle \mathbb{F}^{m_2}, \mathbb{F}, \mathcal{D}_{2,enc}, \mathcal{D}_{2,dec} \rangle$ be the Reed-Muller domain defined by the field $\mathbb{F}$, dimension m_2, encoding degree d_2 and decoding degree d_2'.*

Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function. Then,

1. *If there is a $(\mathcal{D}, k, N)$-RM-LR construction algorithm with structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$, then there is also a $(\mathcal{D}, k, N)$-RM-LR construction algorithm with structural parameters $(\mathsf{size}, \mathsf{block}', \mathsf{degleft}, \mathsf{degright})$ reducing $\mathcal{D} \mapsto \mathcal{D}_2$, where $\mathsf{block}' = d_1^{O(m_1)} \cdot \log |\mathbb{F}|$. If the former algorithm outputs (δ_{min}, l_{max})-RM-LRs, then so does the latter algorithm.*

2. *If there is a $(\mathcal{D}, k, N)$-RM-LPR construction algorithm with structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$ that is uniform in the point association, then there is also a $(\mathcal{D}, k, N)$-RM-LPR construction algorithm with structural parameters $(\mathsf{size}, \mathsf{block}', \mathsf{degleft}, \mathsf{degright})$ reducing $\mathcal{D} \mapsto \mathcal{D}_2$ that is uniform in the point association, where $\mathsf{block}' = d_1^{O(m_1)} \cdot \log |\mathbb{F}|$. If the former algorithm outputs (δ_{min}, l_{max})-RM-LPRs, then so does the latter algorithm.*

8.3 Right Degree Reduction

We can transform construction algorithms that produce readers with large right degree into construction algorithms that produce right regular readers with small right degree. This comes at the cost of enlarging the size of the construction and the left degree. Yet, the increase in the size and the left degree is proportional to the new right degree which is small. Right degree reduction also causes some deterioration in the error and list size parameters of the readers.

Right degree reduction is possible due to the projection property of the readers. We do not have a similar lemma for left degree reduction. Instead we will use a transformation presented in the next subsection that swaps between the left and right degrees.

Lemma 8.5 (Right degree reduction). *There is a constant $\alpha < 1$ and a function $T : \mathbb{N} \to \mathbb{N}^+$ with $T(\Delta) = \tilde{\Theta}(\Delta)$ as in Lemma 5.3, such that the following holds for every natural number Δ: Let $\mathcal{D}$ and $\widetilde{\mathcal{D}}$ be Reed-Muller domains. Let $\mathcal{D}^\diamond$ be a RM$\diamond$Had domain. Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function. Set $\delta_{min}^* \doteq \max\left\{\sqrt{\delta_{min}}, \frac{1}{(T(\Delta))^{1-\alpha}}\right\}$ and $l_{max}^*(\delta) \doteq \frac{1}{\delta} \cdot l_{max}(\delta^2)$.*

1. *If there is a $(\mathcal{D}, k, N)$-RM-LR construction algorithm with structural parameters* $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ *reducing* $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$, *then there is also a* $(\mathcal{D}, k, N)$-RM-LR *construction algorithm with structural parameters* $(O(\Delta \cdot \mathsf{size}), \mathsf{block}, T(\Delta) \cdot \mathsf{degleft}, T(\Delta))$ *reducing* $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$. *If the former algorithm outputs* (δ_{min}, l_{max})-RM-LRs, *then the latter algorithm outputs* $(\delta^*_{min}, l^*_{max})$-RM-LRs. *Moreover, the output of the algorithm is always right regular.*

2. *If there is a* $(\mathcal{D}^\diamond, k, N)$-RM$\diamond$Had-LR *construction algorithm with structural parameters* $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ *that is uniform in the tuple association and in the encoding and list decoding and outputs RM$\diamond$Had-LRs, in which the right degrees of the vertices do not depend on the input to the algorithm, then there is also a* $(\mathcal{D}^\diamond, k, N)$-RM$\diamond$Had-LR *construction algorithm with structural parameters* $(O(\Delta \cdot \mathsf{size}), \mathsf{block}, T(\Delta) \cdot \mathsf{degleft}, T(\Delta))$ *that is uniform in the tuple association and in the encoding and list decoding. If the former algorithm outputs* (δ_{min}, l_{max})-RM$\diamond$Had-LRs, *then the latter algorithm outputs* $(\delta^*_{min}, l^*_{max})$-RM$\diamond$Had-LRs. *Moreover, the output of the algorithm is always right regular.*

3. *If there is a* $(\mathcal{D}, k, N)$-RM-LPR *construction algorithm with structural parameters* $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ *reducing* $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ *that is uniform in the point association, then there is also a* $(\mathcal{D}, k, N)$-RM-LPR *construction algorithm with structural parameters* $(O(\Delta \cdot \mathsf{size}), \mathsf{block}, T(\Delta) \cdot \mathsf{degleft}, T(\Delta))$ *reducing* $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ *that is uniform in the point association. If the former algorithm outputs* (δ_{min}, l_{max})-RM-LPRs, *then the latter algorithm outputs* $(\delta^*_{min}, l^*_{max})$-RM-LPRs.

4. *If there is a* $(\mathcal{D}, k, N)$-RM-RR *construction algorithm with structural parameters* $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright}, \mathsf{depth})$ *reducing* $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$, *then there is also a* $(\mathcal{D}, k, N)$-RM-RR *construction algorithm with structural parameters* $(O(\Delta \cdot \mathsf{size}), \mathsf{block}, T(\Delta) \cdot \mathsf{degleft}, T(\Delta), \mathsf{depth})$ *reducing* $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$. *If the former algorithm outputs* (δ_{min}, l_{max})-RM-RRs, *then the latter algorithm outputs* $(\delta^*_{min}, l^*_{max})$-RM-RRs.

5. *If there is a* $(\mathcal{D}, k, N)$-RM-RPR *construction algorithm with structural parameters* $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright}, \mathsf{depth})$ *reducing* $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ *that is uniform in the point association, then there is also a* $(\mathcal{D}, k, N)$-RM-RPR *construction algorithm with structural parameters* $(O(\Delta \cdot \mathsf{size}), \mathsf{block}, T(\Delta) \cdot \mathsf{degleft}, T(\Delta), \mathsf{depth})$ *reducing* $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ *that is uniform in the point association. If the former algorithm outputs* (δ_{min}, l_{max})-RM-RPRs, *then the latter algorithm outputs* $(\delta^*_{min}, l^*_{max})$-RM-RPRs.

8.4 Transforming Reed-Muller Left Readers Into Reed-Muller Right Readers

Construction algorithms that produce RM-LRs that are right regular with small right degree can be transformed into construction algorithms for RM-RRs. Construction algorithms that produce RM-LPRs that have small right degree (they are right regular by definition) can be transformed into

construction algorithms for RM-RPRs. Moreover, the latter transformation preserves uniformity in point association. The cost is enlarging the block length by a factor equal to the right degree. If the right degree is small, then this cost is small as well. Additional costs are enlarging the error and list size parameters.

The transformation swaps the left and right degrees. If the original left reader has left degree degleft and right degree degright, then the new right reader has left degree degright and right degree degleft. In particular, if the original left reader has small right degree, then the new right reader has small left degree.

The transformation sets the depth parameter of the right reader construction algorithms to 1.

Lemma 8.6 (Switching sides). *Let $\mathcal{D}$ and $\widetilde{\mathcal{D}}$ be Reed-Muller domains. Let $0 < \delta_{min} < 1$. Let $l_{max} : (0,1) \to \mathbb{R}^+$ be a decreasing function. Set $\delta_{min}^* \doteq \sqrt{\delta_{min}}$ and $l_{max}^*(\delta) \doteq \frac{1}{\delta} \cdot l_{max}(\delta^2)$.*

1. *If there is a $(\mathcal{D}, k, N)$-RM-LR construction algorithm with structural parameters* $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ *reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ that outputs right regular RM-LRs, then there is also a $(\mathcal{D}, k, N)$-RM-RR construction algorithm with structural parameters* $(\mathsf{size}, \mathsf{degright} \cdot \mathsf{block}, \mathsf{degright}, \mathsf{degleft}, 1)$ *reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$. If the former algorithm outputs (δ_{min}, l_{max})-RM-LRs, then the latter algorithm outputs $(\delta_{min}^*, l_{max}^*)$-RM-RRs.*

2. *If there is a $(\mathcal{D}, k, N)$-RM-LPR construction algorithm with structural parameters* $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ *reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ that is uniform in the point association, then there is also a $(\mathcal{D}, k, N)$-RM-RPR construction algorithm with structural parameters* $(\mathsf{size}, \mathsf{degright} \cdot \mathsf{block}, \mathsf{degright}, \mathsf{degleft}, 1)$ *reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ that is uniform in the point association. If the former algorithm outputs (δ_{min}, l_{max})-RM-LPRs, then the latter algorithm outputs $(\delta_{min}^*, l_{max}^*)$-RM-RPRs.*

8.5 Transforming Hadamard Left Readers Into RM⋄Had Left Readers

Construction algorithms for Had-LRs can be transformed into construction algorithms for RM⋄Had-LRs. The cost is a huge blow-up in the parameters of the Hadamard domain compared to the parameters of the RM⋄Had domain. Hence, this transformation is useful only when the parameters of the RM⋄Had domain are very small to begin with.

Lemma 8.7 (From Had-LRs to RM⋄Had-LRs). *Let $\mathcal{D}$ be a RM⋄Had domain defined by a finite field $\mathbb{F}$, a prime subfield $\mathbb{L}$ of $\mathbb{F}$, a dimension m, an encoding degree d and any decoding degree d'. Let $\tau = [\mathbb{F} : \mathbb{L}]$. Set $M \doteq \binom{m+d}{m}$ (the number of monomials in an m-variate polynomial of degree at most d). Let $\mathcal{H}$ be a Hadamard domain defined by the finite field $\mathbb{L}$ and the dimension $M \cdot \tau$.*

Let k and N be natural numbers.

If there is a $(\mathcal{H}, k, N)$-Had-LR construction algorithm with structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ *that is uniform in the tuple association and in the encoding and list decoding and outputs Had-LRs, in which the right degrees of the vertices do no depend on the input to the algorithm, then there is a $(\mathcal{D}, k, N)$-RM⋄Had-LR construction algorithm with the same structural parameters* $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ *that is uniform in the tuple association and in*

the encoding and list decoding and outputs RM⋄Had-LRs, in which the right degrees of the vertices do no depend on the input to the algorithm.

If the former algorithm outputs (δ_{min}, l_{max})-Had-LRs, then the latter algorithm outputs (δ_{min}, l_{max})-RM⋄Had-LRs.

8.6 Composition of Reed-Muller Right Reader and Reed-Muller Right+Point Reader Construction Algorithms

For Reed-Muller domains $\mathcal{D}$, $\mathcal{D}_1$ and $\mathcal{D}_2$, we can compose an RM-RR construction algorithm reducing $\mathcal{D} \mapsto \mathcal{D}_1$ ("outer algorithm") and an RM-RPR construction algorithm reducing $\mathcal{D}_1 \mapsto \mathcal{D}_2$ that is uniform in the point association ("inner algorithm") into an RM-RR construction algorithm reducing $\mathcal{D} \mapsto \mathcal{D}_2$ ("composed algorithm").

Through composition we can reduce the block length. Assume that the outer algorithm produces RM-RRs that have left degree $\mathsf{degleft_{out}}$. Assume that the inner algorithm produces RM-RPRs that have block length $\mathsf{block_{in}}$. The block length of the RM-RRs produced by the composed algorithm is $\mathsf{degleft_{out}} \cdot \mathsf{block_{in}}$ (independent of the block length of the outer algorithm). Assume that the left degree $\mathsf{degleft_{out}}$ is small (this can be taken care of by the previous manipulations). Since the inner construction algorithm should work only for the domain $\mathcal{D}_1$, and not the domain $\mathcal{D}$, the block length $\mathsf{block_{in}}$ can be made small, thus making the block length of the composed construction small.

For composition to be possible, the right degree and the depth parameters of the outer algorithm, denoted $\mathsf{degright_{out}}$ and $\mathsf{depth_{out}}$ respectively, should be small. This is because the inner algorithm should be able to handle tuples with $k + \mathsf{degright_{out}} \cdot \mathsf{depth_{out}}$ points.

The costs of composition are an increase in the size, in the left and right degrees and in the depth parameter, as well as a deterioration of the error and list size parameters.

The size parameter of the composed algorithm is typically dominated by the size parameter of the outer algorithm. If the outer algorithm outputs RM-RRs of size $\mathsf{size_{out}}$, and the inner construction algorithm outputs RM-RPRs of size $\mathsf{size_{in}}$, then the composed algorithm outputs RM-RRs of size roughly $\mathsf{size_{out}} \cdot \mathsf{size_{in}}$. Even if the size $\mathsf{size_{in}}$ is relatively large with respect to the domain parameters, since the domain is only $\mathcal{D}_1$, and not $\mathcal{D}$, the contribution of $\mathsf{size_{in}}$ is typically minor.

Lemma 8.8 (Composition of RM-RR and RM-RPR construction algorithms). *Let $\mathcal{D}$, $\mathcal{D}_1$ and $\mathcal{D}_2$ be Reed-Muller domains with a finite field $\mathbb{F}$. Denote the decoding degree of $\mathcal{D}_1$ by d'_1.*

Let k and N be natural numbers. Let $0 < \delta_{min,out}, \delta_{min,in} < 1$. Let $l_{max,out}, l_{max,in} : (0, 1) \to \mathbb{R}^+$ be decreasing functions. Assume that for some constants $b_1, b_2 \geq 1$ for every $0 < \delta < 1$ it holds that $l_{max,in}(\delta) \leq \frac{b_1}{\delta^{b_2}}$. Set

$$\delta_{min} \doteq \max \left\{ \delta_{min,in}, (2b_1^2 \cdot \delta_{min,out})^{1/(2b_2+3)}, \left(2b_1 \cdot \sqrt{\frac{d'_1}{|\mathbb{F}|}} \right)^{1/(b_2+1)} \right\}$$

and

$$l_{max}(\delta) \doteq \frac{b_1}{\delta^{b_2}} \cdot l_{max,out}\left(\frac{1}{2b_1^2} \cdot \delta^{2b_2+3}\right)$$

Assume that $\delta_{min} < 1$. Given:

- *A $(\mathcal{D}, k, N)$-RM-RR construction algorithm $\mathcal{A}_{out}$ with structural parameters $(\mathsf{size}_{out}, \mathsf{block}_{out}, \mathsf{degleft}_{out}, \mathsf{degright}_{out}, \mathsf{depth}_{out})$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$, with $\mathsf{depth}_{out} \leq d_1'$.*

- *A $(\mathcal{D}_1, k + \mathsf{degright}_{out} \cdot \mathsf{depth}_{out}, 1)$-RM-RPR construction algorithm $\mathcal{A}_{in}$ with structural parameters $(\mathsf{size}_{in}, \mathsf{block}_{in}, \mathsf{degleft}_{in}, \mathsf{degright}_{in}, 1)$ reducing $\mathcal{D}_1 \mapsto \mathcal{D}_2$ that is uniform in the point association.*

One can obtain a $(\mathcal{D}, k, N)$-RM-RR construction algorithm $\mathcal{A}$ with structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright}, \mathsf{depth})$ reducing $\mathcal{D} \mapsto \mathcal{D}_2$ for $\mathsf{size} \leq \mathsf{size}_{out} \cdot \mathsf{size}_{in}$, $\mathsf{block} = \mathsf{degleft}_{out} \cdot \mathsf{block}_{in}$, $\mathsf{degleft} = \mathsf{degleft}_{out} \cdot \mathsf{degleft}_{in}$, $\mathsf{degright} = \mathsf{degright}_{out} \cdot \mathsf{degright}_{in}$ and $\mathsf{depth} = \mathsf{depth}_{out} + 1$.

If $\mathcal{A}_{out}$ outputs $(\delta_{min,out}, l_{max,out})$-RM-RRs, and $\mathcal{A}_{in}$ outputs $(\delta_{min,in}, l_{max,in})$-RM-RPRs, then $\mathcal{A}$ outputs (δ_{min}, l_{max})-RM-RRs.

8.7 Composition of Reed-Muller Right Reader and RM⋄Had Left Reader Construction Algorithms

Let $\mathcal{D}^\diamond$ be a RM⋄Had domain. We wish to obtain an edge reading bipartite locally decode/reject code for $\mathcal{D}^\diamond$ with small block length.

Let $\mathcal{D}$ be a Reed-Muller domain corresponding to the outer code of $\mathcal{D}^\diamond$ and suppose that we have a RM-RR construction algorithm reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for some Reed-Muller domain $\mathcal{D}_1$. From this algorithm we can rather easily obtain an edge reading bipartite locally decode/reject code for $\mathcal{D}^\diamond$ whose block length depends on $\log |\mathbb{F}|$. Our aim, however, is to achieve a much smaller block length. In particular, block length that does not depend on the field size, which is inevitably large in a Reed-Muller code.

We show how to compose the RM-RR construction algorithm reducing $\mathcal{D} \mapsto \mathcal{D}_1$ ("outer algorithm") with a RM⋄Had-LR construction algorithm for the domain $\mathcal{D}_1^\diamond$ corresponding to concatenation of $\mathcal{D}_1$ and Hadamard ("inner algorithm"). For the composition to work the inner algorithm must be uniform in the tuple association and in the encoding and list decoding. The composition results in an algorithm whose block length parameter depends only on the left degree of the outer algorithm and the block length of the inner algorithm.

The costs of this composition, namely the increase in the size and graph degree parameters, are similar to the costs of the composition described in Subsection 8.6.

Lemma 8.9 (Composition of RM-RR and RM⋄Had-LR construction algorithms). *Let $\mathcal{D}$ and $\mathcal{D}_1$ be Reed-Muller domains with a finite field $\mathbb{F}$. Let $\mathcal{D}^\diamond$ and $\mathcal{D}_1^\diamond$ be the RM⋄Had domains associated with $\mathcal{D}$ and $\mathcal{D}_1$, respectively, where the subfield is $\mathbb{L} \subseteq \mathbb{F}$. Denote the decoding degree of $\mathcal{D}$ by d' and the decoding degree of $\mathcal{D}_1$ by d_1'. Denote the dimension of $\mathcal{D}_1$ by w.*

Let k and N be natural numbers. Let $0 < \delta_{min,out}, \delta_{min,in} < 1$. Let $l_{max,out}, l_{max,in} : (0,1) \to \mathbb{R}^+$ be decreasing functions. Assume that $l_{max,in}(\delta), l_{max,out}(\delta) \leq \delta^{-O(1)}$.

For a sufficiently small constant $c > 0$, set

$$\delta_{min} \doteq \max\left\{\delta_{min,in}^c, \delta_{min,out}^c, \left(\frac{d'}{|\mathbb{F}|}\right)^c, \left(\frac{1}{|\mathbb{L}|}\right)^c\right\}$$

Given:

- *A $(\mathcal{D}, k, N)$-RM-RR construction algorithm $\mathcal{A}_{out}$ with structural parameters $(\mathsf{size_{out}}, \mathsf{block_{out}}, \mathsf{degleft_{out}}, \mathsf{degright_{out}}, \mathsf{depth_{out}})$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$, where the depth $\mathsf{depth_{out}}$ is constant and smaller than d'_1.*

- *A $(\mathcal{D}_1^\diamond, \mathsf{degleft_{out}} \cdot k + \mathsf{depth_{out}} + 1, |\mathbb{F}|^{w+1})$-RM◇Had-LR construction algorithm $\mathcal{A}_{in}$ with structural parameters $(\mathsf{size_{in}}, \mathsf{block_{in}}, \mathsf{degleft_{in}}, \mathsf{degright_{in}})$ that is uniform in the tuple association and in the encoding and list decoding. The output of the algorithm is always right regular.*

One can obtain a $(\mathcal{D}^\diamond, k, N)$-construction algorithm $\mathcal{A}$ that outputs (left and right) regular edge reading bipartite locally decode/reject codes. The algorithm has structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ for $\mathsf{size} \leq \mathsf{size_{out}} \cdot \mathsf{size_{in}}$, $\mathsf{block} \leq \mathsf{degleft_{out}} \cdot \mathsf{block_{in}}$, $\mathsf{degleft} \leq \mathsf{degleft_{out}} \cdot \mathsf{degleft_{in}}$ and $\mathsf{degright} \leq \mathsf{degright_{out}} \cdot \mathsf{degright_{in}}$.

If $\mathcal{A}_{out}$ outputs $(\delta_{min,out}, l_{max,out})$-RM-RRs, and $\mathcal{A}_{in}$ outputs $(\delta_{min,in}, l_{max,in})$-RM◇Had-LRs, then $\mathcal{A}$ outputs (δ_{min}, l_{max})-edge reading bipartite locally decode/reject codes for some $l_{max}(\delta) \leq \delta^{-O(1)}$.

9 Putting The Pieces Together

In this section we show how to put the pieces together to construct a bipartite locally decode/reject code for k-tuples that has small block length and almost-linear size. We start by outlining our plan. Then we construct each of the components. Finally, we set the parameters and get the final code.

Our plan is to construct an edge reading bipartite locally decode/reject code with small block length and small size, and then derive from it the code we want. The construction is as follows:

1. **Devise outer RM-RR construction algorithm:** We devise an RM-RR construction algorithm that has small size parameter, small left and right degree parameters, small depth parameter, but large block length parameter. Specifically, the block length is polynomial in k, d, $\log |\mathbb{F}|$ and the left and right degrees.

2. **Devise inner RM-RPR construction algorithm:** We devise an RM-RPR construction algorithms that is uniform in the point association and has large size parameter, small left and right degree parameters, small depth parameter and large block length parameter.

We will invoke the inner construction algorithm only on the alphabet domain of outer RM-RRs, and hence the large size and block length would be small in the context of the global construction.

3. **Compose the RM-RR and RM-RPR construction algorithms:** We compose the outer RM-RR construction algorithm from Step 1 with an appropriate inner RM-RPR construction algorithm from Step 2 to get an RM-RR construction algorithm with smaller block length.

   Specifically, the block length would be polynomial in k, $\log d$, $\log |\mathbb{F}|$ and the left and right degrees. Notably, the dependence on d is much smaller than in the block length of the outer RM-RR.

4. **Devise inner RM◇Had-LR construction algorithm:** We devise an RM◇Had-LR construction algorithm that is uniform in the tuple association and in the encoding and list decoding. It has very large size parameter and large left and right degrees.

   We will invoke the inner construction algorithm only on the alphabet domain of composed RM-RRs, and hence the large parameters would be small in the context of the global construction.

5. **Compose the RM-RR and RM◇Had-LR construction algorithms:** We compose the RM-RR construction algorithm from Step 3 with an appropriate inner RM◇Had-LR construction algorithm from Step 4 to get an RM◇Had-LR construction algorithm with smaller block length.

   Specifically, the block length would not depend at all on the degree d or the field $\mathbb{F}$ of the outer Reed-Muller code.
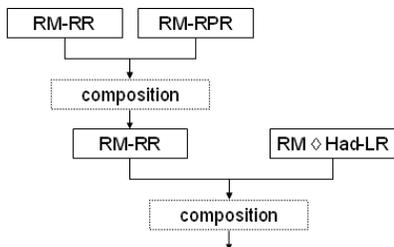

Figure 7: The outline of the construction.

9.1 Outer RM-RR Construction Algorithm

The outer RM-RR construction algorithm is as follows:

Lemma 9.1 (Outer RM-RR construction algorithm). *There is a global constant $c_0 \geq 1$, as well as a function $T : \mathbb{N} \to \mathbb{N}^+$ with $T(\Delta) = \Theta(\Delta)$ (as in Lemma 8.5) as follows.*

Let $\mathcal{D}$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension $m > 4$, an encoding degree d and a decoding degree d'. Let $\mathbb{K} \subseteq \mathbb{F}$ be a subfield of $\mathbb{F}$. Let $1 \leq k \leq \frac{|\mathbb{F}|}{2}$, N and Δ be natural numbers. We assume that the following condition holds:

- $d' \geq c_0(k+1)d \log((k+1)d)$.

There is a Reed-Muller domain $\widetilde{\mathcal{D}}$ defined by the field $\mathbb{F}$, a dimension $4 < \widetilde{m} \leq c_0 \cdot \log((k+1)d)$, an encoding degree $\widetilde{m}$ and a decoding degree $\lfloor d'/((k+1)d) \rfloor$, as well as a $(\mathcal{D}, k, N)$-RM-RR construction algorithm with structural parameters (size, block, degleft, degright, depth) *reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for* size $\leq (N + |\mathbb{F}^m|) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m} \cdot \Delta^2$, block $\leq \mathrm{poly}(k, d, \Delta) \cdot \log |\mathbb{F}|$, degleft $= T(\Delta)^2$, degright $= T(\Delta)$ *and* depth $= 1$. *The algorithm outputs (δ_{min}, l_{max})-RM-RRs for*

$$\delta_{min} \leq \max \left\{ \left(\frac{d'k}{|\mathbb{F}|}\right)^{\Omega(1)}, m^{O(1)} \left(\left(\frac{1}{|\mathbb{K}|}\right)^{\Omega(1)} + \left(\frac{d'}{|\mathbb{F}|}\right)^{\Omega(1)} \right), \left(\frac{1}{\Delta}\right)^{\Omega(1)} \right\}$$

and $l_{max}(\delta) \doteq \frac{2}{\delta^{15}}$.

Proof. The algorithm is obtained as follows:

1. **Generation of RM-LRs:** Let $\mathcal{D}_1$ be the Reed-Muller domain defined by the field $\mathbb{F}$, dimension 4, encoding degree $(k+1) \cdot d$ and decoding degree d'. Invoke Lemma 8.2 to obtain a $(\mathcal{D}, k, N)$-RM-LR construction algorithm $\mathcal{A}_1$ with structural parameters (size$_1$, block$_1$, degleft$_1$, degright$_1$) reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for size$_1 \leq (N + |\mathbb{F}|^m) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m}$, block$_1 \leq \mathrm{poly}(k, d) \cdot \log |\mathbb{F}|$, degleft$_1 \leq |\mathbb{F}|^{O(1)}$ and degright$_1 \leq (N + |\mathbb{F}|^m) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m}$. The lemma guarantees that $\mathcal{A}_1$ outputs $(\delta_{min,1}, l_{max,1})$-RM-LRs for

$$\delta_{min,1} \doteq \max \left\{ \sqrt{\frac{d' \cdot (k+1)}{|\mathbb{F}| - k}}, m \left(\sqrt[8]{\frac{1}{|\mathbb{K}|}} + \sqrt[4]{\frac{md'}{|\mathbb{F}|}} \right) \right\}$$

   and $l_{max,1}(\delta) \doteq \frac{2}{\delta}$.

2. **Power reduction:** Define $b \doteq \lceil \log((k+1) \cdot d + 1) \rceil$ and $\widetilde{m} \doteq 4b$. Let $c_0 \geq 1$ be such that $\widetilde{m} \leq c_0 \log((k+1)d)$. By assumption, $d' \geq 4b(k+1) \cdot d$. Let $\widetilde{\mathcal{D}}$ be the Reed-Muller domain defined by the field $\mathbb{F}$, dimension $\widetilde{m}$, encoding degree $\widetilde{m}$ and decoding degree $\lfloor d'/((k+1)d) \rfloor$. Invoke Lemma 8.4 (1) on $\mathcal{A}_1$ to obtain a $(\mathcal{D}, k, N)$-RM-LR construction algorithm $\mathcal{A}_2$ with structural parameters (size$_2$, block$_2$, degleft$_2$, degright$_2$) reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$, where size$_2 = $ size$_1$, block$_2 = \mathrm{poly}(k, d) \cdot \log |\mathbb{F}|$, degleft$_2 = $ degleft$_1$ and degright$_2 = $ degright$_1$. The lemma guarantees that $\mathcal{A}_2$ outputs $(\delta_{min,2}, l_{max,2})$-RM-LRs, where $\delta_{min,2} = \delta_{min,1}$ and $l_{max,2} \equiv l_{max,1}$.

3. **Right degree reduction:** Let $\alpha < 1$ and $T : \mathbb{N} \to \mathbb{N}^+$ with $T(\Delta) = \Theta(\Delta)$ be as in Lemma 8.5. Invoke Lemma 8.5 (1) on $\mathcal{A}_2$ to obtain a $(\mathcal{D}, k, N)$-RM-LR construction algorithm $\mathcal{A}_3$ with structural parameters (size$_3$, block$_3$, degleft$_3$, degright$_3$) reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$,

where $\text{size}_3 = (N + |\mathbb{F}|^m) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m} \cdot \Delta$, $\text{block}_3 = \text{poly}(k, d) \cdot \log |\mathbb{F}|$, $\text{degleft}_3 = T(\Delta) \cdot \text{degleft}_1$ and $\text{degright}_3 = T(\Delta)$. The lemma guarantees that the algorithm $\mathcal{A}_3$ outputs right regular $(\delta_{min,3}, l_{max,3})$-RM-LRs, for

$$\delta_{min,3} \doteq \max\left\{ \sqrt[4]{\frac{d' \cdot (k+1)}{|\mathbb{F}| - k}}, \sqrt{m}\left(\sqrt[16]{\frac{1}{|\mathbb{K}|}} + \sqrt[8]{\frac{md'}{|\mathbb{F}|}} \right), \frac{1}{T(\Delta)^{1-\alpha}} \right\}$$

and $l_{max,3}(\delta) \doteq \frac{2}{\delta^3}$.

4. **Switching sides:** Invoke Lemma 8.6 (1) on $\mathcal{A}_3$ to obtain a $(\mathcal{D}, k, N)$-RM-RR construction algorithm $\mathcal{A}_4$ with structural parameters $(\text{size}_4, \text{block}_4, \text{degleft}_4, \text{degright}_4, \text{depth}_4)$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$, where $\text{size}_4 = (N + |\mathbb{F}|^m) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m} \cdot \Delta$, $\text{block}_4 = \text{poly}(k, d, \Delta) \cdot \log |\mathbb{F}|$, $\text{degleft}_4 = T(\Delta)$, $\text{degright}_4 = T(\Delta) \cdot \text{degleft}_1$ and $\text{depth}_4 = 1$. The lemma guarantees that the algorithm $\mathcal{A}_4$ outputs $(\delta_{min,4}, l_{max,4})$-RM-RRs, for

$$\delta_{min,4} \doteq \max\left\{ \sqrt[8]{\frac{d' \cdot (k+1)}{|\mathbb{F}| - k}}, \sqrt[4]{m}\left(\sqrt[32]{\frac{1}{|\mathbb{K}|}} + \sqrt[16]{\frac{md'}{|\mathbb{F}|}} \right), \frac{1}{T(\Delta)^{\frac{1-\alpha}{2}}} \right\}$$

and $l_{max,4}(\delta) \doteq \frac{2}{\delta^7}$.

5. **Right degree reduction:** Invoke Lemma 8.5 (4) on $\mathcal{A}_4$ to obtain a $(\mathcal{D}, k, N)$-RM-RR construction algorithm $\mathcal{A}_5$ with structural parameters $(\text{size}_5, \text{block}_5, \text{degleft}_5, \text{degright}_5, \text{depth}_5)$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$, where $\text{size}_5 = (N + |\mathbb{F}|^m) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m} \cdot \Delta^2$, $\text{block}_5 = \text{poly}(k, d, \Delta) \cdot \log |\mathbb{F}|$, $\text{degleft}_5 = T(\Delta)^2$, $\text{degright}_5 = T(\Delta)$ and $\text{depth}_5 = 1$. The lemma guarantees that the algorithm $\mathcal{A}_5$ outputs $(\delta_{min,5}, l_{max,5})$-RM-RRs, for

$$\delta_{min,5} \doteq \max\left\{ \sqrt[16]{\frac{d' \cdot (k+1)}{|\mathbb{F}| - k}}, \sqrt[8]{m}\left(\sqrt[64]{\frac{1}{|\mathbb{K}|}} + \sqrt[32]{\frac{md'}{|\mathbb{F}|}} \right), \frac{1}{T(\Delta)^{\frac{1-\alpha}{4}}} \right\}$$

and $l_{max,5}(\delta) \doteq \frac{2}{\delta^{15}}$.

$\square$

9.2 Inner RM-RPR Construction Algorithm

The inner RM-RPR construction algorithm is obtained similarly to the way the outer RM-RR construction algorithm is obtained. The parameters resemble those of Lemma 9.1, except for the size parameter which is larger.

Lemma 9.2 (Inner RM-RPR construction algorithm). *There is a global constants $c_0 \geq 1$, as well as a function $T : \mathbb{N} \to \mathbb{N}^+$ with $T(\Delta) = \Theta(\Delta)$ (same as in Lemma 9.1) as follows.*

Let $\mathcal{D}$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension $m > 4$, an encoding degree d and a decoding degree d'. Let $1 \leq k \leq \frac{|\mathbb{F}|}{2}$, N and Δ be natural numbers. We assume that the following condition holds:

- $d' \geq c_0(k+1)d \log((k+1)d)$.

There is a Reed-Muller domain $\widetilde{\mathcal{D}}$ defined by the field $\mathbb{F}$, a dimension $\widetilde{m} \leq c_0 \cdot \log((k+1)d)$, encoding degree $\widetilde{m}$ (where $\widetilde{m}$ is as in Lemma 9.1) and decoding degree $\lfloor d'/((k+1)d) \rfloor$, as well as a $(\mathcal{D}, k, N)$-RM-RPR construction algorithm with structural parameters (size, block, degleft, degright, depth) *reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for* size $\leq N \cdot |\mathbb{F}|^{O(m)} \cdot \Delta^2$, block $\leq$ poly$(k, d, \Delta) \cdot \log|\mathbb{F}|$, degleft $= T(\Delta)^2$, degright $= T(\Delta)$ *and* depth $= 1$. *The algorithm is uniform in the point association and outputs* (δ_{min}, l_{max})-*RM-RPRs for*

$$\delta_{min} \leq \max\left\{ \left(\frac{d'k}{|\mathbb{F}|}\right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{d'}{|\mathbb{F}|}\right)^{\Omega(1)}, \left(\frac{1}{\Delta}\right)^{\Omega(1)} \right\}$$

and $l_{max}(\delta) \doteq \frac{2}{\delta^{15}}$.

Proof. The algorithm is obtained as follows:

1. **Generation of RM-LPRs:** Let $\mathcal{D}_1$ be the Reed-Muller domain defined by the field $\mathbb{F}$, dimension 4, encoding degree $(k+1) \cdot d$ and decoding degree d'. Invoke Lemma 8.1 to obtain a $(\mathcal{D}, k, N)$-RM-LPR construction algorithm $\mathcal{A}_1$ with structural parameters (size$_1$, block$_1$, degleft$_1$, degright$_1$) reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for size$_1 \leq N \cdot |\mathbb{F}|^{O(m)}$, block$_1 \leq$ poly$(k, d) \cdot \log|\mathbb{F}|$, degleft$_1 \leq |\mathbb{F}|^{O(1)}$ and degright$_1 \leq N \cdot |\mathbb{F}|^{O(m)}$. The lemma guarantees that $\mathcal{A}_1$ is uniform in the point association and outputs $(\delta_{min,1}, l_{max,1})$-RM-LPRs for

$$\delta_{min,1} \doteq \max\left\{ \sqrt{\frac{d' \cdot (k+1)}{|\mathbb{F}| - k}}, m\left(\sqrt[8]{\frac{1}{|\mathbb{F}|}} + \sqrt[4]{\frac{md'}{|\mathbb{F}|}}\right) \right\}$$

and $l_{max,1}(\delta) \doteq \frac{2}{\delta}$.

2. **Power reduction:** Define $b \doteq \lceil \log((k+1) \cdot d + 1) \rceil$ and $\widetilde{m} \doteq 4b$. Let $c_0 \geq 1$ be such that $\widetilde{m} \leq c_0 \log((k+1)d)$. By assumption, $d' \geq 4b(k+1) \cdot d$. Let $\widetilde{\mathcal{D}}$ be the Reed-Muller domain defined by the field $\mathbb{F}$, dimension $\widetilde{m}$, encoding degree $\widetilde{m}$ and decoding degree $\lfloor d'/((k+1)d) \rfloor$. Invoke Lemma 8.4 (2) on $\mathcal{A}_1$ to obtain a $(\mathcal{D}, k, N)$-RM-LPR construction algorithm $\mathcal{A}_2$ with structural parameters (size$_2$, block$_2$, degleft$_2$, degright$_2$) reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$, where size$_2 =$ size$_1$, block$_2 =$ poly$(k, d) \cdot \log|\mathbb{F}|$, degleft$_2 =$ degleft$_1$ and degright$_2 =$ degright$_1$. The lemma guarantees that $\mathcal{A}_2$ is uniform in the point association and outputs $(\delta_{min,2}, l_{max,2})$-RM-LPRs, where $\delta_{min,2} = \delta_{min,1}$ and $l_{max,2} \equiv l_{max,1}$.

3. **Right degree reduction:** Invoke Lemma 8.5 (3) on $\mathcal{A}_2$ to obtain a $(\mathcal{D}, k, N)$-RM-LPR construction algorithm $\mathcal{A}_3$ with structural parameters (size$_3$, block$_3$, degleft$_3$, degright$_3$) reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$, where size$_3 = N \cdot |\mathbb{F}|^{O(m)} \cdot \Delta$, block$_3 =$ poly$(k, d) \cdot \log|\mathbb{F}|$, degleft$_3 = T(\Delta) \cdot$ degleft$_1$

and $\text{degright}_3 = T(\Delta)$. The lemma guarantees that the algorithm $\mathcal{A}_3$ is uniform in the point association and outputs $(\delta_{min,3}, l_{max,3})$-RM-LPRs, for

$$\delta_{min,3} \doteq \max \left\{ \sqrt[4]{\frac{d' \cdot (k+1)}{|\mathbb{F}| - k}}, \sqrt{m} \left(\sqrt[16]{\frac{1}{|\mathbb{F}|}} + \sqrt[8]{\frac{md'}{|\mathbb{F}|}} \right), \frac{1}{T(\Delta)^{1-\alpha}} \right\}$$

and $l_{max,3}(\delta) \doteq \frac{2}{\delta^3}$.

4. **Switching sides:** Invoke Lemma 8.6 (2) on $\mathcal{A}_3$ to obtain a $(\mathcal{D}, k, N)$-RM-RPR construction algorithm $\mathcal{A}_4$ with structural parameters $(\text{size}_4, \text{block}_4, \text{degleft}_4, \text{degright}_4, \text{depth}_4)$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$, where $\text{size}_4 = N \cdot |\mathbb{F}|^{O(m)} \cdot \Delta$, $\text{block}_4 = \text{poly}(k, d, \Delta) \cdot \log |\mathbb{F}|$, $\text{degleft}_4 = T(\Delta)$, $\text{degright}_4 = T(\Delta) \cdot \text{degleft}_1$ and $\text{depth}_4 = 1$. The lemma guarantees that the algorithm $\mathcal{A}_4$ is uniform in the point association and outputs $(\delta_{min,4}, l_{max,4})$-RM-RPRs, for

$$\delta_{min,4} \doteq \max \left\{ \sqrt[8]{\frac{d' \cdot (k+1)}{|\mathbb{F}| - k}}, \sqrt[4]{m} \left(\sqrt[32]{\frac{1}{|\mathbb{F}|}} + \sqrt[16]{\frac{md'}{|\mathbb{F}|}} \right), \frac{1}{T(\Delta)^{\frac{1-\alpha}{2}}} \right\}$$

and $l_{max,4}(\delta) \doteq \frac{2}{\delta^7}$.

5. **Right degree reduction:** Invoke Lemma 8.5 (5) on $\mathcal{A}_4$ to obtain a $(\mathcal{D}, k, N)$-RM-RPR construction algorithm $\mathcal{A}_5$ with structural parameters $(\text{size}_5, \text{block}_5, \text{degleft}_5, \text{degright}_5, \text{depth}_5)$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$, where $\text{size}_5 = N \cdot |\mathbb{F}|^{O(m)} \cdot \Delta^2$, $\text{block}_5 = \text{poly}(k, d, \Delta) \cdot \log |\mathbb{F}|$, $\text{degleft}_5 = T(\Delta)^2$, $\text{degright}_5 = T(\Delta)$ and $\text{depth}_5 = 1$. The lemma guarantees that the algorithm $\mathcal{A}_5$ is uniform in the point association and outputs $(\delta_{min,5}, l_{max,5})$-RM-RPRs, for

$$\delta_{min,5} \doteq \max \left\{ \sqrt[16]{\frac{d' \cdot (k+1)}{|\mathbb{F}| - k}}, \sqrt[8]{m} \left(\sqrt[64]{\frac{1}{|\mathbb{F}|}} + \sqrt[32]{\frac{md'}{|\mathbb{F}|}} \right), \frac{1}{T(\Delta)^{\frac{1-\alpha}{4}}} \right\}$$

and $l_{max,5}(\delta) \doteq \frac{2}{\delta^{15}}$.

$\square$

9.3 Composition of The RM-RR and RM-RPR Construction Algorithms

Composing the outer RM-RR construction algorithm of Lemma 9.1 with the inner RM-RPR construction algorithm of Lemma 9.2 we get the following:

Lemma 9.3 (Final RM-RR construction algorithm). *Let $T : \mathbb{N} \to \mathbb{N}^+$ with $T(\Delta) = \Theta(\Delta)$ (as in Lemma 8.5). There is a polynomial $q(\cdot, \cdot, \cdot)$ as follows.*

Let $\mathcal{D}$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension $m > 4$, an encoding degree d and a decoding degree d'. Let $\mathbb{K} \subseteq \mathbb{F}$ be a subfield of $\mathbb{F}$. Let k, N and Δ be natural numbers, where $1 \le k \le \frac{|\mathbb{F}|}{2} - T(\Delta)$.

We assume that the following conditions hold:

- $d' \geq d \cdot q(k, \log d, \Delta)$.

- $\log kd \leq m^{o(1)}$.

There is a Reed-Muller domain $\widetilde{\mathcal{D}}$ with field $\mathbb{F}$ and the same dimension and encoding degree which is at most $O(\log(k + \Delta) + \log\log d)$, as well as a $(\mathcal{D}, k, N)$-RM-RR construction algorithm with structural parameters (size, block, degleft, degright, depth) *reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for* size $\leq (N + |\mathbb{F}^m|) \cdot |\mathbb{F}|^{m^{o(1)}} \cdot |\mathbb{K}|^{2m} \cdot \Delta^4$, block $\leq \mathsf{poly}(k, \log d, \Delta) \cdot \log|\mathbb{F}|$, degleft $= T(\Delta)^4$, degright $= T(\Delta)^2$ *and* depth $= 2$. *The algorithm outputs (δ_{min}, l_{max})-RM-RRs where*

$$\delta_{min} \leq \max\left\{ \left(\frac{d'k}{|\mathbb{F}|}\right)^{\Omega(1)}, \left(\frac{d'\Delta}{|\mathbb{F}|}\right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{1}{|\mathbb{K}|}\right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{d'}{|\mathbb{F}|}\right)^{\Omega(1)}, \left(\frac{1}{\Delta}\right)^{\Omega(1)} \right\}$$

and $l_{max}(\delta) \leq \frac{1}{\delta^{O(1)}}$.

Proof. Let c_0 be the global constant from Lemma 9.1. Let us choose q such that:

$$d' \geq c_0(k+1)d\log((k+1)d) \tag{1}$$

(q should also satisfy requirement (2) below).

Invoke Lemma 9.1 on the domain $\mathcal{D}$, the subfield $\mathbb{K}$ and the natural numbers k, N and Δ. Let $\mathcal{D}_1$ be the Reed-Muller domain guaranteed in the lemma. The domain $\mathcal{D}_1$ has dimension and encoding degree $4 < m_1 \leq c_0 \log((k+1)d)$. Its decoding degree is $\lfloor d'/((k+1)d)\rfloor$. Let $\mathcal{A}_{out}$ be the $(\mathcal{D}, k, N)$-RM-RR construction algorithm with structural parameters
(size$_{out}$, block$_{out}$, degleft$_{out}$, degright$_{out}$, depth$_{out}$) reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for size$_{out} \leq (N + |\mathbb{F}^m|) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m} \cdot \Delta^2$, block$_{out} \leq \mathsf{poly}(k, d, \Delta) \cdot \log|\mathbb{F}|$, degleft$_{out} = T(\Delta)^2$, degright$_{out} = T(\Delta)$ and depth$_{out} = 1$. The algorithm outputs $(\delta_{min,out}, l_{max,out})$-RM-RRs for

$$\delta_{min,out} \leq \max\left\{ \left(\frac{d'k}{|\mathbb{F}|}\right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{1}{|\mathbb{K}|}\right)^{\Omega(1)} + m^{O(1)} \cdot \left(\frac{d'}{|\mathbb{F}|}\right)^{\Omega(1)}, \left(\frac{1}{\Delta}\right)^{\Omega(1)} \right\}$$

and $l_{max,out}(\delta) \doteq \frac{2}{\delta^{15}}$.

We choose q such that

$$\left\lfloor \frac{d'}{(k+1)d} \right\rfloor \geq c_0(k + T(\Delta) + 1) \cdot m_1 \log((k + T(\Delta) + 1) \cdot m_1) \tag{2}$$

Invoke Lemma 9.2 on the domain $\mathcal{D}_1$ and the natural numbers $k + T(\Delta)$, 1 and Δ. Let $\widetilde{\mathcal{D}}$ be the Reed-Muller domain guaranteed in the lemma. The domain $\widetilde{\mathcal{D}}$ has the same dimension and encoding degree which is at most $c_0 \log((k + T(\Delta) + 1)m_1) = O(\log(k + \Delta) + \log\log d)$. Let $\mathcal{A}_{in}$ be the $(\mathcal{D}_1, k + T(\Delta), 1)$-RM-RPR construction algorithm with structural parameters
(size$_{in}$, block$_{in}$, degleft$_{in}$, degright$_{in}$, depth$_{in}$) reducing $\mathcal{D}_1 \mapsto \widetilde{\mathcal{D}}$, for size$_{in} \leq |\mathbb{F}|^{O(m_1)} \cdot \Delta^2$, block$_{in} \leq$

$\mathsf{poly}(k, \log d, \Delta) \cdot \log |\mathbb{F}|$, $\mathsf{degleft_{in}} = T(\Delta)^2$, $\mathsf{degright_{in}} = T(\Delta)$ and $\mathsf{depth_{in}} = 1$. The algorithm is uniform in the point association and outputs $(\delta_{min,in}, l_{max,in})$-RM-RPRs for

$$\delta_{min,in} \le \max \left\{ \left(\frac{d'\Delta}{d\,|\mathbb{F}|} \right)^{\Omega(1)}, m_1^{O(1)} \cdot \left(\frac{d'}{kd\,|\mathbb{F}|} \right)^{\Omega(1)}, \left(\frac{1}{\Delta} \right)^{\Omega(1)} \right\}$$

and $l_{max,in}(\delta) \doteq \frac{2}{\delta^{15}}$.

Apply the composition lemma (Lemma 8.8) on $\mathcal{A}_{out}$ and $\mathcal{A}_{in}$. Let $\mathcal{A}$ be the $(\mathcal{D}, k, N)$-RM-RR construction algorithm reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ obtained from the lemma. The algorithm $\mathcal{A}$ has structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright}, \mathsf{depth})$ for:

- $\mathsf{size} \le \mathsf{size_{out}} \cdot \mathsf{size_{in}} \le (N + |\mathbb{F}^m|) \cdot |\mathbb{F}|^{m^{o(1)}} \cdot |\mathbb{K}|^{2m} \cdot \Delta^4$.

- $\mathsf{block} = \mathsf{degleft_{out}} \cdot \mathsf{block_{in}} \le \mathsf{poly}(k, \log d, \Delta) \cdot \log |\mathbb{F}|$.

- $\mathsf{degleft} = \mathsf{degleft_{out}} \cdot \mathsf{degleft_{in}} = T(\Delta)^4$.

- $\mathsf{degright} = \mathsf{degright_{out}} \cdot \mathsf{degright_{in}} = T(\Delta)^2$.

- $\mathsf{depth} = \mathsf{depth_{out}} + 1 = 2$.

The algorithm outputs (δ_{min}, l_{max})-RM-RRs where

$$\delta_{min} \le \max \left\{ \left(\frac{d'k}{|\mathbb{F}|} \right)^{\Omega(1)}, \left(\frac{d'\Delta}{|\mathbb{F}|} \right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{1}{|\mathbb{K}|} \right)^{\Omega(1)} + m^{O(1)} \cdot \left(\frac{d'}{|\mathbb{F}|} \right)^{\Omega(1)}, \left(\frac{1}{\Delta} \right)^{\Omega(1)} \right\}$$

and $l_{max}(\delta) = \frac{1}{\delta^{O(1)}} \cdot l_{max,out}(\delta^{O(1)}) \le \frac{1}{\delta^{O(1)}}$. $\qquad\square$

9.4 Inner RM⋄Had-LR Construction Algorithm

The inner RM⋄Had-LR construction algorithm is obtained from the Had-LR construction in Lemma 8.3 and the transformation to RM⋄Had-LRs in Lemma 8.7. It is transformed to a construction algorithm that outputs right regular RM⋄Had-LRs via the transformation of Lemma 8.5.

Lemma 9.4 (Inner RM⋄Had-LR construction algorithm). *Let $\mathcal{D}^\diamond$ be an RM⋄Had domain defined by a finite field $\mathbb{F}$, a dimension m, an encoding degree d, a decoding degree d' and a prime subfield $\mathbb{L}$, where the extension degree is $\tau = [\mathbb{F} : \mathbb{L}]$. Set $M \doteq \binom{m+d}{m}$. Let $1 \le k \le M \cdot \tau - 2$ and N be natural numbers. Let Δ be a natural number.*

There is a $(\mathcal{D}^\diamond, k, N)$-RM⋄Had-LR construction algorithm with structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ *for* $\mathsf{size} \le N \cdot |\mathbb{F}|^{O(M)} \cdot \Delta$, $\mathsf{block} \le O(k) \cdot \log |\mathbb{L}|$, $\mathsf{degleft} \le \Delta \cdot |\mathbb{L}|^{O(k)}$ *and* $\mathsf{degright} \le O(\Delta)$. *The algorithm is uniform in the tuple association and in the encoding and list decoding and outputs right regular (δ_{min}, l_{max})-RM⋄Had-LRs for* $\delta_{min} \le \max \left\{ \left(\frac{1}{|\mathbb{L}|} \right)^{\Omega(1)}, \left(\frac{1}{\Delta} \right)^{\Omega(1)} \right\}$ *and* $l_{max}(\delta) \le \frac{1}{\delta^{O(1)}}$.

Proof. The algorithm is obtained as follows:

1. **Generation of Had-LRs.** Let $\mathcal{H}$ be the Hadamard domain defined by the field $\mathbb{L}$ and the dimension $M \cdot \tau$. Let $\mathcal{A}_1$ be the $(\mathcal{H}, k, N)$-Had-LR construction algorithm guaranteed by Lemma 8.3. The algorithm $\mathcal{A}_1$ has structural parameters $(\mathsf{size}_1, \mathsf{block}_1, \mathsf{degleft}_1, \mathsf{degright}_1)$ for $\mathsf{size}_1 \leq N \cdot |\mathbb{F}|^{O(M)}$, $\mathsf{block}_1 \leq O(k) \cdot \log|\mathbb{L}|$, $\mathsf{degleft}_1 \leq |\mathbb{L}|^{O(k)}$ and $\mathsf{degright}_1 \leq N \cdot |\mathbb{F}|^{O(M)}$. The algorithm is uniform in the tuple association and in the encoding and list decoding. It outputs $(\delta_{min,1}, l_{max,1})$-Had-LRs for $\delta_{min,1} \doteq 2\sqrt[6]{\frac{1}{|\mathbb{L}|}}$ and $l_{max,1}(\delta) \doteq \frac{2}{\delta^3}$. The right degrees of the vertices in the output do not depend on the input to the algorithm.

2. **Transformation to RM⋄Had-LRs.** Invoke Lemma 8.7 on $\mathcal{A}_1$ to obtain a $(\mathcal{D}^{\diamond}, k, N)$-RM⋄Had-LR construction algorithm $\mathcal{A}_2$. The algorithm $\mathcal{A}_2$ has structural parameters $(\mathsf{size}_2, \mathsf{block}_2, \mathsf{degleft}_2, \mathsf{degright}_2)$ for $\mathsf{size}_2 \leq N \cdot |\mathbb{F}|^{O(M)}$, $\mathsf{block}_2 \leq O(k) \cdot \log|\mathbb{L}|$, $\mathsf{degleft}_2 \leq |\mathbb{L}|^{O(k)}$ and $\mathsf{degright}_2 \leq N \cdot |\mathbb{F}|^{O(M)}$. The algorithm is uniform in the tuple association and in the encoding and list decoding. It outputs $(\delta_{min,2}, l_{max,2})$-RM⋄Had-LRs for $\delta_{min,2} \doteq 2\sqrt[6]{\frac{1}{|\mathbb{L}|}}$ and $l_{max,2}(\delta) \doteq \frac{2}{\delta^3}$. The right degrees of the vertices in the output do not depend on the input to the algorithm.

3. **Right degree reduction.** Invoke Lemma 8.5 (2) on $\mathcal{A}_2$ to obtain a $(\mathcal{D}^{\diamond}, k, N)$-RM⋄Had-LR construction algorithm $\mathcal{A}_3$ that outputs right regular RM⋄Had-LRs. The algorithm $\mathcal{A}_3$ has structural parameters $(\mathsf{size}_3, \mathsf{block}_3, \mathsf{degleft}_3, \mathsf{degright}_3)$ for $\mathsf{size}_3 \leq N \cdot |\mathbb{F}|^{O(M)} \cdot \Delta$, $\mathsf{block}_3 \leq O(k) \cdot \log|\mathbb{L}|$, $\mathsf{degleft}_3 \leq \Delta \cdot |\mathbb{L}|^{O(k)}$ and $\mathsf{degright}_3 \leq O(\Delta)$. The algorithm is uniform in the tuple association and in the encoding and list decoding. It outputs $(\delta_{min,3}, l_{max,3})$-RM⋄Had-LRs for $\delta_{min,3} \leq \max\left\{(\frac{1}{|\mathbb{L}|})^{\Omega(1)}, (\frac{1}{\Delta})^{\Omega(1)}\right\}$ and $l_{max,3}(\delta) \leq \frac{1}{\delta^{O(1)}}$.

$\square$

9.5 Composition of The RM-RR and RM⋄Had-LR Construction Algorithms

Composing the composed RM-RR construction algorithm of Lemma 9.3 with the inner RM⋄Had-LR construction algorithm of Lemma 9.4 we get the following:

Lemma 9.5 (Final RM⋄Had-LR construction algorithm). *Let $T : \mathbb{N} \to \mathbb{N}^+$ with $T(\Delta) = \Theta(\Delta)$ (as in Lemma 8.5). Let $q(\cdot, \cdot, \cdot)$ be the polynomial from Lemma 9.3.*

Let $\mathcal{D}^{\diamond}$ be an RM⋄Had domain defined by a finite field $\mathbb{F}$, a dimension $m > 4$, an encoding degree d, a decoding degree d' and a prime subfield $\mathbb{L} \subseteq \mathbb{F}$. Let $\mathbb{K} \subseteq \mathbb{F}$ be a subfield of $\mathbb{F}$. Let k, N and Δ be natural numbers, where $1 \leq k \leq \frac{|\mathbb{F}|}{2} - T(\Delta)$.

We assume that the following conditions hold:

- $d' \geq d \cdot q(k, \log d, \Delta)$.

- $\log kd \leq m^{o(1)}$.

There is a $(\mathcal{D}^\diamond, k, N)$-construction algorithm with structural parameters (size, block, degleft, degright) *for* $\text{size} \leq (N + |\mathbb{F}^m|) \cdot |\mathbb{F}|^{m^{o(1)} + \mathsf{poly}(k, \Delta, \log d)} \cdot |\mathbb{K}|^{2m}$, $\text{block} \leq \Delta^{O(1)} \cdot k \cdot \log |\mathbb{L}|$, $\text{degleft} \leq |\mathbb{L}|^{\Delta^{O(1)} \cdot k}$, $\text{degright} \leq \Delta^{O(1)}$. *The algorithm outputs left and right regular* (δ_{min}, l_{max})-*edge reading bipartite locally decode/reject codes, where*

$$\delta_{min} \leq \max\left\{ \left(\frac{1}{|\mathbb{L}|}\right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{d'k\Delta}{|\mathbb{F}|}\right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{1}{|\mathbb{K}|}\right)^{\Omega(1)}, \left(\frac{1}{\Delta}\right)^{\Omega(1)} \right\}$$

and $l_{max}(\delta) \leq \frac{1}{\delta^{O(1)}}$.

Proof. Let $\mathcal{D}$ be the Reed-Muller domain associated with $\mathcal{D}^\diamond$. Invoke Lemma 9.3 on the domain $\mathcal{D}$, the subfield $\mathbb{K}$ and the natural numbers k, N and Δ, to obtain a Reed-Muller domain $\mathcal{D}_1$ with field $\mathbb{F}$ and dimension and encoding degree $w = O(\log(k+\Delta) + \log \log d)$, as well as a $(\mathcal{D}, k, N)$-RM-RR construction algorithm $\mathcal{A}_{out}$ with structural parameters $(\text{size}_{out}, \text{block}_{out}, \text{degleft}_{out}, \text{degright}_{out}, \text{depth}_{out})$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for $\text{size}_{out} \leq (N + |\mathbb{F}^m|) \cdot |\mathbb{F}|^{m^{o(1)}} \cdot |\mathbb{K}|^{2m} \cdot \Delta^4$, $\text{block}_{out} \leq \mathsf{poly}(k, \log d, \Delta) \cdot \log |\mathbb{F}|$, $\text{degleft}_{out} = T(\Delta)^4$, $\text{degright}_{out} = T(\Delta)^2$ and $\text{depth}_{out} = 2$. The algorithm outputs $(\delta_{min,out}, l_{max,out})$-RM-RRs where

$$\delta_{min,out} \leq \max\left\{ \left(\frac{d'k}{|\mathbb{F}|}\right)^{\Omega(1)}, \left(\frac{d'\Delta}{|\mathbb{F}|}\right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{1}{|\mathbb{K}|}\right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{d'}{|\mathbb{F}|}\right)^{\Omega(1)}, \left(\frac{1}{\Delta}\right)^{\Omega(1)} \right\}$$

and $l_{max,out}(\delta) \leq \frac{1}{\delta^{O(1)}}$.

Let $\mathcal{D}_1^\diamond$ be the RM⋄Had domain associated with $\mathcal{D}_1$ for the subfield $\mathbb{L}$. Denote $\tau = [\mathbb{F} : \mathbb{L}]$. Denote $k' \doteq \text{degleft}_{out} \cdot k + \text{depth}_{out} + 1$. Let us assume without loss of generality that $k' \leq \binom{2w}{w} \cdot \tau - 2$ and that $w > 2$ (so, in particular, the decoding degree of $\mathcal{D}_1$ is larger than 2).

Invoke Lemma 9.4 on the domain $\mathcal{D}_1^\diamond$ and the natural numbers k' and $|\mathbb{F}|^{w+1}$ to obtain a $(\mathcal{D}_1^\diamond, k', |\mathbb{F}|^{w+1})$-RM⋄Had-LR construction algorithm $\mathcal{A}_{in}$ with structural parameters $(\text{size}_{in}, \text{block}_{in}, \text{degleft}_{in}, \text{degright}_{in})$ for $\text{size}_{in} \leq |\mathbb{F}|^{\mathsf{poly}(k, \Delta, \log d)}$, $\text{block}_{in} \leq \Delta^{O(1)} \cdot k \cdot \log |\mathbb{L}|$, $\text{degleft}_{in} \leq \Delta \cdot |\mathbb{L}|^{\Delta^{O(1)} \cdot k}$ and $\text{degright}_{in} \leq O(\Delta)$. The algorithm is uniform in the tuple association and in the encoding and list decoding, and outputs right regular $(\delta_{min,in}, l_{max,in})$-RM⋄Had-LRs where $\delta_{min,in} \leq \max\left\{ (\frac{1}{|\mathbb{L}|})^{\Omega(1)}, (\frac{1}{\Delta})^{\Omega(1)} \right\}$ and $l_{max,in}(\delta) \leq \frac{1}{\delta^{O(1)}}$.

Apply Lemma 8.9 on the outer algorithm $\mathcal{A}_{out}$ and the inner algorithm $\mathcal{A}_{in}$ to obtain a $(\mathcal{D}^\diamond, k, N)$-construction algorithm $\mathcal{A}$ with structural parameters (size, block, degleft, degright) for $\text{size} \leq (N + |\mathbb{F}^m|) \cdot |\mathbb{F}|^{m^{o(1)} + \mathsf{poly}(k, \Delta, \log d)} \cdot |\mathbb{K}|^{2m}$, $\text{block} \leq \Delta^{O(1)} \cdot k \cdot \log |\mathbb{L}|$, $\text{degleft} \leq |\mathbb{L}|^{\Delta^{O(1)} \cdot k}$, $\text{degright} \leq \Delta^{O(1)}$. The algorithm outputs left and right regular (δ_{min}, l_{max})-edge reading bipartite locally decode/reject codes, where

$$\delta_{min} \leq \max\left\{ \left(\frac{1}{|\mathbb{L}|}\right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{d'k\Delta}{|\mathbb{F}|}\right)^{\Omega(1)}, m^{O(1)} \cdot \left(\frac{1}{|\mathbb{K}|}\right)^{\Omega(1)}, \left(\frac{1}{\Delta}\right)^{\Omega(1)} \right\}$$

and $l_{max}(\delta) \leq \frac{1}{\delta^{O(1)}}$. $\qquad\square$

9.6 Setting Parameters

In this section we construct the bipartite locally decode/reject code we want:

Corollary 17 (Edge reading bipartite locally decode/reject code). *Let n be a natural number. Let $\mathcal{D}$ be the domain associated with the set of binary strings $\{0,1\}^n$. There exists a constant $0 < \alpha < \frac{1}{2}$ such that the following holds. Let k and N be natural numbers such that $k \leq (\log n)^\alpha$. Let $\frac{1}{(\log n)^\alpha} \leq \varepsilon < 1$. Then, for every collection of k-tuples*

$$\langle i_{1,1}, \ldots, i_{1,k} \rangle, \ldots, \langle i_{N,1}, \ldots, i_{N,k} \rangle \in [n]^k$$

there is a (left and right) regular (δ_{min}, l_{max})-bipartite locally decode/reject code, where $\delta_{min} \leq \varepsilon$ and $l_{max}(\delta) \leq \delta^{-O(1)}$. The size of the code is $(N + n) \cdot n^{o(1)}$ and the block length is $k \cdot \mathsf{poly}(\frac{1}{\varepsilon})$. The left degree is at most $2^{k \cdot \mathsf{poly}(\frac{1}{\varepsilon})}$. The right degree is at most $\mathsf{poly}(\frac{1}{\varepsilon})$. Moreover, the alphabet of the right vertices is of size $\mathsf{poly}(\frac{1}{\varepsilon})$.

Proof. We will choose a constant $0 < \alpha < \frac{1}{2}$ later. Let us choose the parameters of the RM⋄Had code we will use with respect to α. In all that follows we omit ceil and floor notation when we refer to natural numbers in order to ease the reading.

Let $h \doteq 2^{(\log n)^\alpha}$ and $m \doteq (\log n)^{1-\alpha}$ so that $h^m \geq n$. Assume without loss of generality that $m > 4$. Let $d \doteq (h-1)m$. Set $d' \doteq d \cdot q(k, \log d, \Delta)$ where q is as in Lemma 9.5.

Set $\Delta \leq (\frac{1}{\varepsilon})^{O(1)}$ large enough so that the term $(\frac{1}{\Delta})^{\Omega(1)}$ appearing in the expression for δ_{min} in Lemma 9.5 is at most ε. Let $\mathbb{L} = GF(p)$ where $p \leq (\frac{1}{\varepsilon})^{O(1)}$ is a prime number which is large enough so the term $(\frac{1}{|\mathbb{L}|})^{\Omega(1)}$ appearing in the expression for δ_{min} in Lemma 9.5 is at most ε. Let $\mathbb{K} = GF(p^{g_1})$ where g_1 is large enough so that the term $m^{O(1)} \cdot \left(\frac{1}{|\mathbb{K}|}\right)^{\Omega(1)}$ appearing in the expression for δ_{min} in Lemma 9.5 is at most ε. Let $\mathbb{F} \doteq GF(p^{g_1 \cdot g_2})$ where g_2 is large enough so that the term $m^{O(1)} \cdot \left(\frac{d'k\Delta}{|\mathbb{F}|}\right)^{\Omega(1)}$ appearing in the expression for δ_{min} in Lemma 9.5 is at most ε and $k \leq \frac{|\mathbb{F}|}{2} - T(\Delta)$ (where T is as in Lemma 9.5). Denote $\tau = g_1 \cdot g_2$. Let $\mathcal{D}^\diamond = \langle \mathbb{F}^m \times \mathbb{L}^\tau, \mathbb{L}, \mathcal{D}^\diamond_{enc}, \mathcal{D}^\diamond_{dec} \rangle$ be the determined RM⋄Had domain. Note that we can do all the above and have:

- $m \leq (\log n)^{1-\alpha}$

- $d \leq d' \leq 2^{(\log n)^\alpha + O(\log \log n)}$

- $|\mathbb{F}| \leq 2^{(\log n)^\alpha + O(\log \log n)}$

- $|\mathbb{K}| \leq 2^{O(\log \log n)}$

- $|\mathbb{L}|, \Delta \leq \mathsf{poly}(\frac{1}{\varepsilon})$

From Lemma 9.5 we get a construction algorithm $\mathcal{A}$ with structural parameters (size, block, degleft, degright) for:

- size $\leq (N + n) \cdot n^{o(1)} \cdot 2^{(\log n)^{O(\alpha)}}$

- block $\leq k \cdot \mathsf{poly}(\frac{1}{\varepsilon})$

- degleft $\leq 2^{k \cdot \mathsf{poly}(\frac{1}{\varepsilon})}$

- degright $\leq \mathsf{poly}(\frac{1}{\varepsilon})$

Let us choose α so that the $2^{(\log n)^{O(\alpha)}}$ term appearing in the expression for the size is $n^{o(1)}$.

The construction algorithm outputs left and right regular (δ_{min}, l_{max})-edge reading bipartite locally decode/reject codes, for $\delta_{min} \leq \varepsilon$ and $l_{max}(\delta) \leq \delta^{-O(1)}$. Looking into the construction reveals that the alphabet of the right vertices is $\mathbb{L}$.

Fix a set $H \subseteq \mathbb{F}$ of size $|H| = h$ such that $H^m \subseteq \mathbb{F}^m$ and $|H^m| \geq n$. Pick an arbitrary vector $\vec{\alpha} \in \mathbb{L}^\tau$. Let us identify $[n]$ with distinct elements $\langle \vec{x}, \vec{\alpha} \rangle \in H^m \times \mathbb{L}^\tau$. Let us consider the encoding of $\{0, 1\}^n$ that sends strings $x \in \{0, 1\}^n$ to codewords $f \in \mathcal{D}^\diamond_{enc}$ such that for every $i \in [n]$ we have $f(i) = x_i$. Note that this is possible by our choice of d.

Given a collection of size N of k-tuples

$$\langle i_{1,1}, \ldots, i_{1,k} \rangle, \ldots, \langle i_{N,1}, \ldots, i_{N,k} \rangle \in [n]^k$$

we invoke $\mathcal{A}$ on the corresponding collection of k-tuples in $\mathbb{F}^m \times \mathbb{L}^\tau$ to obtain the bipartite locally decode/reject code we want. $\qquad\square$

10 Construction of RM-LRs and RM-LPRs

In this section we present algorithms for constructing RM-LPRs and RM-LRs proving Lemma 8.1 and Lemma 8.2. The idea of the construction algorithms is to create a bipartite graph, in which side A corresponds to low degree manifolds in $\mathbb{F}^m$, each passing through an input tuple, and side B consists of all the points in $\mathbb{F}^m$. This way each A vertex has a tuple that is associated with it, and each B vertex has a point that is associated with it. We put edges between A vertices corresponding to manifolds and B vertices corresponding to points on them. Assignments to A vertices naturally project onto assignments to their neighboring B vertices.

A correct encoding of a polynomial $Q : \mathbb{F}^m \to \mathbb{F}$ assigns each B vertex the value of Q on the point associated with the B vertex, and assigns each A vertex the restriction of Q to the manifold associated with the A vertex. This way each A vertex can evaluate Q on the tuple associated with it, and each B vertex can evaluate Q on the point associated with it. Note that the restriction of Q to any manifold of a low degree is a polynomial of low degree.

To show that the construction has a list decoding property, we use a *low degree testing* theorem for sub-constant error. Specifically, we use the low degree testing theorem of [27], and choose the manifolds through the input tuples as to satisfy the conditions of this theorem. The advantage of

the theorem of [27] is that it can be used (together with an additional idea from [26]) to yield constructions of small size, as required in Lemma 8.2. For Lemma 8.1 that does not require particularly small size, we could have used other low degree testing theorems as well [3, 30].

10.1   A Low Degree Testing Theorem

We specify the variant of the low degree testing theorem of [27] we use. Details on the relation between this variant and the original test of [27] can be found in Appendix B.

The Randomness-Efficient Subspace vs. Point test.   Let $\mathbb{F}$ be a finite field. Let m and d' be natural numbers, where degree at most d' is considered "low degree". Let $\mathbb{K}$ be a subfield of $\mathbb{F}$.

Fix a function $f : \mathbb{F}^m \to \mathbb{F}$ we wish to test. If f were a polynomial of degree at most d', then for any $\vec{z}, \vec{y}_1, \vec{y}_2 \in \mathbb{F}^m$, the function $p(\mathbf{t_0}, \mathbf{t_1}, \mathbf{t_2}) = f(\mathbf{t_0}\vec{z} + \mathbf{t_1}\vec{y}_1 + \mathbf{t_2}\vec{y}_2)$ would have been a polynomial in three variables of degree at most d' over $\mathbb{F}$.

Assume access to an oracle $\mathcal{A}$ whose goal is to convince us that f is of degree at most d'. For every $\vec{z} \in \mathbb{F}^m$ and $\vec{y}_1, \vec{y}_2 \in \mathbb{K}^m$, the oracle $\mathcal{A}$ provides a three-variate polynomial of degree at most d', which is supposedly $p(\mathbf{t_0}, \mathbf{t_1}, \mathbf{t_2})$. The oracle $\mathcal{A}$ may be probabilistic, meaning that its answer may depend not only on $\vec{z}, \vec{y}_1, \vec{y}_2$, but also on additional randomness.

A test for checking that f is consistent with a polynomial of degree at most d' is described in Figure 8.

$LDT^{f, \mathcal{A}}$:

1. Pick uniformly at random three vectors $\langle \vec{z}, \vec{y}_1, \vec{y}_2 \rangle \in \mathbb{F}^m \times \mathbb{K}^m \times \mathbb{K}^m$. Using the oracle access to $\mathcal{A}$, obtain a three-variate polynomial $p^*(\mathbf{t_0}, \mathbf{t_1}, \mathbf{t_2})$ over $\mathbb{F}$ of degree at most d' for $\langle \vec{z}, \vec{y}_1, \vec{y}_2 \rangle$ [p^* is supposedly the restriction $p(\mathbf{t_0}, \mathbf{t_1}, \mathbf{t_2}) = f(\mathbf{t_0}\vec{z} + \mathbf{t_1}\vec{y}_1 + \mathbf{t_2}\vec{y}_2)$].

2. Pick uniformly at random $t_0 \neq 0, t_1, t_2 \in \mathbb{F}$. Set $\vec{z}_0 = t_0\vec{z} + t_1\vec{y}_1 + t_2\vec{y}_2$. If indeed $p^*(t_0, t_1, t_2) = f(\vec{z}_0)$, *accept*. Otherwise, *reject*.

Figure 8: Randomness-Efficient Subspace vs. Point Low Degree Tester

This test is similar to the test in [27]. The following follows from a slight strengthening of the statement of [27] appearing in [26] (see Appendix B):

Theorem 18 (Analysis of low degree test, [27, 26]). *For $\delta \geq m \left(\sqrt[8]{\frac{1}{|\mathbb{K}|}} + \sqrt[4]{\frac{md'}{|\mathbb{F}|}} \right)$, for any function $f : \mathbb{F}^m \to \mathbb{F}$, there are $l \leq \frac{2}{\delta}$ polynomials $Q_1, \dots, Q_l : \mathbb{F}^m \to \mathbb{F}$ of degree at most d', such that for any oracle $\mathcal{A}$ the following holds: the probability, over the randomness of $\mathcal{A}$ and over the randomness of the tester, that $LDT^{f, \mathcal{A}}$ accepts, although $f(\vec{z}_0) \notin \{Q_1(\vec{z}_0), \dots, Q_l(\vec{z}_0)\}$ (where $\vec{z}_0 \in \mathbb{F}^m$ is picked by the tester; see Figure 8), is at most $O(\delta)$.*

10.2 The Manifold vs. Point RM-LPR Construction Algorithm

The purpose of this section is to prove Lemma 8.1. We describe an RM-LPR construction algorithm that is uniform in the point association, called the Manifold vs. Point RM-LPR construction algorithm. We specify the properties of the algorithm in Subsection 10.2.1 and analyze it in Subsection 10.2.2.

The description of the algorithm will start with specifying the uniform part of the construction (i.e., the part that is common to all outputs of the algorithm), and proceed by presenting the components that are input-specific.

Construction 1 (Manifold vs. Point RM-LPR algorithm). *We use the notation and assume the restrictions appearing in Lemma 8.1.*

We define a uniform structure and uniform point association as follows:

A **vertices.** *The vertex set A consists of quadruplets $\langle i, \vec{x}, \vec{y}_1, \vec{y}_2 \rangle$ for $i \in [N]$ (indicating an input tuple) and $\vec{x}, \vec{y}_1, \vec{y}_2 \in \mathbb{F}^m$ (needed for the low degree test). We set $V \doteq A$ (for left evaluators) and $\Omega \doteq \mathbb{F}^w$ (recall that we set $w \doteq 4$).*

B **vertices.** *The vertex set B consists of all points $\vec{x} \in \mathbb{F}^m$. For every $b \in B$, we set $pnt(b) \doteq b$.*

Alphabets. *The alphabet Σ_A is the domain $\widetilde{\mathcal{D}}$ defined by the finite field $\mathbb{F}$, the dimension w, the encoding degree $(k+1) \cdot d$ and the decoding degree d'. The alphabet Σ_B is the domain associated with the set $\mathbb{F}$. Let us denote $\Sigma_A = \langle \mathbb{F}^w, \mathbb{F}, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$ and $\Sigma_B = \langle \{1\}, \mathbb{F}, \Sigma_{B,enc}, \Sigma_{B,dec} \rangle$.*

Given as input a collection of size N of k-tuples:

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

The construction algorithm constructs an RM-LPR

$$\mathcal{G} = \langle (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat \overset{\cdot}{\equiv} true, label, proj, tup, eval, pnt, evalp \rangle$$

for the k-tuples as follows:

Associating A vertices with manifolds. *A vertex $a = \langle i, \vec{x}, \vec{y}_1, \vec{y}_2 \rangle \in A$ is associated with the i'th input tuple $tup(a) \doteq \langle \vec{x}_{i,1}, \ldots, \vec{x}_{i,k} \rangle$, and corresponds to a manifold through the i'th tuple defined as follows. Fix arbitrarily $k+1$ different scalars in the field $q_1, \ldots, q_k, q_{k+1} \in \mathbb{F}$. Let $c_{i,\vec{x}} : \mathbb{F} \to \mathbb{F}^m$ denote the single curve of degree k that goes through $\vec{x}_{i,1}, \ldots, \vec{x}_{i,k}$ and $\vec{x}$ at $q_1, \ldots, q_k, q_{k+1}$:*

$$c_{i,\vec{x}}(q_1) = \vec{x}_{i,1}, \ \ldots, \ c_{i,\vec{x}}(q_k) = \vec{x}_{i,k}, \ c_{i,\vec{x}}(q_{k+1}) = \vec{x}$$

The vertex a is associated with the manifold of degree at most $k+1$

$$\mu_a(t, t_0, t_1, t_2) = t_0 \cdot c_{i,\vec{x}}(t) + t_1 \cdot \vec{y}_1 + t_2 \cdot \vec{y}_2$$

where we also denote $\mu_a \doteq \{\mu_a(t, t_0, t_1, t_2) \mid t, t_0, t_1, t_2 \in \mathbb{F}\}$. Note that each $i \in [N]$ has the same number of $a \in A$ with $tup(a) = \langle \vec{x}_{i,1}, \ldots, \vec{x}_{i,k} \rangle$.

Edges. *We connect every vertex $a = \langle i, \vec{x}, \vec{y}_1, \vec{y}_2 \rangle \in A$ to points in B that are on the manifold μ_a. The choice of the points is done as to match the low degree test in Subsection 10.1: for every $t \in \mathbb{F} \setminus \{q_1, \ldots, q_k\}$ and $t_0 \neq 0, t_1, t_2 \in \mathbb{F}$, there is an edge $e = (a, b) \in E$ connecting a to the point $b = t_0 \cdot c_{i,\vec{x}}(t) + t_1 \cdot \vec{y}_1 + t_2 \cdot \vec{y}_2 \in \mathbb{F}^m$ on μ_a. We set $label(e) \doteq (t, t_0, t_1, t_2) \in \Omega$.*

Projection. *For every vertex $a \in A$, assignment $\sigma_a \in \Sigma_{A,dec}$ (which is a w-variate polynomial over the field $\mathbb{F}$) and label $\xi \in \Omega$ (which is a point in $\mathbb{F}^w$), we let $proj(a, \sigma_a, \xi)$ be the element in $\Sigma_{B,enc} = \Sigma_{B,dec}$ corresponding to the field element $\sigma_a(\xi)$.*

Point evaluation. *For every vertex $b \in B$ and assignment $\sigma_b \in \Sigma_{B,dec}$, we let $evalp(b, \sigma_b)$ be the field element corresponding to σ_b.*

Tuple evaluation. *Denote $\vec{p}_1 \doteq (q_1, 1, 0, 0), \ldots, \vec{p}_k \doteq (q_k, 1, 0, 0) \in \mathbb{F}^w$. For every vertex $a \in A$, the tuple $tup(a)$ is on the manifold μ_a in positions $\vec{p}_1, \ldots, \vec{p}_k$, i.e., $tup(a) = \langle \mu_a(\vec{p}_1), \ldots, \mu_a(\vec{p}_k) \rangle$. For every vertex $a \in A$ and assignment $\sigma_a \in \Sigma_{A,dec}$ (which is a polynomial on the manifold μ_a), the evaluation of a on its tuple is given by the evaluation of σ_a on the points $\vec{p}_1, \ldots, \vec{p}_k$, i.e., we let $eval(a, \sigma_a) \doteq \langle \sigma_a(\vec{p}_1), \ldots, \sigma_a(\vec{p}_k) \rangle$.*

10.2.1 Properties of The Manifold vs. Point RM-LPR Construction Algorithm

Note that the algorithm is uniform in the point association. Additionally, the algorithm is efficient, and runs in time polynomial in $|\mathbb{F}^m|$ and N.

- *Size.* On all inputs, the output is of size $\mathsf{size} = N \cdot |\mathbb{F}|^{3m} + |\mathbb{F}^m| + N \cdot |\mathbb{F}|^{3m} \cdot |\mathbb{F}|^{O(1)} = N \cdot |\mathbb{F}|^{O(m)}$.

- *Block length.* On all inputs, the output has block length $\mathsf{block} = \log |\Sigma_{A,enc}| = \mathsf{poly}(k, d) \cdot \log |\mathbb{F}|$, since the number of monomials in a polynomial with $O(1)$ variables of degree at most $(k+1)d$ is $\mathsf{poly}(k, d)$, and for polynomials over a field $\mathbb{F}$, there are $|\mathbb{F}|$ possible coefficients for each monomial.

- *Left degree.* On all inputs, the output is left regular with left degree $\mathsf{degleft} = (|\mathbb{F}| - k) \cdot (|\mathbb{F}| - 1) \cdot |\mathbb{F}|^2 = |\mathbb{F}|^{O(1)}$.

- *Right degree.* On all inputs, the output is right regular, and its right degree is $\mathsf{degright} = \frac{|A|}{|B|} \cdot \mathsf{degleft} = N \cdot |\mathbb{F}|^{O(m)}$. To see why right regularity holds, notice that the distribution induced on B by picking uniformly and independently at random a vertex $a \in A$ and a neighbor of a in the output graph, is uniform: this is the distribution defined on $\mathbb{F}^m$ by picking uniformly and independently at random $i \in [N]$, $\vec{x}, \vec{y}_1, \vec{y}_2 \in \mathbb{F}^m$, $t \in \mathbb{F} \setminus \{q_1, \ldots, q_k\}$, $t_0 \neq 0, t_1, t_2 \in \mathbb{F}$, and computing $t_0 \cdot c_{i,\vec{x}}(t) + t_1 \cdot \vec{y}_1 + t_2 \cdot \vec{y}_2$. By definition of the curves $c_{i,\vec{x}}$, for every $i \in [N]$, for any $t \in \mathbb{F} \setminus \{q_1, \ldots, q_k\}$, for a uniformly distributed $\vec{x} \in \mathbb{F}^m$, we have that $c_{i,\vec{x}}(t)$ is uniform in $\mathbb{F}^m$. Hence, for any $t_0 \neq 0, t_1, t_2 \in \mathbb{F}$ and any $\vec{y}_1, \vec{y}_2 \in \mathbb{F}^m$, the distribution of $t_0 \cdot c_{i,\vec{x}}(t) + t_1 \cdot \vec{y}_1 + t_2 \cdot \vec{y}_2$ is uniform on $\mathbb{F}^m$.

10.2.2 Analysis of The Manifold vs. Point RM-LPR Algorithm. Completing The Proof of Lemma 8.1

To complete the proof of Lemma 8.1 it remains to prove that the algorithm outputs (δ_{min}, l_{max})-RM-LPRs for the δ_{min} and l_{max} stated in the lemma.

Fix an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Denote the output of the algorithm by

$$\mathcal{G} = \langle (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat \doteqdot true, label, proj, tup, eval, pnt, evalp \rangle$$

Let us prove encoding and list decoding:

For a function $f : \mathbb{F}^m \to \mathbb{F}$ and a vertex $a = \langle i, \vec{x}, \vec{y}_1, \vec{y}_2 \rangle \in A$, let the restriction of f to μ_a be $f_{|\mu_a} : \mathbb{F}^w \to \mathbb{F}$ defined by assigning every $(t, t_0, t_1, t_2) \in \mathbb{F}^w$

$$f_{|\mu_a}(t, t_0, t_1, t_2) \doteq f(t_0 \cdot c_{i,\vec{x}}(t) + t_1 \cdot \vec{y}_1 + t_2 \cdot \vec{y}_2)$$

Denote $\mathcal{D} = \langle \mathbb{F}^m, \mathbb{F}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$.

Encoding. Assume a polynomial $f \in \mathcal{D}_{enc}$. For every vertex $b \in B$, take $C_B(b)$ to be the element in $\Sigma_{B,enc}$ corresponding to $f(b)$. Define an assignment $C_A : A \to \Sigma_{A,enc}$ by letting every vertex $a \in A$ be assigned $C_A(a) \doteqdot f_{|\mu_a} \in \Sigma_{A,enc}$. Note that both C_A, C_B can be constructed efficiently given f, and that every edge $e \in E$ is satisfied and reads f in $\mathcal{G}$ under C_A and C_B.

List Decoding. Fix an assignment $C_B : B \to \Sigma_{B,dec}$. Fix a real δ such that $\delta_{min} \leq \delta < 1$.

Invoke the low degree testing theorem given in Theorem 18 (where $\mathbb{K} = \mathbb{F}$) for the function $f_B : \mathbb{F}^m \to \mathbb{F}$ defined for every $\vec{x} \in \mathbb{F}^m$ by letting $f_B(\vec{x})$ be the field element corresponding to $C_B(\vec{x})$. Let $f_1, \ldots, f_l \in \mathcal{D}_{dec}$ be the $l \leq l_{max}(\delta)$ polynomials guaranteed by the theorem.

Fix an assignment $C_A : A \to \Sigma_{A,dec}$.

Proposition 10.0.1. *When picking uniformly and independently at random a vertex $a \in A$ and an edge coming out of it $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B, although $f_B(b) \notin \{f_1(b), \ldots, f_l(b)\}$ is at most $O(\delta)$.*

Proof. There is a probabilistic oracle $\mathcal{A}$, such that picking uniformly at random a vertex $a \in A$ and an edge coming out of it $e = (a, b) \in E$ and checking whether e is satisfied in $\mathcal{G}$ under C_A and C_B is equivalent to performing $LDT^{f_B, \mathcal{A}}$ for the oracle $\mathcal{A}$. To see this, consider $LDT^{f_B, \mathcal{A}}$ when replacing its step 1 by the following procedure (that implicitly defines the oracle $\mathcal{A}$):

- Pick uniformly at random a vertex $a = \langle i, \vec{x}, \vec{y}_1, \vec{y}_2 \rangle \in A$.

- Pick uniformly at random a scalar $t \in \mathbb{F} \setminus \{q_1, \ldots, q_k\}$.

- Output the three vectors $\langle \vec{z} \doteq c_{i,\vec{x}}(t), \vec{y}_1, \vec{y}_2 \rangle$ together with the polynomial $p^*(\mathbf{t_0, t_1, t_2}) \doteq C_A(a)(t, \mathbf{t_0, t_1, t_2})$ of degree at most d'.

For every $t \in \mathbb{F} \setminus \{q_1, \ldots, q_k\}$, for a uniformly distributed $\vec{x} \in \mathbb{F}^m$, the distribution of the point $c_{i,\vec{x}}(t)$ is uniform in $\mathbb{F}^m$. Hence, the distribution of $\langle \vec{z} \doteq c_{i,\vec{x}}(t), \vec{y}_1, \vec{y}_2 \rangle$ is uniform in $\mathbb{F}^m \times \mathbb{F}^m \times \mathbb{F}^m$. The lemma follows from Theorem 18 for $\mathbb{K} = \mathbb{F}$.

$\blacksquare$[of Proposition 10.0.1]

Proposition 10.0.2. *Let $a \in A$ such that $C_A(a) \notin \{f_{1|\mu_a}, \ldots, f_{l|\mu_a}\}$. When picking uniformly at random an edge coming out of a, $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B and $f_B(b) \in \{f_1(b), \ldots, f_l(b)\}$ is at most $O(\delta)$.*

Proof. Write $a = \langle i, \vec{x}, \vec{y}_1, \vec{y}_2 \rangle$. For every $j \in [l]$, the polynomials $C_A(a)$ and $f_{j|\mu_a}$ are *different* w-variate polynomials of degree at most $(k+1) \cdot d'$ over $\mathbb{F}$. Thus, by the Schwartz-Zippel lemma, they can agree on at most a fraction of $\frac{(k+1) \cdot d'}{|\mathbb{F}|}$ of the points in $\mathbb{F}^w$. Hence, $C_A(a)(t, t_0, t_1, t_2) \in \{f_{1|\mu_a}(t, t_0, t_1, t_2), \ldots, f_{l|\mu_a}(t, t_0, t_1, t_2)\}$ for at most a fraction of $\frac{2}{\delta} \cdot \frac{(k+1) \cdot d'}{|\mathbb{F}|}$ of the scalars $t, t_0, t_1, t_2 \in \mathbb{F}$.

By construction, picking uniformly an edge coming out of a is equivalent to picking uniformly and independently at random $t \in \mathbb{F} \setminus \{q_1, \ldots, q_k\}$, $t_0 \neq 0, t_1, t_2 \in \mathbb{F}$ and taking $e = (a, b) \in E$ for $b = t_0 c_{i,\vec{x}}(t) + t_1 \vec{y}_1 + t_2 \vec{y}_2$. Moreover, whenever $e = (a, b)$ is satisfied in $\mathcal{G}$ under C_A, C_B and $f_B(b) \in \{f_1(b), \ldots, f_l(b)\}$, it follows that $C_A(a)(t, t_0, t_1, t_2) = f_B(b) \in \{f_{1|\mu_a}(t, t_0, t_1, t_2), \ldots, f_{l|\mu_a}(t, t_0, t_1, t_2)\}$.

We conclude that, when picking uniformly at random an edge coming out of a, $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B and $f_B(b) \in \{f_1(b), \ldots, f_l(b)\}$ is at most $\frac{2}{\delta} \cdot \frac{(k+1) \cdot d'}{|\mathbb{F}| - k} \cdot \frac{|\mathbb{F}|}{|\mathbb{F}| - 1} = O(\delta)$ (for $\delta \geq \sqrt{\frac{d' \cdot (k+1)}{|\mathbb{F}| - k}}$). $\blacksquare$[of Proposition 10.0.2]

By Proposition 10.0.1 and Proposition 10.0.2 and using left-regularity, when picking uniformly at random an edge $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B, although $C_A(a) \notin \{f_{1|\mu_a}, \ldots, f_{l|\mu_a}\}$ is at most $O(\delta)$. The list decoding property follows noticing that when the edge e is satisfied in $\mathcal{G}$ under C_A and C_B, and $C_A(a) \in \{f_{1|\mu_a}, \ldots, f_{l|\mu_a}\}$, we have that e reads one of $f_1, \ldots, f_l$ in $\mathcal{G}$ under C_A, C_B.

This concludes the proof of Lemma 8.1.

10.3 The Manifold vs. Point RM-LR Construction Algorithm

The purpose of this section is to prove Lemma 8.2. We describe an RM-LR construction algorithm, called the Manifold vs. Point RM-LR construction algorithm. The algorithm is along the same lines as the Manifold vs. Point RM-LPR construction algorithm. The difference between the two comes from savings in the size parameter of the RM-LR algorithm. These savings are obtained by reducing the number of manifolds using ideas from [27, 26].

We will need the following lemma from [26]:

76

Lemma 10.1 (Balancing curves, Lemma 7.1 in [26]). *For a finite field $\mathbb{F}$, a natural number m, natural numbers N and $k < |\mathbb{F}|$, an accuracy parameter $0 < \varepsilon < 1$ and a collection of size N of k-tuples:*

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

there is an algorithm that runs in time polynomial in $|\mathbb{F}^m|$, N and $\frac{1}{\varepsilon}$, and constructs $N \cdot C$ curves $c_{1,1}, \ldots, c_{N,C} : \mathbb{F} \to \mathbb{F}^m$ of degree at most k, where $C \doteq \lceil \frac{|\mathbb{F}^m|}{\varepsilon^2 N} \rceil$ (i.e., there are at most $N + \frac{|\mathbb{F}^m|}{\varepsilon^2}$ curves). The curves have the following properties for fixed (distinct) scalars $q_1, \ldots, q_k \in \mathbb{F}$:

1. *("Curves pass through given points") For every $i \in [N]$, $j \in [C]$, $c_{i,j}(q_1) = \vec{x}_{i,1}, \ldots, c_{i,j}(q_k) = \vec{x}_{i,k}$.*

2. *("Curves cover $\mathbb{F}^m$ almost uniformly") The probability distribution induced on $\mathbb{F}^m$ by picking uniformly and independently at random $i \in [N]$, $j \in [C]$ and $t \in \mathbb{F} \setminus \{q_1, \ldots, q_k\}$, and computing $c_{i,j}(t)$ is ε-close in the l_1-norm to uniform over $\mathbb{F}^m$.*

The presentation of the RM-LR algorithm and its analysis closely follow the presentation of the RM-LPR algorithm.

Construction 2 (Manifold vs. Point RM-LR algorithm). *We use the notation and assume the restrictions appearing in Lemma 8.2.*

Given as input a collection of size N of k-tuples:

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

The construction algorithm constructs an RM-LR

$$\mathcal{G} = \langle (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat \stackrel{.}{\equiv} true, label, proj, tup, eval \rangle$$

for the k-tuples as follows:

A **vertices.** *Set $\varepsilon \doteq \frac{1}{|\mathbb{F}|}$ and $C \doteq \lceil \frac{|\mathbb{F}^m|}{\varepsilon^2 N} \rceil$ as in Lemma 10.1. The vertex set A consists of quadruplets $\langle i, j, \vec{y}_1, \vec{y}_2 \rangle$ for $i \in [N]$ (indicating an input tuple), $j \in [C]$ (indicating a curve through the input tuple) and $\vec{y}_1, \vec{y}_2 \in \mathbb{K}^m$ (needed for the low degree test). We set $V \doteq A$ (for left evaluators) and $\Omega \doteq \mathbb{F}^w$ (recall that we set $w \doteq 4$).*

B **vertices.** *The vertex set B consists of all points $\vec{x} \in \mathbb{F}^m$.*

Alphabets. *The alphabet Σ_A is the domain $\widetilde{\mathcal{D}}$ defined by the finite field $\mathbb{F}$, the dimension w, the encoding degree $(k+1) \cdot d$ and the decoding degree d'. The alphabet Σ_B is the domain associated with the set $\mathbb{F}$. Let us denote $\Sigma_A = \langle \mathbb{F}^w, \mathbb{F}, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$ and $\Sigma_B = \langle \{1\}, \mathbb{F}, \Sigma_{B,enc}, \Sigma_{B,dec} \rangle$.*

Associating A vertices with manifolds. *Invoke the algorithm from Lemma 10.1 on $\mathbb{F}$, m, N, k, ε and the collection $\langle \vec{x}_{1,1}, \dots, \vec{x}_{1,k} \rangle, \dots, \langle \vec{x}_{N,1}, \dots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$. Let $c_{1,1}, \dots, c_{N,C} : \mathbb{F} \to \mathbb{F}^m$ denote the curves outputted by the algorithm.*

A vertex $a = \langle i, j, \vec{y}_1, \vec{y}_2 \rangle \in A$ is associated with the i'th input tuple $tup(a) \doteq \langle \vec{x}_{i,1}, \dots, \vec{x}_{i,k} \rangle$, and corresponds to a manifold of degree at most $k + 1$ through the i'th tuple defined as follows

$$\mu_a(t, t_0, t_1, t_2) = t_0 \cdot c_{i,j}(t) + t_1 \cdot \vec{y}_1 + t_2 \cdot \vec{y}_2$$

where we also denote $\mu_a \doteq \{\mu_a(t, t_0, t_1, t_2) \mid t, t_0, t_1, t_2 \in \mathbb{F}\}$. Note that each $i \in [N]$ has the same number of $a \in A$ with $tup(a) = \langle \vec{x}_{i,1}, \dots, \vec{x}_{i,k} \rangle$.

Edges. *We connect every vertex $a = \langle i, j, \vec{y}_1, \vec{y}_2 \rangle \in A$ to points in B that are on the manifold μ_a. The choice of the points is done as to match the low degree test in Subsection 10.1: for every $t \in \mathbb{F} \setminus \{q_1, \dots, q_k\}$ and $t_0 \neq 0, t_1, t_2 \in \mathbb{F}$, there is an edge $e = (a, b) \in E$ connecting a to the point $b = t_0 \cdot c_{i,j}(t) + t_1 \cdot \vec{y}_1 + t_2 \cdot \vec{y}_2 \in \mathbb{F}^m$ on μ_a. We set $label(e) \doteq (t, t_0, t_1, t_2) \in \Omega$.*

Projection. *For every vertex $a \in A$, assignment $\sigma_a \in \Sigma_{A,dec}$ (which is a w-variate polynomial over the field $\mathbb{F}$) and label $\xi \in \Omega$ (which is a point in $\mathbb{F}^w$), we let $proj(a, \sigma_a, \xi)$ be the element in $\Sigma_{B,enc} = \Sigma_{B,dec}$ corresponding to the field element $\sigma_a(\xi)$.*

Tuple evaluation. *Denote $\vec{p}_1 \doteq (q_1, 1, 0, 0), \dots, \vec{p}_k \doteq (q_k, 1, 0, 0) \in \mathbb{F}^w$. For every vertex $a \in A$, the tuple $tup(a)$ is on the manifold μ_a in positions $\vec{p}_1, \dots, \vec{p}_k$, i.e., $tup(a) = \langle \mu_a(\vec{p}_1), \dots, \mu_a(\vec{p}_k) \rangle$. For every vertex $a \in A$ and assignment $\sigma_a \in \Sigma_{A,dec}$ (which is a polynomial on the manifold μ_a), the evaluation of a on its tuple is given by the evaluation of σ_a on the points $\vec{p}_1, \dots, \vec{p}_k$, i.e., we let $eval(a, \sigma_a) \doteq \langle \sigma_a(\vec{p}_1), \dots, \sigma_a(\vec{p}_k) \rangle$.*

10.3.1 Properties of The Manifold vs. Point RM-LR Construction Algorithm

The algorithm is efficient, and runs in time polynomial in $|\mathbb{F}^m|$ and N.

- *Size.* On all inputs, the output is of size $size = N \cdot C \cdot |\mathbb{K}^m|^2 + |\mathbb{F}^m| + N \cdot C \cdot |\mathbb{K}^m|^2 \cdot |\mathbb{F}|^{O(1)} = (N + |\mathbb{F}^m|) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m}$.

- *Block length.* On all inputs, the output has block length $block = \mathsf{poly}(k, d) \cdot \log |\mathbb{F}|$.

- *Left degree.* On all inputs, the output is left regular with left degree $degleft \leq |\mathbb{F}|^{O(1)}$.

- *Right degree.* The output is not necessarily right regular. We bound $degright \leq (N + |\mathbb{F}^m|) \cdot |\mathbb{F}|^{O(1)} \cdot |\mathbb{K}|^{2m}$.

10.3.2 Analysis of The Manifold vs. Point RM-LR Algorithm. Completing The Proof of Lemma 8.2

To complete the proof of Lemma 8.2 it remains to prove that the algorithm outputs (δ_{min}, l_{max})-RM-LRs for the δ_{min} and l_{max} stated in the lemma.

Fix an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Denote the output of the algorithm by

$$\mathcal{G} = \langle (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat \doteq true, label, proj, tup, eval \rangle$$

Let us prove encoding and list decoding:

For a function $f : \mathbb{F}^m \to \mathbb{F}$ and a vertex $a = \langle i, j, \vec{y}_1, \vec{y}_2 \rangle \in A$, let the restriction of f to μ_a be $f_{|\mu_a} : \mathbb{F}^w \to \mathbb{F}$ defined by assigning every $(t, t_0, t_1, t_2) \in \mathbb{F}^w$

$$f_{|\mu_a}(t, t_0, t_1, t_2) \doteq f(t_0 \cdot c_{i,j}(t) + t_1 \cdot \vec{y}_1 + t_2 \cdot \vec{y}_2)$$

Denote $\mathcal{D} = \langle \mathbb{F}^m, \mathbb{F}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$.

Encoding. Assume a polynomial $f \in \mathcal{D}_{enc}$. For every vertex $b \in B$, take $C_B(b)$ to be the element in $\Sigma_{B,enc}$ corresponding to $f(b)$. Define an assignment $C_A : A \to \Sigma_{A,enc}$ by letting every vertex $a \in A$ be assigned $C_A(a) \doteq f_{|\mu_a} \in \Sigma_{A,enc}$. Note that both C_A, C_B can be constructed efficiently given f, and that every edge $e \in E$ is satisfied and reads f in $\mathcal{G}$ under C_A and C_B.

List Decoding. Fix an assignment $C_B : B \to \Sigma_{B,dec}$. Fix a real δ such that $\delta_{min} \le \delta < 1$.

Invoke the low degree testing theorem given in Theorem 18 for the function $f_B : \mathbb{F}^m \to \mathbb{F}$ defined for every $\vec{x} \in \mathbb{F}^m$ by letting $f_B(\vec{x})$ be the field element corresponding to $C_B(\vec{x})$. Let $f_1, \ldots, f_l \in \mathcal{D}_{dec}$ be the $l \le l_{max}(\delta)$ polynomials guaranteed by the theorem.

Fix an assignment $C_A : A \to \Sigma_{A,dec}$.

Proposition 10.1.1. *When picking uniformly and independently at random a vertex $a \in A$ and an edge coming out of it $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B, although $f_B(b) \notin \{f_1(b), \ldots, f_l(b)\}$ is at most $O(\delta)$.*

Proof. Consider $LDT^{f_B, \mathcal{A}}$ when replacing its step 1 by the following procedure (that implicitly defines the oracle $\mathcal{A}$):

- Pick uniformly at random a vertex $a = \langle i, j, \vec{y}_1, \vec{y}_2 \rangle \in A$.

- Pick uniformly at random a scalar $t \in \mathbb{F} \setminus \{q_1, \ldots, q_k\}$.

- Output the three vectors $\langle \vec{z} \doteq c_{i,j}(t), \vec{y}_1, \vec{y}_2 \rangle$ together with the polynomial $p^*(\mathbf{t_0}, \mathbf{t_1}, \mathbf{t_2}) \doteq C_A(a)(t, \mathbf{t_0}, \mathbf{t_1}, \mathbf{t_2})$ of degree at most d'.

By the properties of the curves $c_{i,j}$, the distribution of the point $c_{i,j}(t)$ is ε-close (in the l_1-norm) to uniform on $\mathbb{F}^m$. Hence, the distribution of $\langle \vec{z} \doteq c_{i,j}(t), \vec{y}_1, \vec{y}_2 \rangle$ is ε-close (in the l_1-norm) to uniform on $\mathbb{F}^m \times \mathbb{K}^m \times \mathbb{K}^m$. The lemma follows from Theorem 18 noticing that $\varepsilon \leq \delta$.

$\blacksquare$[of Proposition 10.1.1]

Similarly to Proposition 10.0.2, relying on the low degree of the curves, we have the following:

Proposition 10.1.2. *Let $a \in A$ such that $C_A(a) \notin \{f_{1|\mu_a}, \ldots, f_{l|\mu_a}\}$. When picking uniformly at random an edge coming out of a, $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B and $f_B(b) \in \{f_1(b), \ldots, f_l(b)\}$ is at most $O(\delta)$.*

By Proposition 10.1.1 and Proposition 10.1.2 and using left-regularity, when picking uniformly at random an edge $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B, although $C_A(a) \notin \{f_{1|\mu_a}, \ldots, f_{l|\mu_a}\}$ is at most $O(\delta)$. The list decoding property follows noticing that when $C_A(a) \in \{f_{1|\mu_a}, \ldots, f_{l|\mu_a}\}$, we have that e reads one of $f_1, \ldots, f_l$ in $\mathcal{G}$ under C_A, C_B.

This concludes the proof of Lemma 8.2.

11 Construction of Hadamard Left Readers

In this section we present an algorithm for constructing Had-LRs proving Lemma 8.3. The idea of the construction algorithm is to create a bipartite graph, in which side A corresponds to low dimensional linear subspaces in $\mathbb{F}^m$, each passing through an input tuple, and side B consists of points in $\mathbb{F}^m$. This way each A vertex has a tuple that is associated with it. We put edges between A vertices corresponding to subspaces and B vertices corresponding to points on them. Assignments to A vertices naturally project onto assignments to their neighboring B vertices.

A correct encoding of a linear function $L : \mathbb{F}^m \to \mathbb{F}$ assigns L to the points in B, and assigns the restrictions of L to the subspaces to the A vertices. This way each A vertex can evaluate L on the tuple associated with it. Note that the restriction of L to any linear subspace is a linear function.

To show that the construction has a list decoding property, we use a *linearity testing* theorem for large finite fields. The linearity test we need is in the form of a projection test. Its analysis follows from the analysis of the more standard Blum-Luby-Rubinfeld test [11]. Specifically, we build upon the analysis for large finite fields by [19]. The analysis requires that the field $\mathbb{F}$ is prime and assumes that the function is *folded* (details follow).

11.1 A Linearity Testing Theorem

Folding. Let $\mathbb{F}$ be a finite field. Let m be a natural number. Fix a function $f : \mathbb{F}^m \to \mathbb{F}$ we wish to test. If f were a linear function, then:

1. *(Multiplication by scalar)* For every $\vec{z} \in \mathbb{F}^m$ and $t \in \mathbb{F}$, $f(t \cdot \vec{z}) = t \cdot f(\vec{z})$.

2. *(Addition)* For every $\vec{z}, \vec{y} \in \mathbb{F}^m$, $f(\vec{z} + \vec{y}) = f(\vec{z}) + f(\vec{y})$.

We can ensure that item 1 holds by using *folding*: consider the equivalence relation $\sim$ on $\mathbb{F}^m$, where $\vec{z} \sim \vec{y}$ if and only if $\vec{z} = t \cdot \vec{y}$ for some $t \neq 0 \in \mathbb{F}$. Let $R \subseteq \mathbb{F}^m$ be a set of representatives of the equivalence classes. For every $\vec{x} \in \mathbb{F}^m$, let $[\vec{x}]$ be the representative of the class that contains $\vec{x}$. A *folded* function is a function that is defined on the representatives $\widetilde{f} : R \to \mathbb{F}$. A folded function $\widetilde{f}$ defines a function $f : \mathbb{F}^m \to \mathbb{F}$ that satisfies item 1 by assigning every $\vec{x} \in \mathbb{F}^m$ the appropriate multiple of the value of its representative, i.e., if $\vec{x} = t \cdot [\vec{x}]$, then $f(\vec{x}) \doteq t \cdot \widetilde{f}([\vec{x}])$.

Linearity Testing (projection form). Let $\widetilde{f} : R \to \mathbb{F}$. Assume access to an oracle $\mathcal{A}$ whose goal is to convince us that f is linear. For every $\vec{z} \in \mathbb{F}^m$ and $\vec{y} \in \mathbb{F}^m$, the oracle $\mathcal{A}$ provides a bi-variate linear function, which is supposedly $f(\mathbf{t_1}\vec{z} + \mathbf{t_2}\vec{y})$. The oracle $\mathcal{A}$ may be probabilistic, meaning that its answer may depend not only on $\vec{z}$ and $\vec{y}$, but also on additional randomness.

A test for checking that f is consistent with a linear function is described in Figure 9.

$LinTest^{\widetilde{f}, \mathcal{A}}$:

1. Pick uniformly at random two vectors $\langle \vec{z}, \vec{y} \rangle \in \mathbb{F}^m \times \mathbb{F}^m$. Using the oracle access to $\mathcal{A}$, obtain a bi-variate linear function $l^*(\mathbf{t_1}, \mathbf{t_2})$ over $\mathbb{F}$ for $\langle \vec{z}, \vec{y} \rangle$ *[l^* is supposedly the restriction $f(\mathbf{t_1}\vec{z} + \mathbf{t_2}\vec{y})$].*

2. Pick uniformly at random $t_1, t_2 \in \mathbb{F}$. Set $\vec{x}_0 = t_1\vec{z} + t_2\vec{y}$. If indeed $l^*(t_1, t_2) = f(\vec{x}_0)$, *accept*. Otherwise, *reject*.

Figure 9: Linearity Tester (Projection form)

The following follows from the analysis of [19] (see Appendix C for details):

Theorem 19 (Analysis of linearity test, [19]). *There are some natural m_0 and F_0, such that for every $m \geq m_0$ and a prime finite field $\mathbb{F}$ with $|\mathbb{F}| \geq F_0$, the following holds. Let $R \subseteq \mathbb{F}^m$ be a set of representatives needed for folding.*

For $\delta \geq 2\sqrt[6]{\frac{1}{|\mathbb{F}|}}$, for any function $\widetilde{f} : R \to \mathbb{F}$, there are $l \leq \frac{2}{\delta^3}$ linear functions $L_1, \ldots, L_l :$ $\mathbb{F}^m \to \mathbb{F}$, such that for every probabilistic oracle $\mathcal{A}$:

The probability, over the randomness of $\mathcal{A}$ and over the randomness of the tester, that $LinTest^{\widetilde{f}, \mathcal{A}}$ accepts, although $f(\vec{x}_0) \notin \{L_1(\vec{x}_0), \ldots, L_l(\vec{x}_0)\}$ (where $f : \mathbb{F}^m \to \mathbb{F}$ is the function defined by $\widetilde{f}$ and $\vec{x}_0 \in \mathbb{F}^m$ is picked by the tester; see Figure 9), is at most $O(\delta)$.

11.2 The Had-LR Construction Algorithm

The purpose of this section is to prove Lemma 8.3. We describe a Had-LR construction algorithm that is uniform in the tuple association and in the encoding and list decoding. We specify the properties of the algorithm in Subsection 11.2.1 and analyze it in Subsection 11.2.2.

The description of the algorithm will start with specifying the uniform part of the construction (i.e., the part that is common to all outputs of the algorithm), and proceed by presenting the components that are input-specific.

Construction 3 (Had-LR construction algorithm). *We use the notation and assume the restrictions appearing in Lemma 8.3.*

We define a uniform structure and uniform tuple association as follows:

A **vertices.** *The vertex set A consists of triples $\langle i, \vec{z}, \vec{y} \rangle$ for $i \in [N]$ (indicating an input tuple) and $\vec{z}, \vec{y} \in \mathbb{F}^m$ (needed for the linearity test). We set $V \doteq A$ (for left evaluators). The uniform tuple associator $\mathsf{tupi} : A \to [N]$ assignes $\langle i, \vec{z}, \vec{y} \rangle$ the index i.*

Set $w \doteq k + 2$ and $\Omega \doteq \mathbb{F}^w$.

B **vertices.** *The vertex set B consists of all representatives $\vec{x} \in R$.*

Alphabets. *The alphabet Σ_A is the Hadamard domain $\widetilde{\mathcal{D}}$ defined by the finite field $\mathbb{F}$ and the dimension w. The alphabet Σ_B is the domain associated with the set $\mathbb{F}$. Let us denote $\Sigma_A = \langle \mathbb{F}^w, \mathbb{F}, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$ and $\Sigma_B = \langle \{1\}, \mathbb{F}, \Sigma_{B,enc}, \Sigma_{B,dec} \rangle$.*

Given as input a collection of size N of k-tuples:

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

The construction algorithm constructs a Had-LR

$$\mathcal{G} = \langle (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat \stackrel{.}{\equiv} true, label, proj, tup, eval \rangle$$

for the k-tuples as follows:

Associating A vertices with manifolds. *A vertex $a = \langle i, \vec{z}, \vec{y} \rangle \in A$ is associated with the i'th input tuple $tup(a) \doteq \langle \vec{x}_{i,1}, \ldots, \vec{x}_{i,k} \rangle$. Let us assume without loss of generality that $\vec{x}_{i,1}, \ldots, \vec{x}_{i,k}$ are linearly independent (otherwise, we find a maximal sub-tuple of linearly independent vectors inside the k-tuple and complete it to a k-tuple $\langle \vec{x}_{i,1}, \ldots, \vec{x}_{i,k} \rangle$ of linearly independent vectors).*

Denote $\vec{x}_{i,k+1} = \vec{z}$ and $\vec{x}_{i,k+2} = \vec{y}$. Then, the vertex a corresponds to the following linear subspace through the i'th tuple:

$$s_a \doteq \left\{ \sum_{j=1}^{k+2} t_j \cdot \vec{x}_{i,j} \ \middle| \ t_1, \ldots, t_{k+2} \in \mathbb{F} \right\}$$

We also use the functional notation $l_a : \mathbb{F}^w \to \mathbb{F}^m$, where for every $\vec{t} = (t_1, \ldots, t_{k+2}) \in \mathbb{F}^w$,

$$l_a(\vec{t}) = \sum_{j=1}^{k+2} t_j \cdot \vec{x}_{i,j}$$

Note that each $i \in [N]$ has the same number of $a \in A$ with $tup(a) = \langle \vec{x}_{i,1}, \ldots, \vec{x}_{i,k} \rangle$.

Edges. *We connect every vertex $a = \langle i, \vec{z}, \vec{y} \rangle \in A$ to representatives in B of points on the subspace s_a. For every $\vec{t} = (t_1, \ldots, t_{k+2}) \in \mathbb{F}^{k+2}$ such that $t_{k+1} \neq 0$, there is an edge $e = (a, b) \in E$ connecting a to the representative of the corresponding point on s_a, $b = [l_a(\vec{t})]$. We set $label(e) \doteq \vec{t} \in \Omega$.*

Projection. *For every vertex $a \in A$, assignment $\sigma_a \in \Sigma_{A,dec}$ (which is a w-variate linear function over the field $\mathbb{F}$) and label $\xi \in \Omega$ (which is a point in $\mathbb{F}^w$), such that $l_a(\xi) = t \cdot [l_a(\xi)]$ for a scalar $t \neq 0 \in \mathbb{F}$, we let $proj(a, \sigma_a, \xi)$ be the element in $\Sigma_{B,enc} = \Sigma_{B,dec}$ corresponding to the field element $\frac{1}{t} \cdot \sigma_a(\xi)$.*

Tuple evaluation. *For every vertex $a \in A$, we let $\vec{p}_1, \ldots, \vec{p}_k \in \mathbb{F}^w$ be such that the k-tuple $tup(a)$ is on the subspace s_a in positions $\vec{p}_1, \ldots, \vec{p}_k$, i.e., $tup(a) = \langle l_a(\vec{p}_1), \ldots, l_a(\vec{p}_k) \rangle$. For example, when the original k-tuple $\langle \vec{x}_{i,1}, \ldots, \vec{x}_{i,k} \rangle$ consists of linearly independent vectors, we have $\vec{p}_1 \doteq (1, 0, \ldots, 0, 0, 0), \ldots, \vec{p}_k \doteq (0, \ldots, 1, 0, 0) \in \mathbb{F}^w$.*

For every vertex $a \in A$ and assignment $\sigma_a \in \Sigma_{A,dec}$ (which is a w-variate linear function over the field $\mathbb{F}$), the evaluation of a on its tuple is given by the evaluation of σ_a on the points $\vec{p}_1, \ldots, \vec{p}_k$, i.e., we let $eval(a, \sigma_a) \doteq \langle \sigma_a(\vec{p}_1), \ldots, \sigma_a(\vec{p}_k) \rangle$.

11.2.1 Properties of The Had-LR Construction Algorithm

The algorithm is efficient, and runs in time polynomial in $|\mathbb{F}^m|$ and N.

- *Size.* On all inputs, the output is of size $size \leq N \cdot |\mathbb{F}|^{2m} + |R| + N \cdot |\mathbb{F}|^{2m} \cdot |\mathbb{F}|^{k+2} = N \cdot |\mathbb{F}|^{O(m)}$.

- *Block length.* On all inputs, the output has block length $block = \log |\Sigma_{A,enc}| = \log |\mathbb{F}|^{k+2} = O(k) \cdot \log |\mathbb{F}|$.

- *Left degree.* On all inputs, the output is left regular with left degree $degleft = (1 - \frac{1}{|\mathbb{F}|}) \cdot |\mathbb{F}^w| \leq |\mathbb{F}|^{O(k)}$.

- *Right degree.* The graph is not right regular, however, the degrees of the B vertices are the same for all inputs, and $degright \leq size$. To see why the right degrees are independent of the input, note that for every fixing of $\vec{x}_{i,1}, \ldots, \vec{x}_{i,k}$ and $\vec{x}_{i,k+2} = \vec{y}$, for any $\vec{t} = (t_1, \ldots, t_{k+2}) \in \mathbb{F}^{k+2}$ such that $t_{k+1} \neq 0 \in \mathbb{F}$, when $\vec{x_{i,k+1}} = \vec{z}$ is uniformly distributed in $\mathbb{F}^m$, so is $\sum_{j=1}^{k+2} t_j \cdot$

83

$\vec{x}_{i,j}$. The vectors $\vec{y}$ and $\vec{z}$ are distributed uniformly in $\mathbb{F}^m$, independent of $\vec{x}_{i,1}, \dots, \vec{x}_{i,k}$. Thus, the degree of the $\frac{|\mathbb{F}^m|-1}{|\mathbb{F}|-1}$ representatives that are not $\vec{0}$ are the same, and all the right degrees are independent of the input.

11.2.2 Analysis of The Had-LR Algorithm. Completing The Proof of Lemma 8.3

To complete the proof of Lemma 8.3 it remains to prove that the algorithm is uniform in the encoding and list decoding and outputs (δ_{min}, l_{max})-Had-LRs for the δ_{min} and l_{max} stated in the lemma.

Fix an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \dots, \vec{x}_{1,k} \rangle, \dots, \langle \vec{x}_{N,1}, \dots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Denote the output of the algorithm by

$$\mathcal{G} = \langle (A, B, E), V, \Omega, \Sigma_A, \Sigma_B, sat \equiv true, label, proj, tup, eval \rangle$$

Let us prove encoding and list decoding:

For a function $f : \mathbb{F}^m \to \mathbb{F}$ and a vertex $a = \langle i, \vec{z}, \vec{y} \rangle \in A$, let the restriction of f to s_a be $f_{|a} : \mathbb{F}^w \to \mathbb{F}$ defined by assigning every $\vec{t} \in \mathbb{F}^w$

$$f_{|a}(\vec{t}) \doteq f(l_a(\vec{t}))$$

Denote $\mathcal{D} = \langle \mathbb{F}^m, \mathbb{F}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$.

Encoding.   Let us prove uniform encoding. Assume a linear function $f \in \mathcal{D}_{enc}$. For every vertex $b \in B$, take $C_B(b)$ to be the element in $\Sigma_{B,enc}$ corresponding to $f(b)$. Define an assignment $C_A : A \to \Sigma_{A,enc}$ by letting every vertex $a \in A$ be assigned $C_A(a) \equiv f_{|a} \in \Sigma_{A,enc}$. Note that both C_A, C_B can be constructed efficiently given f, and that every edge $e \in E$ is satisfied and reads f in $\mathcal{G}$ under C_A and C_B.

List Decoding.   Fix an assignment $C_B : B \to \Sigma_{B,dec}$. Fix a real δ such that $\delta_{min} \leq \delta < 1$.

Invoke the linearity testing theorem given in Theorem 19 for the function $\widetilde{f}_B : R \to \mathbb{F}$ defined for every $\vec{x} \in R$ by letting $\widetilde{f}_B(\vec{x})$ be the field element corresponding to $C_B(\vec{x})$. Let $f_1, \dots, f_l \in \mathcal{D}_{dec}$ be the $l \leq l_{max}(\delta)$ linear functions guaranteed by the theorem. Note that the construction of $f_1, \dots, f_l$ is uniform: it requires only C_B and δ.

Fix an assignment $C_A : A \to \Sigma_{A,dec}$.

Proposition 11.0.3. *When picking uniformly and independently at random a vertex $a \in A$ and an edge coming out of it $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B, although $\widetilde{f}_B(b) \notin \{f_1(b), \dots, f_l(b)\}$ is at most $O(\delta)$.*

Proof. There is a probabilistic oracle $\mathcal{A}$, such that picking uniformly at random a vertex $a \in A$ and an edge coming out of it $e = (a, b) \in E$ and checking whether e is satisfied in $\mathcal{G}$ under C_A and C_B is equivalent (up to a small statistical distance) to performing $LinTest^{\tilde{f}_B, \mathcal{A}}$ for the oracle $\mathcal{A}$. To see this, consider $LinTest^{\tilde{f}_B, \mathcal{A}}$ when replacing its step 1 by the following procedure (that implicitly defines the oracle $\mathcal{A}$):

- Pick uniformly at random a vertex $a = \langle i, \vec{z}, \vec{y} \rangle \in A$. Denote $tup(a) = \langle \vec{x}_{i,1}, \ldots, \vec{x}_{i,k} \rangle$.

- Pick uniformly (and independently of the choice of a) at random $\vec{t} = (t_1, \ldots, t_k) \in \mathbb{F}^k$.

- Output the two vectors $\langle \vec{z} + \sum_{j=1}^{k} t_j \cdot \vec{x}_{i,j}, \vec{y} \rangle$ together with the bi-variate linear function $l^*(\mathbf{t_{k+1}}, \mathbf{t_{k+2}}) \doteq C_A(a)(\mathbf{t_{k+1}} \cdot t_1, \ldots, \mathbf{t_{k+1}} \cdot t_k, \mathbf{t_{k+1}}, \mathbf{t_{k+2}})$.

Note that $\vec{z} + \sum_{j=1}^{k} t_j \cdot \vec{x}_{i,j}$ is uniformly distributed in $\mathbb{F}^m$. Moreover, the follwoing two distributions are at statistical distance $O(\frac{1}{|\mathbb{F}|})$:

1. $LinTest^{\tilde{f}, \mathcal{A}}$ distribution: Pick uniformly at random $t_{k+1}, t_{k+2} \in \mathbb{F}$. Compute

$$
t_{k+1} \cdot \left(\vec{z} + \sum_{j=1}^{k} t_j \cdot \vec{x}_{i,j} \right) + t_{k+2} \cdot \vec{y}
$$

2. Edge distribution: Pick uniformly at random $t'_1, \ldots, t'_{k+2} \in \mathbb{F}$ such that $t'_{k+1} \neq 0$. Compute

$$
\sum_{j=1}^{k} t'_j \cdot \vec{x}_{i,j} + t'_{k+1} \cdot \vec{z} + t'_{k+2} \cdot \vec{y}
$$

The proposition follows from Theorem 19, recalling that $\delta \geq \frac{1}{|\mathbb{F}|}$.

$\blacksquare$[of Proposition 11.0.3]

Proposition 11.0.4. *Let $a \in A$ such that $C_A(a) \notin \{f_{1|a}, \ldots, f_{l|a}\}$. When picking uniformly at random an edge coming out of a, $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B and $\tilde{f}_B(b) \in \{f_1(b), \ldots, f_l(b)\}$ is at most $O(\delta)$.*

Proof. Write $a = \langle i, \vec{z}, \vec{y} \rangle$. For every $j \in [l]$, $C_A(a)$ and $f_{j|a}$ are *different* w-variate linear functions over $\mathbb{F}$. Thus, they can agree on at most a fraction of $\frac{1}{|\mathbb{F}|}$ of the points in $\mathbb{F}^w$. Hence, $C_A(a)(\vec{t}) \in \{f_{1|a}(\vec{t}), \ldots, f_{l|a}(\vec{t})\}$ for at most a fraction of $\frac{2}{\delta^3} \cdot \frac{1}{|\mathbb{F}|}$ of the $\vec{t} \in \mathbb{F}^w$.

By construction, picking uniformly an edge coming out of a is equivalent to picking uniformly $\vec{t} \in \mathbb{F}^w$ such that $t_{k+1} \neq 0$, and taking $e = (a, b) \in E$ for $b = [l_a(\vec{t})]$ (where recall that we denote $\vec{x}_{i,k+1} = \vec{z}$ and $\vec{x}_{i,k+2} = \vec{y}$). Moreover, whenever $e = (a, b)$ is satisfied in $\mathcal{G}$ under C_A, C_B and $\tilde{f}_B(b) \in \{f_1(b), \ldots, f_l(b)\}$, it follows that $C_A(a)(\vec{t}) \in \{f_{1|a}(\vec{t}), \ldots, f_{l|a}(\vec{t})\}$.

We conclude that, when picking uniformly at random an edge coming out of a, $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B and $\widetilde{f}_B(b) \in \{f_1(b), \ldots, f_l(b)\}$ is at most $\frac{2}{\delta^3} \cdot \frac{1}{|\mathbb{F}|} = O(\delta)$.   ■[of Proposition 11.0.4]

By Proposition 11.0.3 and Proposition 11.0.4 and using left-regularity, when picking uniformly at random an edge $e = (a, b) \in E$, the probability that e is satisfied in $\mathcal{G}$ under C_A, C_B, although $C_A(a) \notin \{f_{1|a}, \ldots, f_{l|a}\}$ is at most $O(\delta)$. The list decoding property follows noticing that when the edge e is satisfied in $\mathcal{G}$ under C_A and C_B, and $C_A(a) \in \{f_{1|a}, \ldots, f_{l|a}\}$, we have that e reads one of $f_1, \ldots, f_l$ in $\mathcal{G}$ under C_A, C_B.

This concludes the proof of Lemma 8.3.

12   Power Reduction

In this section we show the power reduction manipulation, proving Lemma 8.4. Our presentation will use the notation introduced in this lemma.

The power reduction manipulation transforms construction algorithms reducing $\mathcal{D} \mapsto \mathcal{D}_1$, where $\mathcal{D}_1$ is a Reed-Muller domain with a small dimension m_1 but a large encoding degree d_1, to construction algorithms reducing $\mathcal{D} \mapsto \mathcal{D}_2$, where $\mathcal{D}_2$ is a Reed-Muller domain in which both the dimension m_2 and the encoding degree d_2 are logarithmic in the encoding degree d_1. The manipulation is based on the *power substitution* technique from [13].

12.1   A Power Reducing Embedding

The manipulation is done using the following embedding:

Embedding $\mathbb{F}^{m_1} \hookrightarrow \mathbb{F}^{m_2}$.   Recall that $b_1 = \lceil \log(d_1 + 1) \rceil$ and $m_2 = m_1 \cdot b_1$. Define an embedding $\phi : \mathbb{F}^{m_1} \to \mathbb{F}^{m_2}$ by mapping every $(x_1, \ldots, x_{m_1}) \in \mathbb{F}^{m_1}$ to

$$\phi(x_1, \ldots, x_{m_1}) \doteq (x_1^{2^0}, x_1^{2^1}, \ldots, x_1^{2^{b_1-1}}, \cdots, x_{m_1}^{2^0}, x_{m_1}^{2^1} \ldots, x_{m_1}^{2^{b_1-1}})$$

Power reduction for polynomials via the embedding.   Assume that we have a polynomial $Q : \mathbb{F}^{m_1} \to \mathbb{F}$ of degree at most d_1. For some coefficients $\left\{ \alpha_{i_1, \ldots, i_{m_1}} \right\}_{i_1, \ldots, i_{m_1}}$ where $\alpha_{i_1, \ldots, i_{m_1}} \in \mathbb{F}$, the polynomial Q can be written as

$$Q(x_1, \ldots, x_{m_1}) = \sum_{i_1, \ldots, i_{m_1}} \alpha_{i_1, \ldots, i_{m_1}} x_1^{i_1} \cdots x_{m_1}^{i_{m_1}}$$

For a number $0 \leq i \leq d_1$, denote by $b(i, b_1 - 1) \cdots b(i, 0)$ the binary representation of i. Then, we define a polynomial $Q_\phi : \mathbb{F}^{m_2} \to \mathbb{F}$ by mapping every $(x_{1,0}, \ldots, x_{1,b_1-1}, \cdots, x_{m_1,0}, \ldots, x_{m_1,b_1-1}) \in \mathbb{F}^{m_2}$ to

$$Q_\phi(x_{1,0}, \ldots, x_{1,b_1-1}, \cdots, x_{m_1,0}, \ldots, x_{m_1,b_1-1}) \doteq$$

$$\sum_{i_1,\dots,i_{m_1}} \alpha_{i_1,\dots,i_{m_1}} \cdot x_{1,0}^{b(i_1,0)} \cdots x_{1,b_1-1}^{b(i_1,b_1-1)} \cdots\cdots x_{m_1,0}^{b(i_{m_1},0)} \cdots x_{m_1,b_1-1}^{b(i_{m_1},b_1-1)}$$

Note that Q_ϕ is efficiently computable and that $Q \equiv Q_\phi \circ \phi$. The polynomial Q_ϕ is a multi-linear polynomial in m_2 variables, and hence $Q_\phi \in \mathcal{D}_{2,enc}$. Moreover, since $d_2' = \lfloor d_1'/d_1 \rfloor$, for every $\widetilde{Q} \in \mathcal{D}_{2,dec}$, it holds that $\widetilde{Q} \circ \phi \in \mathcal{D}_{1,dec}$.

12.2   The Power Reduction Manipulation on RM-LRs

Given an RM-LR reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for some tuples:

$$\mathcal{G} = \langle (A, B, E), V = A, \Omega, \mathcal{D}_1, \widetilde{\mathbb{F}}, sat_\mathcal{G} \doteq true, label_\mathcal{G}, proj_\mathcal{G}, tup_\mathcal{G}, eval_\mathcal{G} \rangle$$

where $\widetilde{\mathbb{F}} = \langle \{1\}, \mathbb{F}, \widetilde{\mathbb{F}}_{enc}, \widetilde{\mathbb{F}}_{dec} \rangle$ is the domain associated with the finite field $\mathbb{F}$, we construct an RM-LR reducing $\mathcal{D} \mapsto \mathcal{D}_2$ for the same tuples (note that $\overline{\mathcal{G}}$ has the same tup function as $\mathcal{G}$):

$$\overline{\mathcal{G}} = \langle (A, B, E), V = A, \overline{\Omega}, \mathcal{D}_2, \widetilde{\mathbb{F}}, sat_{\overline{\mathcal{G}}} \doteq true, label_{\overline{\mathcal{G}}}, proj_{\overline{\mathcal{G}}}, tup_\mathcal{G}, eval_{\overline{\mathcal{G}}} \rangle$$

by performing the following operations:

1. Let $\overline{\Omega} \doteq \mathbb{F}^{m_2}$.

2. For every edge $e \in E$, set $label_{\overline{\mathcal{G}}}(e) \doteq \phi(label_\mathcal{G}(e))$.

3. For a vertex $a \in A$, an assignment for it $\sigma_{a,2} \in \mathcal{D}_{2,dec}$ and a label $\vec{p} \in \mathbb{F}^{m_2}$, we let $proj_{\overline{\mathcal{G}}}(a, \sigma_{a,2}, \vec{p})$ be the element in $\widetilde{\mathbb{F}}_{dec}$ corresponding to the field element $\sigma_{a,2}(\vec{p})$. Note that for every edge $e = (a, b) \in E$ it holds that $proj_{\overline{\mathcal{G}}}(a, \sigma_{a,2}, label_{\overline{\mathcal{G}}}(e)) = proj_\mathcal{G}(a, \sigma_{a,2} \circ \phi, label_\mathcal{G}(e))$.

4. Assume that the points $\vec{p}_1, \dots, \vec{p}_k \in \mathbb{F}^{m_1}$ are such that for every vertex $a \in A$ and assignment $\sigma_{a,1} \in \mathcal{D}_{1,dec}$ we have that $eval_\mathcal{G}(a, \sigma_{a,1}) = \langle \sigma_{a,1}(\vec{p}_1), \dots, \sigma_{a,1}(\vec{p}_k) \rangle$. Then, for every vertex $a \in A$ and assignment $\sigma_{a,2} \in \mathcal{D}_{2,dec}$, let $eval_{\overline{\mathcal{G}}}(a, \sigma_{a,2}) \doteq \langle \sigma_{a,2}(\phi(\vec{p}_1)), \dots, \sigma_{a,2}(\phi(\vec{p}_k)) \rangle = eval_\mathcal{G}(a, \sigma_{a,2} \circ \phi)$.

Properties of the manipulation. $\overline{\mathcal{G}}$ can be computed efficiently from $\mathcal{G}$. Moreover, $\overline{\mathcal{G}}$ has the same size and left and right degrees as $\mathcal{G}$, and its block length is $d_1^{O(m_1)} \cdot \log |\mathbb{F}|$.

Analysis.   We say that assignments $\overline{C}_A : A \to \mathcal{D}_{2,dec}$ and $C_A : A \to \mathcal{D}_{1,dec}$ are *equivalent*, if for every vertex $a \in A$, it holds that $C_A(a) = \overline{C}_A(a) \circ \phi$.

Proposition 12.1. *For any two equivalent assignments $\overline{C}_A : A \to \mathcal{D}_{2,dec}$ and $C_A : A \to \mathcal{D}_{1,dec}$, for any assignment $C_B : B \to \widetilde{\mathbb{F}}_{dec}$, for any edge $e = (a, b) \in E$, we have:*

- *e is satisfied in $\mathcal{G}$ under C_A, C_B if and only if e is satisfied in $\overline{\mathcal{G}}$ under $\overline{C}_A, C_B$.*

- *e reads some $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under C_A, C_B if and only if e reads f in $\overline{\mathcal{G}}$ under $\overline{C}_A, C_B$.*

We thus have:

Lemma 12.2. *Let $0 < \delta_{min} < 1$. Let $l_{max} : (0,1) \to \mathbb{R}^+$ be a decreasing function. If $\mathcal{G}$ is a (δ_{min}, l_{max})-RM-LR reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for some tuples, and $\overline{\mathcal{G}}$ is obtained from $\mathcal{G}$ by the power reduction manipulation, then $\overline{\mathcal{G}}$ is a (δ_{min}, l_{max})-RM-LR reducing $\mathcal{D} \mapsto \mathcal{D}_2$ for the same tuples.*

Proof. Let us prove encoding and list decoding:

Encoding. For a polynomial $f \in \mathcal{D}_{enc}$, let $C_A : A \to \mathcal{D}_{1,enc}$ and $C_B : B \to \widetilde{\mathbb{F}}_{enc}$ be the efficiently computable assignments under which every edge $e \in E$ is satisfied and reads f in $\mathcal{G}$. The assignment $\overline{C}_A : A \to \mathcal{D}_{2,enc}$ equivalent to C_A, defined by assigning every $a \in A$ the polynomial $C_A(a)_\phi$, is also efficiently computable. Moreover, by Proposition 12.1, all edges $e \in E$ are satisfied and read f in $\overline{\mathcal{G}}$ under $\overline{C}_A, C_B$.

List Decoding. Fix an assignment $C_B : B \to \widetilde{\mathbb{F}}_{dec}$. Fix a real δ such that $\delta_{min} \le \delta < 1$. Let $f_1, \ldots, f_l \in \mathcal{D}_{dec}$ be the $l \le l_{max}(\delta)$ elements guaranteed by the list decoding property of $\mathcal{G}$. Let $\overline{C}_A : A \to \mathcal{D}_{2,dec}$ be an assignment. Let $C_A : A \to \mathcal{D}_{1,dec}$ be its equivalent assignment. By the list decoding property of $\mathcal{G}$, for all edges $e \in E$, but at most $O(\delta)$ fraction, e is either not satisfied or reads one of $f_1, \ldots, f_l$ in $\mathcal{G}$ under C_A, C_B. Hence, by Proposition 12.1, for all edges $e \in E$, but at most $O(\delta)$ fraction, e is either not satisfied or reads one of $f_1, \ldots, f_l$ in $\overline{\mathcal{G}}$ under $\overline{C}_A, C_B$. $\qquad\square$

Transforming RM-LR construction algorithms. Assume that we are given a $(\mathcal{D}, k, N)$-RM-LR construction algorithm $\mathcal{A}_1$ with structural parameters $\langle \mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright} \rangle$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$. The $(\mathcal{D}, k, N)$-RM-LR construction algorithm $\mathcal{A}_2$ with structural parameters $\langle \mathsf{size}, \mathsf{block}', \mathsf{degleft}, \mathsf{degright} \rangle$ reducing $\mathcal{D} \mapsto \mathcal{D}_2$ will be as follows: Given an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Invoke $\mathcal{A}_1$ on the input tuples to obtain an RM-LR $\mathcal{G}$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for the input tuples. Then perform the power reduction manipulation on RM-LRs that is described above to obtain an RM-LR $\overline{\mathcal{G}}$ reducing $\mathcal{D} \mapsto \mathcal{D}_2$ for the input tuples, and output $\overline{\mathcal{G}}$.

12.3   The Power Reduction Manipulation on RM-LPRs

Given an RM-LPR $\mathcal{G}$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for some tuples:

$$\mathcal{G} = \langle (A, B, E), V = A, \Omega, \mathcal{D}_1, \widetilde{\mathbb{F}}, sat_{\mathcal{G}} \doteq true, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}}, pnt_{\mathcal{G}}, evalp_{\mathcal{G}} \rangle$$

Let $\mathcal{G}^-$ be the RM-LR reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for the same tuples that is induced by $\mathcal{G}$. Perform the power reduction manipulation on RM-LRs that is described above to obtain an RM-LR $\overline{\mathcal{G}}^-$ reducing

$\mathcal{D} \mapsto \mathcal{D}_2$ for the input tuples:

$$\overline{\mathcal{G}^-} = \langle (A, B, E), V = A, \overline{\Omega}, \mathcal{D}_2, \widetilde{\mathbb{F}}, sat_{\overline{\mathcal{G}^-}} \doteq true, label_{\overline{\mathcal{G}^-}}, proj_{\overline{\mathcal{G}^-}}, tup_{\overline{\mathcal{G}^-}}, eval_{\overline{\mathcal{G}^-}} \rangle$$

Obtain an RM-LPR reducing $\mathcal{D} \mapsto \mathcal{D}_2$ by considering:

$$\overline{\mathcal{G}} = \langle (A, B, E), V = A, \overline{\Omega}, \mathcal{D}_2, \widetilde{\mathbb{F}}, sat_{\overline{\mathcal{G}^-}} \doteq true, label_{\overline{\mathcal{G}^-}}, proj_{\overline{\mathcal{G}^-}}, tup_{\overline{\mathcal{G}^-}}, eval_{\overline{\mathcal{G}^-}}, pnt_{\mathcal{G}}, evalp_{\mathcal{G}} \rangle$$

Note that the RM-LR induced by $\overline{\mathcal{G}}$, denoted $(\overline{\mathcal{G}})^-$, is $\overline{\mathcal{G}^-}$.

Properties of the manipulation. $\overline{\mathcal{G}}$ can be obtained efficiently from $\mathcal{G}$. Moreover, $\overline{\mathcal{G}}$ has the same size and left and right degrees as $\mathcal{G}$, and its block length is $d_1^{O(m_1)} \cdot \log |\mathbb{F}|$.

Analysis. Applying Proposition 12.1 we get:

Proposition 12.3. *For any two equivalent assignments $\overline{C}_A : A \to \mathcal{D}_{2,dec}$ and $C_A : A \to \mathcal{D}_{1,dec}$, for any assignment $C_B : B \to \widetilde{\mathbb{F}}_{dec}$, for any edge $e = (a, b) \in E$, we have:*

- *e is satisfied in $\mathcal{G}$ under C_A, C_B if and only if e is satisfied in $\overline{\mathcal{G}}$ under $\overline{C}_A, C_B$.*

- *e reads some $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under C_A, C_B if and only if e reads f in $\overline{\mathcal{G}}$ under $\overline{C}_A, C_B$.*

Proof. Let us prove the second item. Recall that e reads f in $\mathcal{G}$ under C_A, C_B if and only if e reads f in $\mathcal{G}^-$ under C_A, C_B and $evalp_{\mathcal{G}}(b, C_B(b)) = f(pnt_{\mathcal{G}}(b))$. Similarly, e reads f in $\overline{\mathcal{G}}$ under $\overline{C}_A, C_B$ if and only if e reads f in $(\overline{\mathcal{G}})^- \equiv \overline{\mathcal{G}^-}$ under $\overline{C}_A, C_B$ and $evalp_{\mathcal{G}}(b, C_B(b)) = f(pnt_{\mathcal{G}}(b))$. By Proposition 12.1, e reads f in $\mathcal{G}^-$ under C_A, C_B if and only if e reads f in $\overline{\mathcal{G}^-}$ under $\overline{C}_A, C_B$. $\square$

Hence, similarly to the case of RM-LRs, we have:

Lemma 12.4. *Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function. If $\mathcal{G}$ is a (δ_{min}, l_{max})-RM-LPR reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for some tuples, and $\overline{\mathcal{G}}$ is obtained from $\mathcal{G}$ by the power reduction manipulation, then $\overline{\mathcal{G}}$ is a (δ_{min}, l_{max})-RM-LPR reducing $\mathcal{D} \mapsto \mathcal{D}_2$ for the same tuples.*

Transforming RM-LPR construction algorithms. Assume that we are given a $(\mathcal{D}, k, N)$-RM-LPR construction algorithm $\mathcal{A}_1$ with structural parameters $\langle size, block, degleft, degright \rangle$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$ that is uniform in the point association. Consider the $(\mathcal{D}, k, N)$-RM-LPR construction algorithm $\mathcal{A}_2$ with structural parameters $\langle size, block', degleft, degright \rangle$ reducing $\mathcal{D} \mapsto \mathcal{D}_2$ defined as follows: Given an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \dots, \vec{x}_{1,k} \rangle, \dots, \langle \vec{x}_{N,1}, \dots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Invoke $\mathcal{A}_1$ on the input tuples to obtain an RM-LPR $\mathcal{G}$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$ for the input tuples. Then perform the power reduction manipulation on RM-LPRs that is described above to obtain an RM-LPR $\overline{\mathcal{G}}$ reducing $\mathcal{D} \mapsto \mathcal{D}_2$ for the input tuples, and output $\overline{\mathcal{G}}$. Note that $\mathcal{A}_2$ is uniform in the point association.

13 Right Degree Reduction

In this section we show how to reduce the (graph) degree of the B vertices in the various building blocks we consider, thus proving Lemma 8.5. We start by presenting the right degree manipulation on bipartite constraint graphs. Then we will describe the manipulation on RM-LRs, RM$\diamond$Had-LRs, RM-LPRs, RM-RRs and RM-RPRs.

13.1 The Right Degree Reduction Manipulation on Bipartite Constraint Graphs

The idea of right degree reduction is to split each vertex $b \in B$ into many copies. The number of copies will be the original degree of b in the graph. Each copy will have a small degree in the new graph. This is achieved by putting an expander with small degree in the new graph between the neighbors of b in the original graph and b's copies. Formally, the manipulation is defined as follows:

Given a natural number Δ and a bipartite constraint graph

$$\mathcal{G} = \langle G = (A, B, E), \Omega, \Sigma_A, \Sigma_B, sat_{\mathcal{G}}, label_{\mathcal{G}}, proj_{\mathcal{G}} \rangle$$

we construct a bipartite constraint graph that is right regular with right degree $\Theta(\Delta)$:

$$\overline{\mathcal{G}} = \langle \overline{G} = (A, \overline{B}, \overline{E}), \Omega, \Sigma_A, \Sigma_B, sat_{\mathcal{G}}, label_{\overline{\mathcal{G}}}, proj_{\mathcal{G}} \rangle$$

by performing the following operations:

1. **Bipartite expanders.** Let $\alpha < 1$ and $T : \mathbb{N} \to \mathbb{N}^+$, where $T(\Delta) = \Theta(\Delta)$, be as in Lemma 5.3. For $n = 1, \ldots, |A|$, compute as follows from Lemma 5.3 a $T(\Delta)$-regular bipartite (multi-)graph $\mathcal{H}_n = ([n], [n], E_n)$ satisfying the following expansion property: for every two sets $X \subseteq [n], Y \subseteq [n]$,

$$\frac{|E_n(X,Y)|}{T(\Delta) \cdot n} \leq \frac{|X|}{n} \cdot \frac{|Y|}{n} + \frac{1}{(T(\Delta))^{1-\alpha}} \cdot \sqrt{\frac{|X|}{n}} \cdot \sqrt{\frac{|Y|}{n}}$$

2. $\overline{B}$ **vertices.** For every vertex $b \in B$, for every $i \in [\Delta_G(b)]$, there is a vertex $\overline{b} = \langle b, i \rangle \in \overline{B}$.

   In the remainder of this section we shorthand and write $\Delta(b)$ to denote $\Delta_G(b)$.

3. **Edges.** For every $j \in [\Delta(b)]$, assuming the j'th edge coming into b is $e_G(b,j) = e = (a, b) \in E$, for every edge $(j, i) \in E_{\Delta(b)}$, we put an edge $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$. We set $label_{\overline{\mathcal{G}}}(\overline{e}) \doteq label_{\mathcal{G}}(e)$.
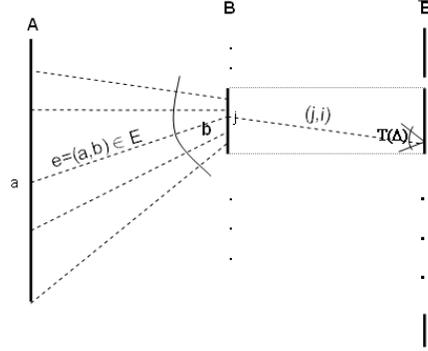
Figure 10: Right degree reduction.

Properties of the manipulation.

- *Running time.* $\overline{\mathcal{G}}$ can be computed from $\mathcal{G}$ efficiently.

- *Size.* The size of $\overline{G}$ is

$$\left|\overline{G}\right| = |A| + \left|\overline{B}\right| + \left|\overline{E}\right| = |A| + \sum_{b \in B} \Delta(b) + \sum_{b \in B} T(\Delta) \cdot \Delta(b) \leq O\left(\Delta \cdot |G|\right)$$

- *Block length.* The manipulation does not change the alphabets.

- *Left degree.* The degree of each of the A vertices in $\overline{G}$ is larger than its degree in G by a factor of $T(\Delta)$.

- *Right degree.* $\overline{G}$ is right regular with right degree $T(\Delta)$.

Analysis. Given an assignment $C_{\overline{B}} : \overline{B} \to \Sigma_B$ to the vertices in $\overline{B}$ and a real parameter $\frac{1}{(T(\Delta))^{1-\alpha}} \leq \delta < 1$, we define a list of size $s \doteq \lfloor \frac{1}{\delta} \rfloor$ of assignments to the B vertices corresponding to $C_{\overline{B}}$ and δ as follows. For every vertex $b \in B$, let $\sigma_1(b), \ldots, \sigma_s(b) \in \Sigma_B$ be all elements $\sigma \in \Sigma_B$ that are assigned to at least a δ fraction of b's copies, i.e.,

$$\left|\{i \in [\Delta(b)] \mid C_{\overline{B}}(\langle b, i \rangle) = \sigma\}\right| \geq \delta \cdot \Delta(b)$$

Note that there are at most s such elements $\sigma \in \Sigma_B$. If there are less than s elements, we pad the list arbitrarily. We define the assignments to the B vertices corresponding to $C_{\overline{B}}$ and δ, $C_{B,1}, \ldots, C_{B,s} : B \to \Sigma_B$, by letting, for every $j \in [s]$ and $b \in B$, $C_{B,j}(b) \doteq \sigma_j(b)$.

We prove the following proposition:

Proposition 13.1. *Let $C_A : A \to \Sigma_A$ be any assignment to the A vertices. When picking uniformly at random an edge $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$, the probability that the edge $\overline{e}$ is satisfied in $\overline{\mathcal{G}}$ under the assignments C_A and $C_{\overline{B}}$, however, $C_{\overline{B}}(\langle b, i \rangle) \notin \{C_{B,1}(b), \ldots, C_{B,s}(b)\}$, is at most 2δ.*

91

Proof. Let $b \in B$. For $\sigma \in \Sigma_B$, define $Y_{b,\sigma} \subseteq [\Delta(b)]$ to be the indices of copies of b assigned σ by $C_{\overline{B}}$,

$$Y_{b,\sigma} \doteq \{i \in [\Delta(b)] \mid C_{\overline{B}}(\langle b, i \rangle) = \sigma \}$$

Let us say that an edge $e = (a, b) \in E$ coming into b *votes* for $\sigma \in \Sigma_B$, if the projection of $C_A(a)$ onto b is σ, i.e., $proj_{\mathcal{G}}(a, C_A(a), label_{\mathcal{G}}(e)) = \sigma$. Define $X_{b,\sigma} \subseteq [\Delta(b)]$ to be the indices of the edges coming into b that vote for σ,

$$X_{b,\sigma} \doteq \{j \in [\Delta(b)] \mid e_G(b, j) \text{ votes for } \sigma \}$$

Note that the sets $X_{b,\cdot}$, as well as the sets $Y_{b,\cdot}$, are pairwise disjoint.

Let $S \subseteq \Sigma_B$ denote the "uncovered" elements $S \doteq \Sigma_B \setminus \{C_{B,1}(b), \dots, C_{B,s}(b)\}$. By definition, for every $\sigma \in S$, it holds that $|Y_{b,\sigma}| \leq \delta \cdot \Delta(b)$. Hence, by the expansion property of $\mathcal{H}_{\Delta(b)}$, we have

$$\frac{\left|E_{\Delta(b)}(X_{b,\sigma}, Y_{b,\sigma})\right|}{T(\Delta) \cdot \Delta(b)} \leq \delta \cdot \frac{|X_{b,\sigma}|}{\Delta(b)} + \frac{1}{(T(\Delta))^{1-\alpha}} \cdot \sqrt{\frac{|X_{b,\sigma}|}{\Delta(b)}} \cdot \sqrt{\frac{|Y_{b,\sigma}|}{\Delta(b)}}$$

Thus, we get the following upper bound on the fractional number of edges $e = (a, \langle b, i \rangle) \in \overline{E}$ that are satisfied by $C_A, C_{\overline{B}}$, although $C_{\overline{B}}(\langle b, i \rangle) \notin \{C_{B,1}(b), \dots, C_{B,s}(b)\}$:

$$\frac{\sum_{\sigma \in S}\left|E_{\Delta(b)}(X_{b,\sigma}, Y_{b,\sigma})\right|}{T(\Delta) \cdot \Delta(b)} \leq \delta \cdot \sum_{\sigma \in S}\frac{|X_{b,\sigma}|}{\Delta(b)} + \frac{1}{(T(\Delta))^{1-\alpha}} \cdot \sum_{\sigma \in S}\sqrt{\frac{|X_{b,\sigma}|}{\Delta(b)}} \cdot \sqrt{\frac{|Y_{b,\sigma}|}{\Delta(b)}}$$

By the Cauchy-Schwarz inequality and using the disjointness of the sets $X_{b,\cdot}$ and of the sets $Y_{b,\cdot}$, we have:

$$\begin{aligned}
\frac{\sum_{\sigma \in S}\left|E_{\Delta(b)}(X_{b,\sigma}, Y_{b,\sigma})\right|}{T(\Delta) \cdot \Delta(b)} &\leq \delta + \frac{1}{(T(\Delta))^{1-\alpha}} \cdot \sqrt{\sum_{\sigma \in S}\frac{|X_{b,\sigma}|}{\Delta(b)}} \cdot \sqrt{\sum_{\sigma \in S}\frac{|Y_{b,\sigma}|}{\Delta(b)}} \\
&\leq \delta + \frac{1}{(T(\Delta))^{1-\alpha}} \\
&\leq 2\delta
\end{aligned}$$

The proposition follows noticing that the probability we wish to bound is at most

$$\frac{\sum_{b \in B} 2\delta \cdot T(\Delta) \cdot \Delta(b)}{\sum_{b \in B} T(\Delta) \cdot \Delta(b)} = 2\delta$$

$\square$

13.2 The Right Degree Manipulation on Composable Bipartite Locally Decode/Reject Codes that are Left Evaluators

Given a natural number Δ and a composable bipartite locally decode/reject code for some tuples:

$$\mathcal{G} = \langle G = (A, B, E), V = A, \Omega, \Sigma_A, \Sigma_B, sat_{\mathcal{G}}, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}} \rangle$$

We construct a composable bipartite locally decode/reject code for the same tuples which is right regular with right degree $T(\Delta)$

$$\overline{\mathcal{G}} = \langle \overline{G} = (A, \overline{B}, \overline{E}), V = A, \Omega, \Sigma_A, \Sigma_B, sat_\mathcal{G}, label_{\overline{\mathcal{G}}}, proj_\mathcal{G}, tup_\mathcal{G}, eval_\mathcal{G} \rangle$$

as follows:

1. Consider the bipartite evaluation graph underlying $\mathcal{G}$:

   $$\mathcal{G}' = \langle G = (A, B, E), V = A, \Omega, \Sigma_{A,dec}, \Sigma_{B,dec}, sat_\mathcal{G}, label_\mathcal{G}, proj_\mathcal{G}, tup_\mathcal{G}, eval_\mathcal{G} \rangle$$

   where $\Sigma_{A,dec}$ and $\Sigma_{B,dec}$ are the decoded domains of Σ_A and Σ_B, respectively.

2. Consider the bipartite constraint graph underlying $\mathcal{G}'$:

   $$\mathcal{G}'' = \langle G = (A, B, E), \Omega, \Sigma_{A,dec}, \Sigma_{B,dec}, sat_\mathcal{G}, label_\mathcal{G}, proj_\mathcal{G} \rangle$$

3. Perform the right degree reduction manipulation on the constraint graph $\mathcal{G}''$ to compute the bipartite constraint graph $\overline{\mathcal{G}''}$:

   $$\overline{\mathcal{G}''} = \langle \overline{G} = (A, \overline{B}, \overline{E}), \Omega, \Sigma_{A,dec}, \Sigma_{B,dec}, sat_\mathcal{G}, label_{\overline{\mathcal{G}''}}, proj_\mathcal{G} \rangle$$

   [Note that $\overline{G}$ is left regular]

4. Define a bipartite evaluation graph:

   $$\overline{\mathcal{G}'} = \langle \overline{G} = (A, \overline{B}, \overline{E}), V = A, \Omega, \Sigma_{A,dec}, \Sigma_{B,dec}, sat_\mathcal{G}, label_{\overline{\mathcal{G}''}}, proj_\mathcal{G}, tup_\mathcal{G}, eval_\mathcal{G} \rangle$$

5. Let the corresponding composable bipartite locally decode/reject code be:

   $$\overline{\mathcal{G}} = \langle \overline{G} = (A, \overline{B}, \overline{E}), V = A, \Omega, \Sigma_A, \Sigma_B, sat_\mathcal{G}, label_{\overline{\mathcal{G}}} \doteq label_{\overline{\mathcal{G}''}}, proj_\mathcal{G}, tup_\mathcal{G}, eval_\mathcal{G} \rangle$$

Properties of the manipulation. $\overline{\mathcal{G}}$ can be obtained efficiently from $\mathcal{G}$. If $\mathcal{G}$ has size size, then $\overline{\mathcal{G}}$ has size $O(\Delta \cdot \text{size})$. $\overline{\mathcal{G}}$ has the same block length as $\mathcal{G}$. It is left regular with left degree larger by a factor of $T(\Delta)$ than the left degree of $\mathcal{G}$. It is right regular with right degree $T(\Delta)$.

Analysis.
Proposition 13.2. *Let $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$ be an edge, and let $e = (a, b) \in E$ be its corresponding edge. For every assignment $C_A : A \to \Sigma_{A,dec}$, for every two assignments $C_B : B \to \Sigma_{B,dec}$ and $C_{\overline{B}} : \overline{B} \to \Sigma_{B,dec}$ such that $C_B(b) = C_{\overline{B}}(\langle b, i \rangle)$ it holds that:*

- *e is satisfied in $\mathcal{G}$ under C_A, C_B if and only if $\overline{e}$ is satisfied in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$.*

- *e reads some $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under C_A, C_B if and only if $\overline{e}$ reads $f \in \mathcal{D}_{dec}$ in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$.*

We thus have:

Lemma 13.3. *Let $0 < \delta_{min} < 1$. Let $l_{max} : (0,1) \to \mathbb{R}^+$ be a decreasing function. Set $\delta^*_{min} \doteq \max\left\{ \sqrt{\delta_{min}}, \frac{1}{(T(\Delta))^{1-\alpha}} \right\}$ and $l^*_{max}(\delta) \doteq \frac{1}{\delta} \cdot l_{max}(\delta^2)$. If $\mathcal{G}$ is a (δ_{min}, l_{max})-composable bipartite locally decode/reject code for some tuples, and $\overline{\mathcal{G}}$ is obtained from $\mathcal{G}$ by the right degree reduction manipulation, then $\overline{\mathcal{G}}$ is a $(\delta^*_{min}, l^*_{max})$-composable bipartite locally decode/reject code for the same tuples.*

Proof. Let us prove encoding and list decoding:

Encoding. Denote the encoded domain of $\mathcal{D}$ by $\mathcal{D}_{enc}$ and the encoded domains of the alphabet domains of $\mathcal{G}$ (and $\overline{\mathcal{G}}$) by $\Sigma_{A,enc}$ and $\Sigma_{B,enc}$. For a polynomial $f \in \mathcal{D}_{enc}$, let $C_A : A \to \Sigma_{A,enc}$ and $C_B : B \to \Sigma_{B,enc}$ be the efficiently computable assignments under which every edge $e \in E$ is satisfied and reads f in $\mathcal{G}$. The assignment $C_{\overline{B}} : \overline{B} \to \Sigma_{B,enc}$, defined by assigning every $\langle b, i \rangle \in \overline{B}$ the value $C_B(b)$, is also efficiently computable. Moreover, by Proposition 13.2, all edges $e \in E$ are satisfied and read f in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$.

List Decoding. Fix an assignment $C_{\overline{B}} : \overline{B} \to \Sigma_{B,dec}$. Fix a real δ such that $\delta^*_{min} \leq \delta < 1$. Let $C_{B,1}, \ldots, C_{B,s} : B \to \Sigma_{B,dec}$ be the $s = \lfloor \frac{1}{\delta} \rfloor$ assignments corresponding to $C_{\overline{B}}$ and δ as in the analysis of the right degree reduction manipulation on bipartite constraint graphs. Fix a real parameter $\delta' \doteq \delta^2 \geq \delta_{min}$. For every $j \in [s]$, let $f_{j,1}, \ldots, f_{j,l} \in \mathcal{D}_{dec}$ be the $l \leq l_{max}(\delta')$ elements guaranteed by the list decoding property of $\mathcal{G}$ for $C_{B,j}$ and δ'. In total we defined at most $\frac{1}{\delta} \cdot l_{max}(\delta^2) = l^*_{max}(\delta)$ elements.

Let $C_A : A \to \Sigma_{A,dec}$ be an assignment.

By the list decoding property of $\mathcal{G}$, for at most $O(s \cdot \delta') = O(\delta)$ fraction of the edges $e \in E$, for some $j \in [s]$, the edge e is satisfied but does not read one of $f_{j,1}, \ldots, f_{j,l}$ in $\mathcal{G}$ under $C_A, C_{B,j}$.

By Proposition 13.1, for all edges $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$, but at most $O(\delta)$ fraction, $\overline{e}$ is not satisfied in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$, or $C_{\overline{B}}(\langle b, i \rangle) \in \{C_{B,1}(b), \ldots, C_{B,s}(b)\}$. Hence, by Proposition 13.2, for all edges $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$, but at most $O(\delta)$ fraction, either $\overline{e}$ is not satisfied in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$, or for some $j \in [s]$ we have $C_{\overline{B}}(\langle b, i \rangle) = C_{B,j}(b)$ and the edge $e = (a, b) \in E$ is satisfied in $\mathcal{G}$ under $C_A, C_{B,j}$.

For a uniformly distributed edge $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$, the edge $e = (a, b)$ is uniformly distributed in E. Hence, for all edges $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$, but at most $O(\delta)$ fraction, either $\overline{e}$ is not satisfied in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$, or for some $j \in [s]$ we have $C_{\overline{B}}(\langle b, i \rangle) = C_{B,j}(b)$ and the edge $e = (a, b) \in E$ reads one of $f_{j,1}, \ldots, f_{j,l}$ in $\mathcal{G}$ under $C_A, C_{B,j}$. Thus, by Proposition 13.2, for all edges $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$, but at most $O(\delta)$ fraction, $\overline{e}$ is either not satisfied or reads one of $f_{1,1}, \ldots, f_{s,l}$ in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$. $\square$

Transforming RM-LR construction algorithms. Assume that we are given a natural number Δ and a $(\mathcal{D}, k, N)$-RM-LR construction algorithm $\mathcal{A}_1$ with structural parameters

$\langle\text{size, block, degleft, degright}\rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$. Consider the $(\mathcal{D}, k, N)$-RM-LR construction algorithm $\mathcal{A}_2$ with structural parameters $\langle O(\Delta \cdot \text{size}), \text{block}, T(\Delta) \cdot \text{degleft}, T(\Delta)\rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ defined as follows: Given an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k}\rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k}\rangle \in (\mathbb{F}^m)^k$$

Invoke $\mathcal{A}_1$ on the input tuples to obtain an RM-LR $\mathcal{G}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples. Then perform the right degree reduction manipulation on composable bipartite locally decode/reject codes that is described above to obtain an RM-LR $\overline{\mathcal{G}}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples, and output $\overline{\mathcal{G}}$.

Transforming RM◇Had-LR construction algorithms. Assume that we are given a natural number Δ and a $(\mathcal{D}, k, N)$-RM◇Had-LR construction algorithm $\mathcal{A}_1$ with structural parameters $\langle\text{size, block, degleft, degright}\rangle$ that is uniform in the tuple association and in the encoding and list decoding, and outputs RM◇Had-LRs whose right degrees do not depend on the input to the algorithm.

Consider the $(\mathcal{D}, k, N)$-RM◇Had-LR construction algorithm $\mathcal{A}_2$ with structural parameters $\langle O(\Delta \cdot \text{size}), \text{block}, T(\Delta) \cdot \text{degleft}, T(\Delta)\rangle$ defined as follows: Given an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k}\rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k}\rangle \in (\mathbb{F}^m)^k$$

Invoke $\mathcal{A}_1$ on the input tuples to obtain an RM◇Had-LR $\mathcal{G}$ for the input tuples. Then perform the right degree reduction manipulation on composable bipartite locally decode/reject codes that is described above to obtain an RM◇Had-LR $\overline{\mathcal{G}}$ for the input tuples, and output $\overline{\mathcal{G}}$.

Note that the algorithm $\mathcal{A}_2$ is uniform in the tuple association, as well as in the encoding and list decoding.

13.3 The Right Degree Manipulation on RM-LPRs

Given a natural number Δ and an RM-LPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for some tuples:

$$\mathcal{G} = \langle (A, B, E), V = A, \Omega, \widetilde{\mathcal{D}}, \widetilde{\mathbb{F}}, sat_{\mathcal{G}} \doteq true, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}}, pnt_{\mathcal{G}}, evalp_{\mathcal{G}}\rangle$$

We construct an RM-LPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the same tuples that has right degree $T(\Delta)$

$$\overline{\mathcal{G}} = \langle (A, \overline{B}, \overline{E}), V = A, \Omega, \widetilde{\mathcal{D}}, \widetilde{\mathbb{F}}, sat_{\overline{\mathcal{G}}} \doteq true, label_{\overline{\mathcal{G}}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}}, pnt_{\overline{\mathcal{G}}}, evalp_{\overline{\mathcal{G}}}\rangle$$

as follows:

1. Consider the RM-LR induced by $\mathcal{G}$:

$$\mathcal{G}^- = \langle (A, B, E), V = A, \Omega, \widetilde{\mathcal{D}}, \widetilde{\mathbb{F}}, sat_{\mathcal{G}}, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}}\rangle$$

2. Perform the right degree reduction manipulation on the RM-LR $\mathcal{G}^-$ to obtain the RM-LR $\overline{\mathcal{G}^-}$:

$$\overline{\mathcal{G}^-} = \langle (A, \overline{B}, \overline{E}), V = A, \Omega, \widetilde{\mathcal{D}}, \widetilde{\mathbb{F}}, sat_{\mathcal{G}}, label_{\overline{\mathcal{G}^-}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}}\rangle$$

3. Assume that the field of the Reed-Muller domain $\mathcal{D}$ is $\mathbb{F}$ and the dimension is m. Define $pnt_{\overline{\mathcal{G}}} : \overline{B} \to \mathbb{F}^m$ by assigning every vertex $\langle b, i \rangle \in \overline{B}$ the point $pnt_{\mathcal{G}}(b)$. Define $evalp_{\overline{\mathcal{G}}} : \overline{B} \times \widetilde{\mathbb{F}}_{dec} \to \mathbb{F}$ by assigning every vertex $\langle b, i \rangle \in \overline{B}$ and assignment $\sigma_b \in \widetilde{\mathbb{F}}_{dec}$ the evaluation $evalp_{\mathcal{G}}(b, \sigma_b)$ [Note that since $\mathcal{G}$ is an RM-LPR, for a uniformly distributed vertex $\langle b, i \rangle \in \overline{B}$, the point $pnt_{\overline{\mathcal{G}}}(\langle b, i \rangle)$ is uniformly distributed in $\mathbb{F}^m$].

4. Let the corresponding RM-LPR be:

$$\overline{\mathcal{G}} = \langle (A, \overline{B}, \overline{E}), V = A, \Omega, \widetilde{\mathcal{D}}, \widetilde{\mathbb{F}}, sat_{\overline{\mathcal{G}}} \doteq true, label_{\overline{\mathcal{G}}} \doteq label_{\overline{\mathcal{G}^=}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}}, pnt_{\overline{\mathcal{G}}}, evalp_{\overline{\mathcal{G}}} \rangle$$

Properties of the manipulation. $\overline{\mathcal{G}}$ can be obtained efficiently from $\mathcal{G}$. If $\mathcal{G}$ has size size, then $\overline{\mathcal{G}}$ has size $O(\Delta \cdot size)$. $\overline{\mathcal{G}}$ has the same block length as $\mathcal{G}$. It is left regular with left degree larger by a factor of $T(\Delta)$ than the left degree of $\mathcal{G}$. It is right regular with right degree $T(\Delta)$.

Analysis. Applying Proposition 13.2, we get:

Proposition 13.4. *Let $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$ be an edge, and let $e = (a, b) \in E$ be its corresponding edge. For every assignment $C_A : A \to \widetilde{\mathcal{D}}_{dec}$, for every two assignments $C_B : B \to \widetilde{\mathbb{F}}_{dec}$ and $C_{\overline{B}} : \overline{B} \to \widetilde{\mathbb{F}}_{dec}$ such that $C_B(b) = C_{\overline{B}}(\langle b, i \rangle)$ it holds that:*

- *e is satisfied in $\mathcal{G}$ under C_A, C_B if and only if $\overline{e}$ is satisfied in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$.*

- *e reads some $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under C_A, C_B if and only if $\overline{e}$ reads $f \in \mathcal{D}_{dec}$ in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$.*

Thus, similarly to Lemma 13.3, we have:

Lemma 13.5. *Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function. Set $\delta^*_{min} \doteq \max \left\{ \sqrt{\delta_{min}}, \frac{1}{(T(\Delta))^{1-\alpha}} \right\}$ and $l^*_{max}(\delta) \doteq \frac{1}{\delta} \cdot l_{max}(\delta^2)$. If $\mathcal{G}$ is a (δ_{min}, l_{max})-RM-LPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for some tuples, and $\overline{\mathcal{G}}$ is obtained from $\mathcal{G}$ by the right degree reduction manipulation, then $\overline{\mathcal{G}}$ is a $(\delta^*_{min}, l^*_{max})$-RM-LPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the same tuples.*

Transforming RM-LPR construction algorithms. Assume that we are given a natural number Δ and a $(\mathcal{D}, k, N)$-RM-LPR construction algorithm $\mathcal{A}_1$ with structural parameters $\langle size, block, degleft, degright \rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$. Consider the $(\mathcal{D}, k, N)$-RM-LPR construction algorithm $\mathcal{A}_2$ with structural parameters $\langle O(\Delta \cdot size), block, T(\Delta) \cdot degleft, T(\Delta) \rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ defined as follows: Given an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Invoke $\mathcal{A}_1$ on the input tuples to obtain an RM-LPR $\mathcal{G}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples. Then perform the right degree reduction manipulation on RM-LPRs that is described above to obtain an RM-LPR $\overline{\mathcal{G}}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples, and output $\overline{\mathcal{G}}$. Note that if $\mathcal{A}_1$ is uniform in the point association, then so is $\mathcal{A}_2$.

13.4 The Right Degree Manipulation on RM-RRs

Given a natural number Δ and an RM-RR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for some tuples:

$$\mathcal{G} = \langle (A, B, E), V = B, \Omega, \Sigma_A, \widetilde{\mathcal{D}}, sat_{\mathcal{G}}, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}} \rangle$$

We construct an RM-RR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the same tuples that has right degree $T(\Delta)$

$$\overline{\mathcal{G}} = \langle (A, \overline{B}, \overline{E}), \overline{V} = \overline{B}, \Omega, \Sigma_A, \widetilde{\mathcal{D}}, sat_{\mathcal{G}}, label_{\overline{\mathcal{G}}}, proj_{\mathcal{G}}, tup_{\overline{\mathcal{G}}}, eval_{\overline{\mathcal{G}}} \rangle$$

as follows:

1. Consider the bipartite evaluation graph underlying $\mathcal{G}$:

   $$\mathcal{G}' = \langle (A, B, E), V = B, \Omega, \Sigma_{A,dec}, \widetilde{\mathcal{D}}_{dec}, sat_{\mathcal{G}}, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}} \rangle$$

   where $\Sigma_{A,dec}$ and $\widetilde{\mathcal{D}}_{dec}$ are the decoded domains of Σ_A and $\widetilde{\mathcal{D}}$, respectively.

2. Consider the bipartite constraint graph underlying $\mathcal{G}'$:

   $$\mathcal{G}'' = \langle (A, B, E), \Omega, \Sigma_{A,dec}, \widetilde{\mathcal{D}}_{dec}, sat_{\mathcal{G}}, label_{\mathcal{G}}, proj_{\mathcal{G}} \rangle$$

3. Perform the right degree reduction manipulation on the constraint graph $\mathcal{G}''$ to compute the bipartite constraint graph $\overline{\mathcal{G}''}$:

   $$\overline{\mathcal{G}''} = \langle (A, \overline{B}, \overline{E}), \Omega, \Sigma_{A,dec}, \widetilde{\mathcal{D}}_{dec}, sat_{\mathcal{G}}, label_{\overline{\mathcal{G}''}}, proj_{\mathcal{G}} \rangle$$

   [Note that: (1) For every vertex $a \in A$, for all labels $\xi \in \Omega$, there is the same number of edges $\overline{e} \in \overline{E}$ coming out of a with $label_{\overline{\mathcal{G}''}}(\overline{e}) = \xi$; (2) The graph $\overline{G} = (A, \overline{B}, \overline{E})$ is left regular]

4. Assume that the field of the Reed-Muller domain $\mathcal{D}$ is $\mathbb{F}$ and the dimension is m. Define $tup_{\overline{\mathcal{G}}} : \overline{B} \to (\mathbb{F}^m)^k$ by assigning every vertex $\langle b, i \rangle \in \overline{B}$ the tuple $tup_{\mathcal{G}}(b)$. Define $eval_{\overline{\mathcal{G}}} : \overline{B} \times \widetilde{\mathcal{D}}_{dec} \to \mathbb{F}^k$ by assigning every vertex $\langle b, i \rangle \in \overline{B}$ and assignment $\sigma_b \in \widetilde{\mathcal{D}}_{dec}$ the evaluation $eval_{\mathcal{G}}(b, \sigma_b)$ [Note that the distribution obtained by picking uniformly at random a vertex $b \in B$ and computing $tup_{\mathcal{G}}(b)$ and the distribution obtained by picking uniformly at random a vertex $\langle b, i \rangle \in \overline{B}$ and computing $tup_{\overline{\mathcal{G}}}(\langle b, i \rangle)$ are identical].

5. Define a bipartite evaluation graph:

   $$\overline{\mathcal{G}'} = \langle (A, \overline{B}, \overline{E}), \overline{V} = \overline{B}, \Omega, \Sigma_{A,dec}, \widetilde{\mathcal{D}}_{dec}, sat_{\mathcal{G}}, label_{\overline{\mathcal{G}''}}, proj_{\mathcal{G}}, tup_{\overline{\mathcal{G}}}, eval_{\overline{\mathcal{G}}} \rangle$$

6. Let the corresponding RM-RR be:

   $$\overline{\mathcal{G}} = \langle (A, \overline{B}, \overline{E}), \overline{V} = \overline{B}, \Omega, \Sigma_A, \widetilde{\mathcal{D}}, sat_{\mathcal{G}}, label_{\overline{\mathcal{G}''}}, proj_{\mathcal{G}}, tup_{\overline{\mathcal{G}}}, eval_{\overline{\mathcal{G}}} \rangle$$

Properties of the manipulation. $\overline{\mathcal{G}}$ can be obtained efficiently from $\mathcal{G}$. If $\mathcal{G}$ has size size, then $\overline{\mathcal{G}}$ has size $O(\Delta \cdot \text{size})$. $\overline{\mathcal{G}}$ has the same block length as $\mathcal{G}$. It is left regular with left degree larger by a factor of $T(\Delta)$ than the left degree of $\mathcal{G}$. It is right regular with right degree $T(\Delta)$. The depth of the tree satisfiability constraints of $\overline{\mathcal{G}}$ is the same as that of $\mathcal{G}$.

Analysis.

Proposition 13.6. *Let* $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$ *be an edge, and let* $e = (a, b) \in E$ *be its corresponding edge. For every assignment* $C_A : A \to \Sigma_{A,dec}$, *for every two assignments* $C_B : B \to \widetilde{\mathcal{D}}_{dec}$ *and* $C_{\overline{B}} : \overline{B} \to \widetilde{\mathcal{D}}_{dec}$ *such that* $C_B(b) = C_{\overline{B}}(\langle b, i \rangle)$ *it holds that:*

- *e is satisfied in $\mathcal{G}$ under C_A, C_B if and only if $\overline{e}$ is satisfied in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$.*

- *e reads some $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under C_A, C_B if and only if $\overline{e}$ reads $f \in \mathcal{D}_{dec}$ in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$.*

Thus, similarly to Lemma 13.3, we have:

Lemma 13.7. *Let* $0 < \delta_{min} < 1$. *Let* $l_{max} : (0, 1) \to \mathbb{R}^+$ *be a decreasing function. Set* $\delta^*_{min} \doteq \max \left\{ \sqrt{\delta_{min}}, \frac{1}{(T(\Delta))^{1-\alpha}} \right\}$ *and* $l^*_{max}(\delta) \doteq \frac{1}{\delta} \cdot l_{max}(\delta^2)$. *If $\mathcal{G}$ is a (δ_{min}, l_{max})-RM-RR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for some tuples, and $\overline{\mathcal{G}}$ is obtained from $\mathcal{G}$ by the right degree reduction manipulation, then $\overline{\mathcal{G}}$ is a $(\delta^*_{min}, l^*_{max})$-RM-RR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the same tuples.*

Transforming RM-RR construction algorithms. Assume that we are given a natural number Δ and a $(\mathcal{D}, k, N)$-RM-RR construction algorithm $\mathcal{A}_1$ with structural parameters $\langle \text{size}, \text{block}, \text{degleft}, \text{degright}, \text{depth} \rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$. Consider the $(\mathcal{D}, k, N)$-RM-RR construction algorithm $\mathcal{A}_2$ with structural parameters $\langle O(\Delta \cdot \text{size}), \text{block}, T(\Delta) \cdot \text{degleft}, T(\Delta), \text{depth} \rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ defined as follows: Given an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Invoke $\mathcal{A}_1$ on the input tuples to obtain an RM-RR $\mathcal{G}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples. Then perform the right degree reduction manipulation on RM-RRs that is described above to obtain an RM-RR $\overline{\mathcal{G}}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples, and output $\overline{\mathcal{G}}$.

13.5 The Right Degree Manipulation on RM-RPRs

Given a natural number Δ and an RM-RPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for some k-tuples:

$$\mathcal{G} = \langle (A, B, E), V = B, \Omega, \Sigma_A, \widetilde{\mathcal{D}}, sat_{\mathcal{G}}, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}}, pnt_{\mathcal{G}}, evalp_{\mathcal{G}} \rangle$$

We construct an RM-RPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the same tuples which has right degree $T(\Delta)$

$$\overline{\mathcal{G}} = \langle (A, \overline{B}, \overline{E}), \overline{V} = \overline{B}, \Omega, \Sigma_A, \widetilde{\mathcal{D}}, sat_{\mathcal{G}}, label_{\overline{\mathcal{G}}}, proj_{\mathcal{G}}, tup_{\overline{\mathcal{G}}}, eval_{\overline{\mathcal{G}}}, pnt_{\mathcal{G}}, evalp_{\mathcal{G}} \rangle$$

as follows:

1. Consider the RM-RR induced by $\mathcal{G}$:

$$\mathcal{G}^- = \langle (A, B, E), V = B, \Omega, \Sigma_A, \widetilde{\mathcal{D}}, sat_\mathcal{G}, label_\mathcal{G}, proj_\mathcal{G}, tup_\mathcal{G}, eval_\mathcal{G} \rangle$$

2. Perform the right degree reduction manipulation on the RM-RR $\mathcal{G}^-$ to obtain the RM-RR $\overline{\mathcal{G}^-}$:

$$\overline{\mathcal{G}^-} = \langle (A, \overline{B}, \overline{E}), \overline{V} = \overline{B}, \Omega, \Sigma_A, \widetilde{\mathcal{D}}, sat_\mathcal{G}, label_{\overline{\mathcal{G}^-}}, proj_\mathcal{G}, tup_{\overline{\mathcal{G}^-}}, eval_{\overline{\mathcal{G}^-}} \rangle$$

3. Let the corresponding RM-RPR be:

$$\overline{\mathcal{G}} = \langle (A, \overline{B}, \overline{E}), \overline{V} = \overline{B}, \Omega, \Sigma_A, \widetilde{\mathcal{D}}, sat_\mathcal{G}, label_{\overline{\mathcal{G}^-}}, proj_\mathcal{G}, tup_{\overline{\mathcal{G}^-}}, eval_{\overline{\mathcal{G}^-}}, pnt_\mathcal{G}, evalp_\mathcal{G} \rangle$$

Properties of the manipulation. $\overline{\mathcal{G}}$ can be obtained efficiently from $\mathcal{G}$. If $\mathcal{G}$ has size size, then $\overline{\mathcal{G}}$ has size $O(\Delta \cdot \text{size})$. $\overline{\mathcal{G}}$ has the same block length as $\mathcal{G}$. It is left regular with left degree larger by a factor of $T(\Delta)$ than the left degree of $\mathcal{G}$. It is right regular with right degree $T(\Delta)$. The depth of the tree satisfiability constraints of $\overline{\mathcal{G}}$ is the same as that of $\mathcal{G}$.

Analysis. Applying Proposition 13.6, we get:

Proposition 13.8. *Let $\overline{e} = (a, \langle b, i \rangle) \in \overline{E}$ be an edge, and let $e = (a, b) \in E$ be its corresponding edge. For every assignment $C_A : A \to \Sigma_{A,dec}$, for every two assignments $C_B : B \to \widetilde{\mathcal{D}}_{dec}$ and $C_{\overline{B}} : \overline{B} \to \widetilde{\mathcal{D}}_{dec}$ such that $C_B(b) = C_{\overline{B}}(\langle b, i \rangle)$ it holds that:*

- *e is satisfied in $\mathcal{G}$ under C_A, C_B if and only if $\overline{e}$ is satisfied in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$.*

- *e reads some $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under C_A, C_B if and only if $\overline{e}$ reads $f \in \mathcal{D}_{dec}$ in $\overline{\mathcal{G}}$ under $C_A, C_{\overline{B}}$.*

Thus, similarly to Lemma 13.3, we have:

Lemma 13.9. *Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function. Set $\delta_{min}^* \doteq \max\left\{\sqrt{\delta_{min}}, \frac{1}{(T(\Delta))^{1-\alpha}}\right\}$ and $l_{max}^*(\delta) \doteq \frac{1}{\delta} \cdot l_{max}(\delta^2)$. If $\mathcal{G}$ is a (δ_{min}, l_{max})-RM-RPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for some tuples, and $\overline{\mathcal{G}}$ is obtained from $\mathcal{G}$ by the right degree reduction manipulation, then $\overline{\mathcal{G}}$ is a $(\delta_{min}^*, l_{max}^*)$-RM-RPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the same tuples.*

Transforming RM-RPR construction algorithms. Assume that we are given a natural number Δ and a $(\mathcal{D}, k, N)$-RM-RPR construction algorithm $\mathcal{A}_1$ with structural parameters $\langle \text{size}, \text{block}, \text{degleft}, \text{degright}, \text{depth} \rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$. Consider the $(\mathcal{D}, k, N)$-RM-RPR construction algorithm $\mathcal{A}_2$ with structural parameters $\langle O(\Delta \cdot \text{size}), \text{block}, T(\Delta) \cdot \text{degleft}, T(\Delta), \text{depth} \rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ defined as follows: Given an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Invoke $\mathcal{A}_1$ on the input tuples to obtain an RM-RPR $\mathcal{G}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples. Then perform the right degree reduction manipulation on RM-RPRs that is described above to obtain an RM-RPR $\overline{\mathcal{G}}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples, and output $\overline{\mathcal{G}}$. Note that if $\mathcal{A}_1$ is uniform in the point association, then so is $\mathcal{A}_2$.

14 Transforming Reed-Muller Left Readers Into Reed-Muller Right Readers

In this section we show how to transform RM-LR construction algorithms into RM-RR construction algorithms and RM-LPR construction algorithms into RM-RPR construction algorithms, thus proving Lemma 8.6.

14.1 Transforming RM-LRs Into RM-RRs

To transform RM-LRs into RM-RRs we switch the roles of the A and the B vertices. After the switch, the B vertices project onto the A vertices, instead of the A vertices projecting onto the B vertices. This is achieved by letting assignments to each vertex $b \in B$ contain assignments to all the A vertices neighboring it. This causes the block length to increase by a factor equal to the degree of the B vertices.

We add satisfiability constraints to the B vertices. The constraints ensure that the assignments to A vertices contained in an assignment to a vertex $b \in B$ are consistent on their projection onto b in the original RM-LR. These constraints are formulated in terms of tree satisfiability constraints (see the discussion before Definition 7.5).

Formally, the switching sides manipulation on RM-LRs is as follows:

Given a right regular RM-LR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for some tuples that has right degree degright:

$$\mathcal{G} = \langle G = (A, B, E), V = A, \Omega, \widetilde{\mathcal{D}}, \widetilde{\mathbb{F}}, sat_{\mathcal{G}} \equiv true, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}} \rangle$$

we construct an RM-RR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the same tuples

$$\overline{\mathcal{G}} = \langle \overline{G} = (\overline{A}, \overline{B}, \overline{E}), \overline{V} = \overline{B}, \overline{\Omega}, \Sigma_{\overline{A}}, \widetilde{\mathcal{D}}, sat_{\overline{\mathcal{G}}}, label_{\overline{\mathcal{G}}}, proj_{\overline{\mathcal{G}}}, tup_{\overline{\mathcal{G}}}, eval_{\overline{\mathcal{G}}} \rangle$$

by performing the following operations:

1. Set $\overline{A} \doteq B$ and $\overline{B} \doteq A$.

2. For every edge $e = (a, b) \in E$ there is an edge $\overline{e} = (b, a) \in \overline{E}$. Note that $\overline{G} = (\overline{A}, \overline{B}, \overline{E})$ is left regular.

3. Set $\overline{\Omega} \doteq [\text{degright}]$. If $e = (a, b) \in E$ is the i'th edge coming into b in G for $i \in [\text{degright}]$ and $\overline{e} = (b, a) \in \overline{E}$, then we set $label_{\overline{\mathcal{G}}}(\overline{e}) \doteq i$. Note that for every vertex $b \in \overline{A}$, for all labels $i \in [\text{degright}]$ there is one edge $\overline{e} \in \overline{E}$ coming out of b with $label_{\overline{\mathcal{G}}}(\overline{e}) = i$.

4. The alphabet domain $\Sigma_{\overline{A}}$ and the projection function $proj_{\overline{\mathcal{G}}}$ are as in the definition of RM-RRs (Definition 7.5).

5. Set $tup_{\overline{\mathcal{G}}} \equiv tup_{\mathcal{G}}$ and $eval_{\overline{\mathcal{G}}} \equiv eval_{\mathcal{G}}$.

6. Recall that by the definition of RM-LRs, we have that $\Omega = \mathbb{F}^w$ where $\mathbb{F}$ is the field of $\widetilde{\mathcal{D}}$, and w is its dimension. For every vertex $\overline{a} = b \in \overline{A}$, define a satisfiability tree $T_{\overline{a}}$ of depth 1 as follows. The tree contains a root $u_0 \doteq b$ that has as its children the elements in $\overline{\Omega} = $ [degright]. The elements in $\overline{\Omega}$ are the leaves of the tree. To complete the description of the tree satisfiability constraints we need to define for every $i \in$ [degright] a function $P_{\overline{a},i} : \{0\} \rightarrow \mathbb{F}^w$ specifying a point in $\mathbb{F}^w$ for the root u_0. For every index $i \in$ [degright] corresponding to an edge $e = e_G(b, i) \in E$, set $P_{\overline{a},i}(0) \doteq label_{\mathcal{G}}(e) \in \mathbb{F}^w$.

Properties of the manipulation. $\overline{\mathcal{G}}$ can be obtained efficiently from $\mathcal{G}$. $\overline{\mathcal{G}}$ has the same size as $\mathcal{G}$. The block length of $\overline{\mathcal{G}}$ is larger than the block length of $\mathcal{G}$ by a factor equal to the right degree of $\mathcal{G}$. $\overline{\mathcal{G}}$ is left and right regular with left degree equal to the right degree of $\mathcal{G}$ and right degree equal to the left degree of $\mathcal{G}$. The depth of the tree satisfiability constraints of $\overline{\mathcal{G}}$ is 1.

Analysis. Denote $\mathcal{D} = \langle \mathbb{F}^m, \mathbb{F}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle, \widetilde{\mathcal{D}} = \langle \mathbb{F}^w, \mathbb{F}, \widetilde{\mathcal{D}}_{enc}, \widetilde{\mathcal{D}}_{dec} \rangle, \Sigma_{\overline{A}} = \langle \overline{\Omega}, \widetilde{\mathcal{D}}_{dec}, \Sigma_{\overline{A},enc}, \Sigma_{\overline{A},dec} \rangle$ and $\widetilde{\mathbb{F}} = \langle \{1\}, \mathbb{F}, \widetilde{\mathbb{F}}_{enc}, \widetilde{\mathbb{F}}_{dec} \rangle$.

Let $\overline{C_{\overline{A}}} : \overline{A} \rightarrow \Sigma_{\overline{A},dec}$ be an assignment to the $\overline{A}$ vertices in $\overline{\mathcal{G}}$. We define its *projected assignment* $C_B : B \rightarrow \widetilde{\mathbb{F}}_{dec}$ to the B vertices in $\mathcal{G}$ by assigning every vertex $b = \overline{a} \in B$ an element in $\widetilde{\mathbb{F}}_{dec}$ as follows: If $\sigma_{\overline{a}} \doteq \overline{C}_{\overline{A}}(\overline{a})$ is not satisfying, i.e., $sat_{\overline{\mathcal{G}}}(\overline{a}, \sigma_{\overline{a}}) = false$, let $C_B(b)$ be an arbitrary element in $\widetilde{\mathbb{F}}_{dec}$. Otherwise, denote by $\vec{x} = P_{\overline{a},1}(0) \in \mathbb{F}^w$ the point that the (arbitrary) leaf 1 specifies for the root of $T_{\overline{a}}$. Let $C_B(b)$ be the element in $\widetilde{\mathbb{F}}_{dec}$ corresponding to $\sigma_{\overline{a}}(1)(\vec{x})$.

Proposition 14.1. *Assume that $C_A : A \rightarrow \widetilde{\mathcal{D}}_{dec}$ and $C_B : B \rightarrow \widetilde{\mathbb{F}}_{dec}$ are assignments to $\mathcal{G}$. Assume that $\overline{C_{\overline{A}}} : \overline{A} \rightarrow \Sigma_{\overline{A},dec}$ and $\overline{C_{\overline{B}}} : \overline{B} \rightarrow \widetilde{\mathcal{D}}_{dec}$ are assignments to $\overline{\mathcal{G}}$, such that the projected assignment of $\overline{C_{\overline{A}}}$ to the B vertices is C_B and $\overline{C_{\overline{B}}} \equiv C_A$. Let $e = (a, b) \in E$ and let $\overline{e} = (b, a) \in \overline{E}$ be its corresponding edge. It holds that:*

- *If $\overline{e}$ is satisfied in $\overline{\mathcal{G}}$ under $\overline{C_{\overline{A}}}$ and $\overline{C_{\overline{B}}}$, then e is satisfied in $\mathcal{G}$ under C_A and C_B.*

- *$\overline{e}$ reads an element $f \in \mathcal{D}_{dec}$ in $\overline{\mathcal{G}}$ under $\overline{C_{\overline{A}}}$ and $\overline{C_{\overline{B}}}$ if and only if e reads f in $\mathcal{G}$ under C_A and C_B.*

Moreover, given assignments $C_A : A \rightarrow \widetilde{\mathcal{D}}_{enc}$ and $C_B : B \rightarrow \widetilde{\mathbb{F}}_{enc}$ such that all edges are satisfied in $\mathcal{G}$ under C_A and C_B, one can efficiently compute assignments $\overline{C_{\overline{A}}} : \overline{A} \rightarrow \Sigma_{\overline{A},enc}$ and $\overline{C_{\overline{B}}} : \overline{B} \rightarrow \widetilde{\mathcal{D}}_{enc}$, such that the projected assignment of $\overline{C_{\overline{A}}}$ to the B vertices is C_B, $\overline{C_{\overline{B}}} \equiv C_A$, and it holds that all edges are satisfied in $\overline{\mathcal{G}}$ under $\overline{C_{\overline{A}}}$ and $\overline{C_{\overline{B}}}$.

Lemma 14.2. *Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \rightarrow \mathbb{R}^+$ be a decreasing function. Set $\delta^*_{min} \doteq \sqrt{\delta_{min}}$ and $l^*_{max}(\delta) \doteq \frac{1}{\delta} \cdot l_{max}(\delta^2)$. If $\mathcal{G}$ is a (δ_{min}, l_{max})-RM-LR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for some tuples, and $\overline{\mathcal{G}}$ is obtained from $\mathcal{G}$ by the switching sides manipulation, then $\overline{\mathcal{G}}$ is a $(\delta^*_{min}, l^*_{max})$-RM-RR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the same tuples.*

Proof. Let us prove encoding and list decoding:

Encoding. Let $f \in \mathcal{D}_{enc}$. Assume that $C_A : A \to \widetilde{\mathcal{D}}_{enc}$ and $C_B : B \to \widetilde{\mathbb{F}}_{enc}$ are the efficiently computable assignments guaranteed by the encoding property of $\mathcal{G}$ for f. By Proposition 14.1, we can efficiently construct assignments $\overline{C}_{\overline{A}} : \overline{A} \to \Sigma_{\overline{A},enc}$ and $\overline{C}_{\overline{B}} : \overline{B} \to \mathcal{D}_{enc}$, such that all edges are satisfied and read f in $\overline{\mathcal{G}}$ under $\overline{C}_{\overline{A}}$ and $\overline{C}_{\overline{B}}$.

List Decoding. Fix an assignment $\overline{C}_{\overline{B}} : \overline{B} \to \widetilde{\mathcal{D}}_{dec}$ and a real δ such that $\delta^*_{min} \leq \delta < 1$. Let $f_1, \ldots, f_l \in \mathcal{D}_{dec}$ be the $l \leq l^*_{max}(\delta)$ elements guaranteed by Proposition 6.11 invoked on the (δ_{min}, l_{max})-generic bipartite locally decode/reject code induced by $\mathcal{G}$, the assignment $C_A \doteqdot \overline{C}_{\overline{B}}$ and the parameter δ.

Let $\overline{C}_{\overline{A}} : \overline{A} \to \Sigma_{\overline{A},dec}$ be an assignment to the $\overline{A}$ vertices in $\overline{\mathcal{G}}$. Let $C_B : B \to \widetilde{\mathbb{F}}_{dec}$ be the projected assignment to the B vertices in $\mathcal{G}$. When picking uniformly at random an edge $e \in E$, the probability that e is satisfied but does not read one of $f_1, \ldots, f_l$ in $\mathcal{G}$ under C_A, C_B, is at most $O(\delta)$. Hence, by Proposition 14.1, when picking uniformly at random an edge $\overline{e} \in \overline{E}$, the probability that $\overline{e}$ is satisfied but does not read one of $f_1, \ldots, f_l$ in $\overline{\mathcal{G}}$ under $\overline{C}_{\overline{A}}, \overline{C}_{\overline{B}}$, is at most $O(\delta)$. $\qquad\square$

Transforming RM-LR construction algorithms. Assume that we are given a $(\mathcal{D}, k, N)$-RM-LR construction algorithm $\mathcal{A}_1$ with structural parameters $\langle \text{size}, \text{block}, \text{degleft}, \text{degright} \rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ that outputs right regular RM-LRs. Consider the $(\mathcal{D}, k, N)$-RM-RR construction algorithm $\mathcal{A}_2$ with structural parameters $\langle \text{size}, \text{degright} \cdot \text{block}, \text{degright}, \text{degleft}, 1 \rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ defined as follows: Given an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Invoke $\mathcal{A}_1$ on the input tuples to obtain a right regular RM-LR $\mathcal{G}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples. Then perform the switching sides manipulation on RM-LRs that is described above to obtain an RM-RR $\overline{\mathcal{G}}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples, and output $\overline{\mathcal{G}}$.

14.2 Transforming RM-LPRs Into RM-RPRs

Given an RM-LPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for some tuples that has right degree degright:

$$\mathcal{G} = \langle G = (A, B, E), V = A, \Omega, \widetilde{\mathcal{D}}, \widetilde{\mathbb{F}}, sat_{\mathcal{G}} \doteqdot true, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}}, pnt_{\mathcal{G}}, evalp_{\mathcal{G}} \rangle$$

where $\widetilde{\mathcal{D}} = \langle \mathbb{F}^w, \mathbb{F}, \widetilde{\mathcal{D}}_{enc}, \widetilde{\mathcal{D}}_{dec} \rangle$ and $\widetilde{\mathbb{F}} = \langle \{1\}, \mathbb{F}, \widetilde{\mathbb{F}}_{enc}, \widetilde{\mathbb{F}}_{dec} \rangle$, we construct an RM-RPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the same tuples

$$\overline{\mathcal{G}} = \langle \overline{G} = (\overline{A}, \overline{B}, \overline{E}), \overline{V} = \overline{B}, \overline{\Omega}, \Sigma_{\overline{A}}, \widetilde{\mathcal{D}}, sat_{\overline{\mathcal{G}}}, label_{\overline{\mathcal{G}}}, proj_{\overline{\mathcal{G}}}, tup_{\overline{\mathcal{G}}}, eval_{\overline{\mathcal{G}}}, pnt_{\overline{\mathcal{G}}}, evalp_{\overline{\mathcal{G}}} \rangle$$

by performing the following operations:

1. Consider the RM-LR induced by $\mathcal{G}$:

$$\mathcal{G}^- = \langle G = (A, B, E), V = A, \Omega, \widetilde{\mathcal{D}}, \widetilde{\mathbb{F}}, sat_{\mathcal{G}} \doteq true, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}} \rangle$$

[Note that it is right regular]

2. Perform the switching sides manipulation on RM-LRs to obtain an RM-RR $\overline{\mathcal{G}^-}$:

$$\overline{\mathcal{G}^-} = \langle \overline{G} = (\overline{A}, \overline{B}, \overline{E}), \overline{V} = \overline{B}, \Omega, \Sigma_{\overline{A}}, \widetilde{\mathcal{D}}, sat_{\overline{\mathcal{G}^-}}, label_{\overline{\mathcal{G}^-}}, proj_{\overline{\mathcal{G}^-}}, tup_{\overline{\mathcal{G}^-}}, eval_{\overline{\mathcal{G}^-}} \rangle$$

So $\overline{A} = B$. Denote $\Sigma_{\overline{A}} = \langle \overline{\Omega}, \widetilde{\mathcal{D}}_{dec}, \Sigma_{\overline{A}, enc}, \Sigma_{\overline{A}, dec} \rangle$.

3. Set $pnt_{\overline{\mathcal{G}}} \doteq pnt_{\mathcal{G}}$. Define a function $evalp_{\overline{\mathcal{G}}} : \overline{A} \times \Sigma_{\overline{A}, dec} \to \mathbb{F}$ as in Definition 7.6.

4. Let the corresponding RM-RPR $\overline{\mathcal{G}}$ be:

$$\overline{\mathcal{G}} = \langle \overline{G}, \overline{V}, \overline{\Omega}, \Sigma_{\overline{A}}, \widetilde{\mathcal{D}}, sat_{\overline{\mathcal{G}^-}}, label_{\overline{\mathcal{G}^-}}, proj_{\overline{\mathcal{G}^-}}, tup_{\overline{\mathcal{G}^-}} eval_{\overline{\mathcal{G}^-}}, pnt_{\overline{\mathcal{G}}}, evalp_{\overline{\mathcal{G}}} \rangle$$

Properties of the manipulation. $\overline{\mathcal{G}}$ can be obtained efficiently from $\mathcal{G}$. $\overline{\mathcal{G}}$ has the same size as $\mathcal{G}$. The block length of $\overline{\mathcal{G}}$ is larger than the block length of $\mathcal{G}$ by a factor equal to the right degree of $\mathcal{G}$. $\overline{\mathcal{G}}$ is left and right regular with left degree equal to the right degree of $\mathcal{G}$ and right degree equal to the left degree of $\mathcal{G}$. The depth of the tree satisfiability constraints of $\overline{\mathcal{G}}$ is 1.

Analysis. Using Proposition 14.1, we get:

Proposition 14.3. *Assume that $C_A : A \to \widetilde{\mathcal{D}}_{dec}$ and $C_B : B \to \widetilde{\mathbb{F}}_{dec}$ are assignments to $\mathcal{G}$. Assume that $\overline{C}_{\overline{A}} : \overline{A} \to \Sigma_{\overline{A}, dec}$ and $\overline{C}_{\overline{B}} : \overline{B} \to \widetilde{\mathcal{D}}_{dec}$ are assignments to $\overline{\mathcal{G}}$, such that the projected assignment of $\overline{C}_{\overline{A}}$ to the B vertices is C_B and $\overline{C}_{\overline{B}} \equiv C_A$. Let $e = (a, b) \in E$ and let $\overline{e} = (b, a) \in \overline{E}$ be its corresponding edge. It holds that:*

- *If $\overline{e}$ is satisfied in $\overline{\mathcal{G}}$ under $\overline{C}_{\overline{A}}$ and $\overline{C}_{\overline{B}}$, then e is satisfied in $\mathcal{G}$ under C_A and C_B.*

- *If $\overline{e}$ is satisfied in $\overline{\mathcal{G}}$ under $\overline{C}_{\overline{A}}$ and $\overline{C}_{\overline{B}}$ and e reads an element $f \in \mathcal{D}_{dec}$ in $\mathcal{G}$ under C_A and C_B, then $\overline{e}$ reads f in $\overline{\mathcal{G}}$ under $\overline{C}_{\overline{A}}$ and $\overline{C}_{\overline{B}}$.*

Moreover, given assignments $C_A : A \to \widetilde{\mathcal{D}}_{enc}$ and $C_B : B \to \widetilde{\mathbb{F}}_{enc}$ such that all edges are satisfied in $\mathcal{G}$ under C_A and C_B, one can efficiently compute assignments $\overline{C}_{\overline{A}} : \overline{A} \to \Sigma_{\overline{A}, enc}$ and $\overline{C}_{\overline{B}} : \overline{B} \to \widetilde{\mathcal{D}}_{enc}$, such that the projected assignment of $\overline{C}_{\overline{A}}$ to the B vertices is C_B, $\overline{C}_{\overline{B}} \equiv C_A$, and it holds that all edges are satisfied in $\overline{\mathcal{G}}$ under $\overline{C}_{\overline{A}}$ and $\overline{C}_{\overline{B}}$.

Similarly to Lemma 14.2, we have:

Lemma 14.4. *Let $0 < \delta_{min} < 1$. Let $l_{max} : (0, 1) \to \mathbb{R}^+$ be a decreasing function. Set $\delta_{min}^* \doteq \sqrt{\delta_{min}}$ and $l_{max}^*(\delta) \doteq \frac{1}{\delta} \cdot l_{max}(\delta^2)$. If $\mathcal{G}$ is a (δ_{min}, l_{max})-RM-LPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for some tuples, and $\overline{\mathcal{G}}$ is obtained from $\mathcal{G}$ by the switching sides manipulation, then $\overline{\mathcal{G}}$ is a $(\delta_{min}^*, l_{max}^*)$-RM-RPR reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the same tuples.*

Transforming RM-LPR construction algorithms. Assume that we are given a $(\mathcal{D}, k, N)$-RM-LPR construction algorithm $\mathcal{A}_1$ with structural parameters $\langle \text{size}, \text{block}, \text{degleft}, \text{degright} \rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ that is uniform in the point association. Consider the $(\mathcal{D}, k, N)$-RM-RPR construction algorithm $\mathcal{A}_2$ with structural parameters $\langle \text{size}, \text{degright} \cdot \text{block}, \text{degright}, \text{degleft}, 1 \rangle$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ defined as follows: Given an input to the construction algorithm

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k} \rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k} \rangle \in (\mathbb{F}^m)^k$$

Invoke $\mathcal{A}_1$ on the input tuples to obtain an RM-LPR $\mathcal{G}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples. Then perform the switching sides manipulation on RM-LPRs that is described above to obtain an RM-RPR $\overline{\mathcal{G}}$ reducing $\mathcal{D} \mapsto \widetilde{\mathcal{D}}$ for the input tuples, and output $\overline{\mathcal{G}}$. Note that $\mathcal{A}_2$ is uniform in the point association.

15 Transforming Hadamard Left Readers Into RM◇Had Left Readers

In this section we show how to transform a Had-LR construction algorithm into a RM◇Had-LR construction algorithm, thus proving Lemma 8.7. Recall that a Reed-Muller codeword corresponds to a low degree polynomial. The idea is to take the coefficients of the low degree polynomial, and encode them via a Hadamard code over $\mathbb{L}$. The symbols of the concatenation of the Reed-Muller codeword with a Hadamard code appear in the encoding (we argue that below).

This results in a very wasteful construction, and is hence used only as an inner construction.

We use the notation appearing in Lemma 8.7. Specifically, $\mathcal{D}$ is a RM◇Had domain defined by a finite field $\mathbb{F}$, a prime subfield $\mathbb{L}$ of $\mathbb{F}$, a dimension m, an encoding degree d and a decoding degree d'. The extension degree of $\mathbb{F}$ over $\mathbb{L}$ is $\tau = [\mathbb{F} : \mathbb{L}]$. The number of monomials in an m-variate polynomial of degree at most d is $M \doteq \binom{m+d}{m}$. We consider the Reed-Muller encoding $\mathbb{F}^M \to \mathbb{F}^{|\mathbb{F}^m|}$ which is a linear function over the field $\mathbb{F}$. We consider the Hadamard encoding $\mathbb{L}^\tau \to \mathbb{L}$ which is a linear function over the field $\mathbb{L}$. Identifying $\mathbb{F}$ with the linear subspace $\mathbb{L}^\tau$, the concatenation of the two encodings can be viewed as a linear function over $\mathbb{L}$.

The linear functions corresponding to symbols of RM◇Had Code. By the linearity of the concatenation of the Reed-Muller encoding and the Hadamard encoding, for every pair $\langle \vec{x}, \vec{y} \rangle \in \mathbb{F}^m \times \mathbb{L}^\tau$ representing a position in the concatenation, there is a linear function over $\mathbb{L}$ mapping each vector in $\mathbb{L}^{M \cdot \tau}$ to the symbol of the corresponding codeword at position $\langle \vec{x}, \vec{y} \rangle$. Let $\vec{e}_{\vec{x}, \vec{y}} \in \mathbb{L}^{M \cdot \tau}$ be the coefficients vector of this linear function.

Transforming Had-LR construction algorithms. $\mathcal{H}$ is a Hadamard domain defined by the finite field $\mathbb{L}$ and the dimension $M \cdot \tau$. Note that there is a natural bijection between the encoded domain of $\mathcal{D}$ and the encoded (and decoded) domain of $\mathcal{H}$: the bijection that maps a Reed-Muller codeword

in the encoded domain of $\mathcal{D}$ to the Hadamard encoding of its coefficients in the encoded domain of $\mathcal{H}$.

Assume that we are given a $(\mathcal{H}, k, N)$-Had-LR construction algorithm $\mathcal{A}_1$ with structural parameters $\langle \text{size}, \text{block}, \text{degleft}, \text{degright} \rangle$ that is uniform in the tuple association and in the encoding and list decoding. Moreover, $\mathcal{A}_1$ outputs Had-LRs in which the right degrees of the vertices do not depend on the input to the algorithm.

Let us construct a $(\mathcal{D}, k, N)$-RM$\diamond$Had-LR construction algorithm $\mathcal{A}_2$ with structural parameters $\langle \text{size}, \text{block}, \text{degleft}, \text{degright} \rangle$ that is uniform in the tuple association and in the encoding and list decoding, and outputs RM$\diamond$Had-LRs, in which the right degrees of the vertices do not depend on the input to the algorithm:

Given an input to the construction algorithm

$$\langle (\vec{x}_{1,1}, \vec{y}_{1,1}), \ldots, (\vec{x}_{1,k}, \vec{y}_{1,k}) \rangle, \ldots, \langle (\vec{x}_{N,1}, \vec{y}_{N,1}), \ldots, (\vec{x}_{N,k}, \vec{y}_{N,k}) \rangle \in (\mathbb{F}^m \times \mathbb{L}^\tau)^k$$

Invoke $\mathcal{A}_1$ on the following input:

$$\langle \vec{e}_{\vec{x}_{1,1}, \vec{y}_{1,1}}, \ldots, \vec{e}_{\vec{x}_{1,k}, \vec{y}_{1,k}} \rangle, \ldots, \langle \vec{e}_{\vec{x}_{N,1}, \vec{y}_{N,1}}, \ldots, \vec{e}_{\vec{x}_{N,k}, \vec{y}_{N,k}} \rangle \in (\mathbb{L}^{M \cdot \tau})^k$$

Assume that the output of $\mathcal{A}_1$ is the Had-LR

$$\mathcal{G} = \langle G = (A, B, E), V = A, \Omega, \Sigma_A, \Sigma_B, sat_{\mathcal{G}} \equiv true, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\mathcal{G}}, eval_{\mathcal{G}} \rangle$$

Let us construct a RM$\diamond$Had-LR

$$\overline{\mathcal{G}} = \langle G = (A, B, E), V = A, \Omega, \Sigma_A, \Sigma_B, sat_{\overline{\mathcal{G}}} \equiv true, label_{\mathcal{G}}, proj_{\mathcal{G}}, tup_{\overline{\mathcal{G}}}, eval_{\mathcal{G}} \rangle$$

as follows:

1. For every vertex $a \in A$, assuming that $tup_{\mathcal{G}}(a) = \langle \vec{e}_{\vec{x}_{i,1}, \vec{y}_{i,1}}, \ldots, \vec{e}_{\vec{x}_{i,k}, \vec{y}_{i,k}} \rangle$, let $tup_{\overline{\mathcal{G}}}(a) \doteq \langle (\vec{x}_{i,1}, \vec{y}_{i,1}), \ldots, (\vec{x}_{i,k}, \vec{y}_{i,k}) \rangle$.

Note that $\mathcal{A}_2$ is uniform in the tuple association. Moreover, $\mathcal{A}_2$ outputs RM$\diamond$Had-LRs in which the right degrees of the vertices do not depend on the input to the algorithm.

Assume that $\mathcal{G}$ is a (δ_{min}, l_{max})-Had-LR. Let us show that $\overline{\mathcal{G}}$ is a (δ_{min}, l_{max})-RM$\diamond$Had-LR: Denote $\mathcal{D} = \langle \mathbb{F}^m \times \mathbb{L}^\tau, \mathbb{L}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ and $\mathcal{H} = \langle \mathbb{L}^{M \cdot \tau}, \mathbb{L}, \mathcal{H}_{enc}, \mathcal{H}_{dec} \rangle$. Recall that $\mathcal{H}_{enc} = \mathcal{H}_{dec}$. Denote $\Sigma_A = \langle D_A, R_A, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$ and $\Sigma_B = \langle D_B, R_B, \Sigma_{B,enc}, \Sigma_{B,dec} \rangle$.

Encoding. Let $f \in \mathcal{D}_{enc}$ and denote by $\overline{f} \in \mathcal{H}_{enc}$ the corresponding element. Let $C_A : A \to \Sigma_{A,enc}$ and $C_B : B \to \Sigma_{B,dec}$ be the assignments guaranteed by the encoding property of $\mathcal{G}$ for $\overline{f}$. In $\overline{\mathcal{G}}$ under C_A and C_B, all edges are satisfied and read f.

List Decoding. Fix an assignment $C_B : B \to \Sigma_{B,dec}$ and a real δ such that $\delta_{min} \leq \delta < 1$. Let $\overline{f}_1, \ldots, \overline{f}_l \in \mathcal{H}_{dec} = \mathcal{H}_{enc}$ be the $l \leq l_{max}(\delta)$ elements guaranteed by the list decoding property of $\mathcal{G}$. Let $f_1, \ldots, f_l \in \mathcal{D}_{enc} \subseteq \mathcal{D}_{dec}$ be the corresponding elements. Let $C_A : A \to \Sigma_{A,dec}$. In $\overline{\mathcal{G}}$ under C_A and C_B, when picking uniformly at random an edge $e \in E$, the probability that e is satisfied but does not read one of $f_1, \ldots, f_l$ is at most $O(\delta)$.

Since $\mathcal{A}_1$ is uniform in the encoding and list decoding, so is $\mathcal{A}_2$.

16   Composition of Reed-Muller Right Readers

In this section we show how to compose RM-RR and RM-RPR construction algorithms, thus proving Lemma 8.8.

1. Let $\mathcal{D} = \langle \mathbb{F}^m, \mathbb{F}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension m, an encoding degree d and a decoding degree d'.

2. Let $\mathcal{D}_1 = \langle \mathbb{F}^w, \mathbb{F}, \mathcal{D}_{1,enc}, \mathcal{D}_{1,dec} \rangle$ be a Reed-Muller domain defined by the field $\mathbb{F}$, a dimension w, an encoding degree d_1 and a decoding degree d'_1.

3. Let $\mathcal{D}_2 = \langle \mathbb{F}^\mu, \mathbb{F}, \mathcal{D}_{2,enc}, \mathcal{D}_{2,dec} \rangle$ be a Reed-Muller domain.

Assume that we have outer and inner construction algorithms as follows:

- $\mathcal{A}_{out}$: $(\mathcal{D}, k, N)$-RM-RR construction algorithm with structural parameters
  $(\mathsf{size}_{out}, \mathsf{block}_{out}, \mathsf{degleft}_{out}, \mathsf{degright}_{out}, \mathsf{depth}_{out})$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$, with $\mathsf{depth}_{out} \leq d'_1$.

- $\mathcal{A}_{in}$: $(\mathcal{D}_1, k + \mathsf{degright}_{out} \cdot \mathsf{depth}_{out}, 1)$-RM-RPR construction algorithm with structural parameters $(\mathsf{size}_{in}, \mathsf{block}_{in}, \mathsf{degleft}_{in}, \mathsf{degright}_{in}, 1)$ reducing $\mathcal{D}_1 \mapsto \mathcal{D}_2$ that is uniform in the point association.

We design a composed algorithm $\mathcal{A}$. It will be a $(\mathcal{D}, k, N)$-RM-RR construction algorithm with structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright}, \mathsf{depth})$ reducing $\mathcal{D} \mapsto \mathcal{D}_2$ for $\mathsf{size} \leq \mathsf{size}_{out} \cdot \mathsf{size}_{in}$, $\mathsf{block} = \mathsf{degleft}_{out} \cdot \mathsf{block}_{in}$, $\mathsf{degleft} = \mathsf{degleft}_{out} \cdot \mathsf{degleft}_{in}$, $\mathsf{degright} = \mathsf{degright}_{out} \cdot \mathsf{degright}_{in}$ and $\mathsf{depth} = \mathsf{depth}_{out} + 1$:

Assume that the input to the construction algorithm $\mathcal{A}$ is a collection of k-tuples of points $\langle \vec{x}_{i,1}, \ldots, \vec{x}_{i,k} \rangle \in (\mathbb{F}^m)^k$ for $i \in [N]$. The construction algorithm constructs an RM-RR

$$\mathcal{G} = \langle G = (A, B, E), V = B, \Omega, \Sigma_A, \mathcal{D}_2, sat_\mathcal{G}, label_\mathcal{G}, proj_\mathcal{G}, tup_\mathcal{G}, eval_\mathcal{G} \rangle$$

as follows:

1. **Outer construction.** Invoke construction algorithm $\mathcal{A}_{out}$ on the input k-tuples to obtain an RM-RR

$$\mathcal{G}_{out} = \langle G_{out}, V_{out} = B_{out}, \Omega_{out}, \Sigma_{A_{out}}, \mathcal{D}_1, sat_{\mathcal{G}_{out}}, label_{\mathcal{G}_{out}}, proj_{\mathcal{G}_{out}}, tup_{\mathcal{G}_{out}}, eval_{\mathcal{G}_{out}} \rangle$$

   where $G_{out} = (A_{out}, B_{out}, E_{out})$ and $\Sigma_{A_{out}} = \langle \Omega_{out}, \mathcal{D}_{1,dec}, \Sigma_{A_{out},enc}, \Sigma_{A_{out},dec} \rangle$.

2. **Queried points.** Recall that for every vertex $a_{out} \in A_{out}$ there are tree satisfiability constraints given by a satisfiability tree $T_{a_{out}}$ whose leaves are the elements in Ω_{out} and by ancestors point specification functions $\{P_{a_{out}, \xi_{out}}\}_{\xi_{out} \in \Omega_{out}}$ (See Section 7.5). We say that an edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$ *queries* a point $\vec{x} \in \mathbb{F}^w$, if $\vec{x}$ is one of the points along the path from $\xi_{out} \doteq label_{\mathcal{G}_{out}}(e_{out})$ to the root of the satisfiability tree $T_{a_{out}}$, i.e., there is a depth $i \in \{0, \ldots, \mathsf{depth_{out}} - 1\}$ such that $P_{a_{out}, \xi_{out}}(i) = \vec{x}$.

   For a vertex $b_{out} \in B_{out}$ we define $k + \mathsf{degright_{out}} \cdot \mathsf{depth_{out}}$ *queried points* of b_{out} (possibly with repetitions) as follows:

   - *Vertex queried points (k points):* If $\vec{p}_1, \ldots, \vec{p}_k \in \mathbb{F}^w$ are the points such that for every $\sigma_{b_{out}} \in \mathcal{D}_{1,dec}$ it holds that $eval_{\mathcal{G}_{out}}(b_{out}, \sigma_{b_{out}}) = \langle \sigma_{b_{out}}(\vec{p}_1), \ldots, \sigma_{b_{out}}(\vec{p}_k) \rangle$, then $\vec{p}_1, \ldots, \vec{p}_k$ are the vertex queried points of b_{out}.

   - *Edge queried points (*$\mathsf{degright_{out}} \cdot \mathsf{depth_{out}}$ *points):* For every edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$ coming into b_{out} in G_{out} that queries a point $\vec{x} \in \mathbb{F}^w$, the point $\vec{x}$ is an edge queried point of b_{out}.

   We denote the queried points of b_{out} by $\vec{x}_1^{(b_{out})}, \ldots, \vec{x}_{k + \mathsf{degright_{out}} \cdot \mathsf{depth_{out}}}^{(b_{out})} \in \mathbb{F}^w$, where the first k points are the vertex queried points.

3. **Inner construction.** For every vertex $b_{out} \in B_{out}$, we define an RM-RPR. The purpose of the RM-RPR is to read the queried points of b_{out}.

   The algorithm $\mathcal{A}_{in}$ is uniform in the point association, and, in particular, uniform in structure. Let A_{in}, B_{in}, Ω_{in} and $\Sigma_{A_{in}}$ be such that $\mathcal{A}_{in}$ is uniform in structure $\langle A_{in}, B_{in}, V_{in} = B_{in}, \Omega_{in}, \Sigma_{A_{in}}, \mathcal{D}_2 \rangle$. Denote $\Sigma_{A_{in}} = \langle \Omega_{in}, \mathcal{D}_{2,dec}, \Sigma_{A_{in},enc}, \Sigma_{A_{in},dec} \rangle$. Let $pnt_{in} : A_{in} \to \mathbb{F}^w$ be the uniform point associator.

   For $b_{out} \in B_{out}$, invoke $\mathcal{A}_{in}$ on the queried points of b_{out} to obtain the RM-RPR $\mathcal{G}_{in}^{b_{out}}$:

$$\mathcal{G}_{in}^{b_{out}} = \langle G_{in}^{b_{out}}, V_{in}, \Omega_{in}, \Sigma_{A_{in}}, \mathcal{D}_2, sat_{\mathcal{G}_{in}^{b_{out}}}, label_{\mathcal{G}_{in}^{b_{out}}}, proj_{\mathcal{G}_{in}^{b_{out}}}, tup_{\mathcal{G}_{in}^{b_{out}}}, eval_{\mathcal{G}_{in}^{b_{out}}}, pnt_{in}, evalp_{\mathcal{G}_{in}^{b_{out}}} \rangle$$

   where $G_{in}^{b_{out}} = (A_{in}, B_{in}, E_{in}^{b_{out}})$.

   By definition, there are $k + \mathsf{degright_{out}} \cdot \mathsf{depth_{out}}$ points $\vec{p}_1, \ldots, \vec{p}_{k + \mathsf{degright_{out}} \cdot \mathsf{depth_{out}}} \in \mathbb{F}^\mu$ that correspond to evaluation of the queried points in $\mathcal{G}_{in}^{b_{out}}$, i.e., for every $b_{in} \in B_{in}$ and $\sigma_{b_{in}} \in \mathcal{D}_{2,dec}$,

$$eval_{\mathcal{G}_{in}^{b_{out}}}(b_{in}, \sigma_{b_{in}}) = \langle \sigma_{b_{in}}(\vec{p}_1), \ldots, \sigma_{b_{in}}(\vec{p}_{k + \mathsf{degright_{out}} \cdot \mathsf{depth_{out}}}) \rangle$$

   We call these points *the evaluating points of b_{out}*.

4. **Composed graph.** To construct the composed graph G, for each vertex $a_{out} \in A_{out}$ we produce a copy of A_{in}, and for each vertex $b_{out} \in B_{out}$, we produce a copy of B_{in}. That is, we take:

$$A \doteq \{\langle a_{out}, a_{in}\rangle \mid a_{out} \in A_{out} \wedge a_{in} \in A_{in}\}, \quad B \doteq \{\langle b_{out}, b_{in}\rangle \mid b_{out} \in B_{out} \wedge b_{in} \in B_{in}\}$$

For every two edges $e_{out} = (a_{out}, b_{out}) \in E_{out}$ and $e_{in} = (a_{in}, b_{in}) \in E_{in}^{b_{out}}$, we put an edge $e \in E$ between $\langle a_{out}, a_{in}\rangle \in A$ and $\langle b_{out}, b_{in}\rangle \in B$. Note that the composed graph is left and right regular with left degree $\mathsf{degleft}_{out} \cdot \mathsf{degleft}_{in}$ and right degree $\mathsf{degright}_{out} \cdot \mathsf{degright}_{in}$. The size of the composed graph is less than $\mathsf{size}_{out} \cdot \mathsf{size}_{in}$.
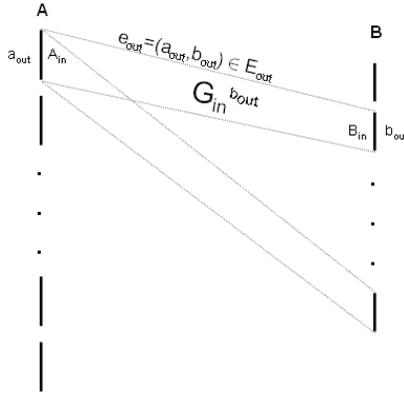
Figure 11: Composed graph.

5. **Labels.** We let $\Omega \doteq [\mathsf{degleft}_{out}] \times \Omega_{in}$. If an edge $e \in E$ corresponds to the outer edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$ and the inner edge $e_{in} \in E_{in}^{b_{out}}$, then $label_G(e)$ is the pair $\langle i, \xi_{in}\rangle \in \Omega$, where $i \in [\mathsf{degleft}_{out}]$ is the index of the outer edge e_{out} among the edges coming out of a_{out}, i.e., $e_{out} = e_{G_{out}}(a_{out}, i)$, and ξ_{in} is the label of the inner edge e_{in} in $\mathcal{G}_{in}^{b_{out}}$, i.e., $\xi_{in} = label_{\mathcal{G}_{in}^{b_{out}}}(e_{in})$. Note that for every vertex $a \in A$, there is the same number of edges coming out of a with each label.

6. **Alphabets.** We let $\Sigma_A = \langle \Omega, \mathcal{D}_{2,dec}, \Sigma_{A,enc}, \Sigma_{A,dec}\rangle$ and $proj_\mathcal{G}$ be as in Definition 7.5. Note that the block length is $\mathsf{degleft}_{out} \cdot \mathsf{block}_{in}$.

7. **Evaluation.** For every vertex $b = \langle b_{out}, b_{in}\rangle \in B$, we set $tup_\mathcal{G}(b) \doteq tup_{\mathcal{G}_{out}}(b_{out})$. Note that each tuple is associated with the same number of B vertices.

   For $\sigma_b \in \mathcal{D}_{2,dec}$, we let $eval_\mathcal{G}(b, \sigma_b)$ be $eval_{\mathcal{G}_{in}^{b_{out}}}(b_{in}, \sigma_b)$ truncated to the first k positions (corresponding to the k vertex queried points).

8. **Tree satisfiability constraints.** Let $a = \langle a_{out}, a_{in}\rangle \in A$. We define tree satisfiability constraints for a. Their purpose is:

(a) To check the satisfiability constraints of a_{out} in $\mathcal{G}_{out}$.

(b) To check the satisfiability constraints of a_{in} in $\mathcal{G}_{in}^{b_{out}}$ for every vertex $b_{out} \in B_{out}$ that neighbors a_{out} in G_{out}.

(c) To check consistency between the assignments to every two vertices $b_{out}^{(1)}, b_{out}^{(2)} \in B_{out}$ that neighbor a_{out} in G_{out} such that the edges $e_{out}^{(1)} = (a_{out}, b_{out}^{(1)}) \in E_{out}$ and $e_{out}^{(2)} = (a_{out}, b_{out}^{(2)}) \in E_{out}$ have the same label $label_{\mathcal{G}_{out}}(e_{out}^{(1)}) = label_{\mathcal{G}_{out}}(e_{out}^{(2)})$. The check is by comparing them on the point $pnt_{in}(a_{in})$ (which is uniformly distributed in $\mathbb{F}^w$ for a uniformly distributed $a_{in} \in A_{in}$).

Let the tree satisfiability constraints of a_{out} in $\mathcal{G}_{out}$ be given by the tree $T_{a_{out}} = (U_{a_{out}} \cup \Omega_{out}, E_{a_{out}})$ and the ancestors point specification functions $\{P_{a_{out}, \xi_{out}}\}_{\xi_{out} \in \Omega_{out}}$. To construct the satisfiability tree T_a for a we take the tree $T_{a_{out}}$ and place in each leaf $\xi_{out} \in \Omega_{out}$ a sub-tree of depth 1. The leaves of the sub-tree are the elements $\xi = \langle i, \xi_{in} \rangle \in \Omega$ for which the i'th edge coming out of a_{out} in G_{out} has label ξ_{out}. Let us denote these elements by $\Omega_{\xi_{out}} \subseteq \Omega$. Then, formally we define $T_a \doteq (U_a \cup \Omega, E_a)$ for $U_a \doteq U_{a_{out}} \cup \Omega_{out}$ and $E_a \doteq E_{a_{out}} \cup \{(\xi_{out}, \xi) \mid \xi_{out} \in \Omega_{out} \wedge \xi \in \Omega_{\xi_{out}}\}$. Note that for every depth, all nodes in that depth have the same number of children, since this property holds for $T_{a_{out}}$ and there is the same number of edges coming out of a_{out} with each label $\xi_{out} \in \Omega_{out}$.

We set the ancestors point specification functions as follows: Let $\xi = \langle i, \xi_{in} \rangle \in \Omega$. Let $b_{out} \in B_{out}$ be the vertex touching the i'th edge $e_{out} \in E_{out}$ coming out of a_{out} in G_{out}, and let $\xi_{out} = label_{\mathcal{G}_{out}}(e_{out})$ be the label of this edge. Let $h \in \{0, \ldots, \mathsf{depth} - 1\}$ be a depth in the tree T_a. Recall that $\mathsf{depth} = \mathsf{depth_{out}} + 1$. We handle the following two cases separately:

- $0 \le h \le \mathsf{depth_{out}} - 1$: Let $\vec{x} = P_{a_{out}, \xi_{out}}(h) \in \mathbb{F}^w$ be the point specified in the tree $T_{a_{out}}$. Then, $\vec{x}$ is a queried point of b_{out}. Assume that it is the j'th queried point of b_{out} where $j \in [k + \mathsf{degright_{out}} \cdot \mathsf{depth_{out}}]$, and let $\vec{p}_j \in \mathbb{F}^\mu$ be the corresponding evaluating point of b_{out}. We set $P_{a, \xi}(h) \doteq \vec{p}_j$.

- $h = \mathsf{depth_{out}}$: Let $P_{a_{in}, \xi_{in}} : \{0\} \to \mathbb{F}^\mu$ be the ancestors point specification function associated with the tree $T_{a_{in}}$ in the RM-RPR $\mathcal{G}_{in}^{b_{out}}$. Then, $P_{a, \xi}(h) \doteq P_{a_{in}, \xi_{in}}(0)$.

16.1 Analysis

Lemma 16.1 (Composition). *Let* $0 < \delta_{min,out}, \delta_{min,in} < 1$. *Let* $l_{max,out}, l_{max,in} : (0, 1) \to \mathbb{R}^+$ *be decreasing functions. Assume that for some constants* $b_1, b_2 \ge 1$ *for every* $0 < \delta < 1$ *it holds that* $l_{max,in}(\delta) \le \frac{b_1}{\delta^{b_2}}$. *Set*

$$\delta_{min} \doteq \max \left\{ \delta_{min,in}, (2b_1^2 \cdot \delta_{min,out})^{1/(2b_2+3)}, \left(2b_1 \cdot \sqrt{\frac{d_1'}{|\mathbb{F}|}}\right)^{1/(b_2+1)} \right\}$$

and

$$l_{max}(\delta) \doteq \frac{b_1}{\delta^{b_2}} \cdot l_{max,out}\left(\frac{1}{2b_1^2} \cdot \delta^{2b_2+3}\right)$$

Assume that $\delta_{min} < 1$.

Then, if $\mathcal{A}_{out}$ outputs $(\delta_{min,out}, l_{max,out})$-RM-RRs, and $\mathcal{A}_{in}$ outputs $(\delta_{min,in}, l_{max,in})$-RM-RPRs, then $\mathcal{A}$ outputs (δ_{min}, l_{max})-RM-RRs.

Proof. We will prove encoding and list decoding:

Encoding. Let $f \in \mathcal{D}_{enc}$. We efficiently construct assignments $C_A : A \to \Sigma_{A,enc}$ and $C_B : B \to \mathcal{D}_{2,enc}$ as follows: Let $C_{A_{out}} : A_{out} \to \Sigma_{A_{out},enc}$ and $C_{B_{out}} : B_{out} \to \mathcal{D}_{1,enc}$ be the assignments for $\mathcal{G}_{out}$ following from the encoding property for f. Let $b_{out} \in B_{out}$. Let $C_{A_{in},b_{out}} : A_{in} \to \Sigma_{A_{in},enc}$ and $C_{B_{in},b_{out}} : B_{in} \to \mathcal{D}_{2,enc}$ be the assignments for $\mathcal{G}_{in}^{b_{out}}$ following from the encoding property for $C_{B_{out}}(b_{out}) \in \mathcal{D}_{1,enc}$.

Let $a = \langle a_{out}, a_{in} \rangle \in A$. Then, $C_A(a)$ is taken to be the function $\sigma_a : [\text{degleft}_{out}] \times \Omega_{in} \to \mathcal{D}_{2,enc}$ defined as follows: For $i \in [\text{degleft}_{out}]$, let $e_{out} = (a_{out}, b_{out}) \in E_{out}$ be the i'th edge coming out of a_{out} in G_{out}. For every $\xi_{in} \in \Omega_{in}$, let $\sigma_a(\langle i, \xi_{in} \rangle) \doteq C_{A_{in},b_{out}}(a_{in})(\xi_{in})$ (Recall that $\Sigma_{A_{in},enc}$ is the set of functions $\Omega_{in} \to \mathcal{D}_{2,enc}$). For every $b = \langle b_{out}, b_{in} \rangle \in B$, let $C_B(b) \doteq C_{B_{in},b_{out}}(b_{in})$.

Let $e = (a, b) \in E$ for $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$. Denote the label of e by $label(e) = \langle i, \xi_{in} \rangle \in \Omega$. Let $e_{out} = (a_{out}, b_{out}) \in E_{out}$ and $e_{in} = (a_{in}, b_{in}) \in E_{in}^{b_{out}}$. We have the following properties:

Reading. We have that $eval_{\mathcal{G}}(b, C_B(b))$ is the first k positions in $eval_{\mathcal{G}_{in}^{b_{out}}}(b_{in}, C_{B_{in},b_{out}}(b_{in}))$, and, since e_{in} reads $C_{B_{out}}(b_{out})$ in $\mathcal{G}_{in}^{b_{out}}$ under $C_{A_{in},b_{out}}$ and $C_{B_{in},b_{out}}$, it holds that $eval_{\mathcal{G}}(b, C_B(b)) = eval_{\mathcal{G}_{out}}(b_{out}, C_{B_{out}}(b_{out}))$. Since e_{out} reads f in $\mathcal{G}_{out}$ under $C_{A_{out}}$ and $C_{B_{out}}$, also e reads f in $\mathcal{G}$ under C_A and C_B.

Projection. We have $proj_{\mathcal{G}}(a, C_A(a), \langle i, \xi_{in} \rangle) = proj_{\mathcal{G}_{in}^{b_{out}}}(a_{in}, C_{A_{in},b_{out}}(a_{in}), \xi_{in}) = C_{B_{in},b_{out}}(b_{in}) = C_B(b)$.

Let $a = \langle a_{out}, a_{in} \rangle \in A$, and let us show that $sat_{\mathcal{G}}(a, C_A(a)) = true$:

Satisfaction. We define an assignment $\sigma : U_a \to \mathbb{F}$ of field elements to the inner nodes of T_a; recall that $U_a = U_{a_{out}} \cup \Omega_{out}$:

- Let $\sigma_1 : U_{a_{out}} \to \mathbb{F}$ be the satisfying assignment to the inner nodes of the satisfiability tree $T_{a_{out}}$ of a_{out} in $\mathcal{G}_{out}$ as follows from $sat_{\mathcal{G}_{out}}(a_{out}, C_{A_{out}}(a_{out})) = true$. For every node $u \in U_{a_{out}}$ let $\sigma(u) \doteq \sigma_1(u)$.

- Let $\sigma_2 : \Omega_{out} \to \mathbb{F}$ be defined as follows. Let $\xi_{out} \in \Omega_{out}$. Denote the polynomial corresponding to the label ξ_{out} by $Q_{\xi_{out}} \stackrel{\mbox{\tiny$\equiv$}}{} C_{A_{out}}(a_{out})(\xi_{out}) \in \mathcal{D}_{1,enc}$. Then, $\sigma_2(\xi_{out})$ evaluates the polynomial $Q_{\xi_{out}}$ on the point associated with a_{in}, namely, $\sigma_2(\xi_{out}) \doteq Q_{\xi_{out}}(pnt_{in}(a_{in}))$ (recall the uniform point association). For every node $u \in \Omega_{out}$, let $\sigma(u) \doteq \sigma_2(u)$.

110

Next we show that this assignment is indeed consistent with the evaluations $C_A(a)(\xi)$ for the leaves $\xi \in \Omega$ of the tree T_a. Let $\xi = \langle i, \xi_{in} \rangle \in \Omega$. Denote the polynomial assigned to ξ by $Q_\xi \doteq C_A(a)(\xi) \in \mathcal{D}_{2,enc}$. Let $e_{out} = (a_{out}, b_{out}) \in E_{out}$ be the i'th edge coming out of a_{out} in G_{out}. Let $\xi_{out} \in \Omega_{out}$ be the label of e_{out}. As before, denote the polynomial assigned to ξ_{out} by the outer assignment $C_{A_{out}}(a_{out})$ by $Q_{\xi_{out}} \doteq C_{A_{out}}(a_{out})(\xi_{out}) \in \mathcal{D}_{1,enc}$.

Let $h \in \{0, \ldots, \mathsf{depth} - 1\}$, and let $u \in U_a$ be the ancestor of ξ in depth h in the tree T_a. We handle the following two cases separately:

- $0 \leq h \leq \mathsf{depth}_{out} - 1$: Let $e_{in} = (a_{in}, b_{in}) \in E_{in}^{b_{out}}$ be some edge with $label_{\mathcal{G}_{in}^{b_{out}}}(e_{in}) = \xi_{in}$. Since $Q_\xi = C_{A_{in},b_{out}}(a_{in})(\xi_{in}) = C_{B_{in},b_{out}}(b_{in})$, $Q_{\xi_{out}} = C_{B_{out}}(b_{out})$ and e_{in} reads $C_{B_{out}}(b_{out})$ in $\mathcal{G}_{in}^{b_{out}}$ under $C_{A_{in},b_{out}}$ and $C_{B_{in},b_{out}}$, we have that $Q_\xi(P_{a,\xi}(h)) = Q_{\xi_{out}}(P_{a_{out},\xi_{out}}(h))$. Since $sat_{\mathcal{G}_{out}}(a_{out}, C_{A_{out}}(a_{out})) = true$, it holds that $Q_{\xi_{out}}(P_{a_{out},\xi_{out}}(h)) = \sigma_1(u)$.

- $h = \mathsf{depth}_{out}$: Since $Q_\xi = C_{A_{in},b_{out}}(a_{in})(\xi_{in})$ and $sat_{\mathcal{G}_{in}^{b_{out}}}(a_{in}, C_{A_{in},b_{out}}(a_{in})) = true$ and by the definition of the $evalp$ function in an RM-RPR (see Definition 7.6) we have that $Q_\xi(P_{a,\xi}(h)) = evalp_{\mathcal{G}_{in}^{b_{out}}}(a_{in}, C_{A_{in},b_{out}}(a_{in}))$. Since $Q_{\xi_{out}} = C_{B_{out}}(b_{out})$ and e_{in} reads $C_{B_{out}}(b_{out})$ in $\mathcal{G}_{in}^{b_{out}}$ under $C_{A_{in},b_{out}}$ and $C_{B_{in},b_{out}}$, we have $Q_\xi(P_{a,\xi}(h)) = Q_{\xi_{out}}(pnt_{in}(a_{in})) = \sigma_2(u)$.

Overall, we get that $Q_\xi(P_{a,\xi}(h)) = \sigma(u)$.

List decoding. Let $C_B : B \to \mathcal{D}_{2,dec}$. Let $\delta_{min} \leq \delta < 1$.

We use the list decoding properties of the inner and outer constructions to define a list decoding for the composed construction.

Set $l_{in} \doteq \lfloor \frac{b_1}{\delta^{b_2}} \rfloor$. Let $b_{out} \in B_{out}$. Let $C_{B_{in},b_{out}} : B_{in} \to \mathcal{D}_{2,dec}$ be the assignment induced by C_B for $\mathcal{G}_{in}^{b_{out}}$ defined by letting every $b_{in} \in B_{in}$ be assigned $C_B(\langle b_{out}, b_{in} \rangle)$. Note that $\delta_{min,in} \leq \delta < 1$. Let $f_{b_{out},1}, \ldots, f_{b_{out},l_{in}} \in \mathcal{D}_{1,dec}$ be the list decoding guaranteed by the property of the RM-RPR $\mathcal{G}_{in}^{b_{out}}$ for the assignment $C_{B_{in},b_{out}}$ and confidence parameter δ (we pad the list arbitrarily if there are less than l_{in} elements in the list decoding). Define l_{in} assignments $C_{B_{out},1}, \ldots, C_{B_{out},l_{in}} : B_{out} \to \mathcal{D}_{1,dec}$ by assigning, for $i \in [l_{in}]$, every vertex $b_{out} \in B_{out}$ to $C_{B_{out},i}(b_{out}) \doteq f_{b_{out},i}$.

Set $\delta_{out} \doteq \frac{1}{2b_1^2} \cdot \delta^{2b_2+3}$, and note that $\delta_{min,out} \leq \delta_{out} < 1$. Set $l_{out} \doteq \lfloor l_{max,out}(\delta_{out}) \rfloor$. For every $i \in [l_{in}]$, let $f_{i,1}, \ldots, f_{i,l_{out}} \in \mathcal{D}_{dec}$ be the list decoding guaranteed by the property of the RM-RR $\mathcal{G}_{out}$ for the assignment $C_{B_{out},i}$ and confidence parameter δ_{out} (we pad the list arbitrarily if there are less than l_{out} elements in the list decoding). In total, we defined a list decoding of size $l_{in} \cdot l_{out} \leq \frac{b_1}{\delta^{b_2}} \cdot l_{max,out}(\frac{1}{2b_1^2} \cdot \delta^{2b_2+3}) = l_{max}(\delta)$.

Fix an assignment $C_A : A \to \Sigma_{A,dec}$. For every edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$ this assignment induces an assignment $C_{A_{in},e_{out}} : A_{in} \to \Sigma_{A_{in},dec}$ to A_{in} in $\mathcal{G}_{in}^{b_{out}}$: Assume that e_{out} is the i'th edge coming out of a_{out} in G_{out} for $i \in [\mathsf{degleft}_{out}]$, i.e., $e_{out} = e_{G_{out}}(a_{out}, i)$. For every vertex $a_{in} \in A_{in}$, take $C_{A_{in},e_{out}}(a_{in})$ to be the function that assigns each $\xi_{in} \in \Omega_{in}$ the value $C_A(\langle a_{out}, a_{in} \rangle)(\langle i, \xi_{in} \rangle)$.

Set $l_{in}^* \doteq \lceil \frac{2b_1}{\delta^{b_2+2}} \rceil$. We will construct assignments $C_{A_{out},1}, \ldots, C_{A_{out},l_{in}^*} : A_{out} \to \Sigma_{A_{out},dec}$, such that the following holds:

Proposition 16.2 (Target outer assignments). *Pick uniformly at random an edge $e = (a, b) \in E$. Let $a = \langle a_{out}, a_{in} \rangle \in A$, $b = \langle b_{out}, b_{in} \rangle \in B$, $e_{out} = (a_{out}, b_{out}) \in E_{out}$, $\xi_{out} = label_{\mathcal{G}_{out}}(e_{out})$ and $e_{in} = (a_{in}, b_{in}) \in E_{in}^{b_{out}}$. With probability at least $1 - O(\delta)$:*

Either the edge e is not satisfied in $\mathcal{G}$ under the assignments C_A and C_B, or there are $i_0 \in [l_{in}]$ and $j_0 \in [l_{in}^*]$, for which: (i) the edge e_{in} reads $C_{B_{out},i_0}(b_{out})$ in $\mathcal{G}_{in}^{b_{out}}$ under the assignments $C_{A_{in},e_{out}}$ and $C_{B_{in},b_{out}}$; (ii) the edge e_{out} is satisfied in $\mathcal{G}_{out}$ under the assignments C_{A_{out},j_0} and C_{B_{out},i_0}.

Note that once we prove Proposition 16.2, we are done, since (using the notation of the proposition): The edge e_{out} is uniformly distributed in E_{out}. Thus, for every $i_0 \in [l_{in}]$ and $j_0 \in [l_{in}^*]$, the probability of the following event is at most $O(\delta_{out})$: (ii) holds, but not (iii) e_{out} reads one of $f_{i_0,1}, \ldots, f_{i_0,l_{out}}$ in $\mathcal{G}_{out}$ under the assignments C_{A_{out},j_0} and C_{B_{out},i_0}. Hence, the probability that this event happens for *some* $i_0 \in [l_{in}]$ and $j_0 \in [l_{in}^*]$ is at most $O(\delta_{out} \cdot l_{in} \cdot l_{in}^*) = O(\delta)$. Moreover, whenever both (i) and (iii) hold, e reads one of $f_{i_0,1}, \ldots, f_{i_0,l_{out}}$ in $\mathcal{G}$ under C_A and C_B.

Constructing the target outer assignments. For every vertex $a_{out} \in A_{out}$, we construct l_{in}^* assignments $\sigma_{a_{out},1}, \ldots, \sigma_{a_{out},l_{in}^*} : \Omega_{out} \to \mathcal{D}_{1,dec}$ to a_{out}. The assignments $C_{A_{out},1}, \ldots, C_{A_{out},l_{in}^*} : A_{out} \to \Sigma_{A_{out},dec}$ are defined for every $i \in [l_{in}^*]$, by assigning the vertex $a_{out} \in A_{out}$ the value $C_{A_{out},i}(a_{out}) \doteq \sigma_{a_{out},i}$. We construct the assignments $\sigma_{a_{out},1}, \ldots, \sigma_{a_{out},l_{in}^*}$ in two steps:

1. *Projection step.* Set $l'_{in} \doteq \lfloor \frac{2}{\delta} \cdot l_{max,in}(\delta) \rfloor$. For every $\xi_{out} \in \Omega_{out}$, we construct a list $f_{a_{out},\xi_{out},1}, \ldots, f_{a_{out},\xi_{out},l'_{in}} \in \mathcal{D}_{1,dec}$ of candidates for ξ_{out}. The list satisfies the following property:

   Proposition 16.3. *Let $a_{out} \in A_{out}$. Let $e_{out} = (a_{out}, b_{out}) \in E_{out}$ be an edge coming out of a_{out} that has label $label_{\mathcal{G}_{out}}(e_{out}) = \xi_{out}$. When picking uniformly at random an edge $e_{in} = (a_{in}, b_{in}) \in E_{in}^{b_{out}}$ and setting $e = (a, b) \in E$ for $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$, the probability that the following holds is at most $O(\delta)$:*

   The edge e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, but e_{in} does not read an element in $\{f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}\} \cap \{f_{a_{out},\xi_{out},1}, \ldots, f_{a_{out},\xi_{out},l'_{in}}\}$ in $\mathcal{G}_{in}^{b_{out}}$ under the assignments $C_{A_{in},e_{out}}$ and $C_{B_{in},b_{out}}$.

2. *Satisfaction step.* By matching the lists $f_{a_{out},\xi_{out},1}, \ldots, f_{a_{out},\xi_{out},l'_{in}} \in \mathcal{D}_{1,dec}$ for different ξ_{out}'s, we construct the l_{in}^* assignments $\sigma_{a_{out},1}, \ldots, \sigma_{a_{out},l_{in}^*} : \Omega_{out} \to \mathcal{D}_{1,dec}$, so that the following property is satisfied:

   Proposition 16.4. *Let $a_{out} \in A_{out}$.*

   (a) *For every $i \in [l_{in}^*]$, it holds that $sat_{\mathcal{G}_{out}}(a_{out}, \sigma_{a_{out},i}) = true$.*

   (b) *Pick uniformly at random an edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$ coming out of a_{out} in $\mathcal{G}_{out}$ and an edge $e_{in} = (a_{in}, b_{in}) \in E_{in}^{b_{out}}$. Set $e = (a, b) \in E$ for $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$. Let $\xi_{out} = label_{\mathcal{G}_{out}}(e_{out})$. The probability that the following holds is at most $O(\delta)$:*

112

The edge e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, but e_{in} does not read an element in $\{f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}\} \cap \{\sigma_{a_{out},1}(\xi_{out}), \ldots, \sigma_{a_{out},l_{in}^*}(\xi_{out})\}$ in $\mathcal{G}_{in}^{b_{out}}$ under the assignments $C_{A_{in},e_{out}}$ and $C_{B_{in},b_{out}}$.

When (in the notation of Proposition 16.4), the edge e_{in} reads an element in $\{f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}\} \cap \{\sigma_{a_{out},1}(\xi_{out}), \ldots, \sigma_{a_{out},l_{in}^*}(\xi_{out})\}$ in $\mathcal{G}_{in}^{b_{out}}$ under the assignments $C_{A_{in},e_{out}}$ and $C_{B_{in},b_{out}}$, it holds that for some $i_0 \in [l_{in}]$ and $j_0 \in [l_{in}^*]$, the edge e_{in} reads $C_{B_{out},i_0}(b_{out})$ in $\mathcal{G}_{in}^{b_{out}}$ under the assignments $C_{A_{in},e_{out}}$ and $C_{B_{in},b_{out}}$, and that the edge e_{out} is satisfied in $\mathcal{G}_{out}$ under C_{A_{out},j_0} and C_{B_{out},i_0}. Thus, once we prove Proposition 16.4, Proposition 16.2 is proved as well, noticing that when $e = (\langle a_{out}, a_{in} \rangle, \langle b_{out}, b_{in} \rangle)$ is uniformly distributed in E, we also have that $e_{out} = (a_{out}, b_{out})$ is uniformly distributed among the edges coming out of a_{out} in G_{out} and $e_{in} = (a_{in}, b_{in})$ is uniformly distributed in $E_{in}^{b_{out}}$.

Let us turn to the construction.

The projection step (Proof of Proposition 16.3).   We will use Lemma 6.8 and our definition of the satisfiability tree.

Let $\xi_{out} \in \Omega_{out}$. Let us define a point evaluation function $pe : A_{in} \to \mathbb{F}$. Let $a_{in} \in A_{in}$ and denote $a = \langle a_{out}, a_{in} \rangle \in A$. If $sat_{\mathcal{G}}(a, C_A(a)) = false$, let $pe(a_{in})$ be an arbitrary field element. Otherwise, let $\sigma : U_a \to \mathbb{F}$ be the implied assignment to the inner nodes of the satisfiability tree T_a. Recall that ξ_{out} is a node in this tree, and define $pe(a_{in}) \doteq \sigma(\xi_{out})$. The decoded domain $\mathcal{D}_{1,dec}$ defines a code with (relative) distance $1 - \frac{d_1'}{|\mathbb{F}|}$, and for every real δ' satisfying $\left(2b_1 \cdot \sqrt{\frac{d_1'}{|\mathbb{F}|}} \right)^{1/(b_2+1)} \le \delta' < 1$ it holds that $\frac{\delta'}{l_{max,in}(\delta')} \ge 2\sqrt{\frac{d_1'}{|\mathbb{F}|}}$. Let $f_{a_{out},\xi_{out},1}, \ldots, f_{a_{out},\xi_{out},l_{in}'} \in \mathcal{D}_{1,dec}$ be the list we get from Lemma 6.8 for the construction algorithm $\mathcal{A}_{in}$, the function pe and the parameter δ (we pad the list arbitrarily if there are less than l_{in}' elements).

Fix an edge $e_{out} = (a_{out}, b_{out})$ with label $label_{\mathcal{G}_{out}}(e_{out}) = \xi_{out}$. Let $e_{in} = (a_{in}, b_{in}) \in E_{in}^{b_{out}}$ and $e = (a, b) \in E$ for $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$. Denote the label of e by $label_{\mathcal{G}}(e) = \langle i, \xi_{in} \rangle$. By the definition of the tree T_a, when $sat_{\mathcal{G}}(a, C_A(a)) = true$ we also have that $sat_{\mathcal{G}_{in}^{b_{out}}}(a_{in}, C_{A_{in},e_{out}}(a_{in})) = true$ and $evalp_{\mathcal{G}_{in}^{b_{out}}}(a_{in}, C_{A_{in},e_{out}}(a_{in})) = pe(a_{in})$. In addition, when $C_A(a)(\langle i, \xi_{in} \rangle) = C_B(b)$, we have $C_{A_{in},e_{out}}(a_{in})(\xi_{in}) = C_{B_{in},b_{out}}(b_{in})$. Hence, when e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, we have that e_{in} is satisfied in $\mathcal{G}_{in}^{b_{out}}$ under the assignments $C_{A_{in},e_{out}}$ and $C_{B_{in},b_{out}}$ and $evalp_{\mathcal{G}_{in}^{b_{out}}}(a_{in}, C_{A_{in},e_{out}}(a_{in})) = pe(a_{in})$. Proposition 16.3 follows from Lemma 6.8.

The satisfaction step (Proof of Proposition 16.4).   Let us say that a vertex $a = \langle a_{out}, a_{in} \rangle \in A$ *checked* an assignment $f_{a_{out},\xi_{out},i} \in \mathcal{D}_{1,dec}$ for $\xi_{out} \in \Omega_{out}$, if there is an edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$ with label $label_{\mathcal{G}_{out}}(e_{out}) = \xi_{out}$ and an edge $e_{in} = (a_{in}, b_{in}) \in E_{in}^{b_{out}}$, such that:

1. The edge e_{in} reads $f_{a_{out},\xi_{out},i}$ in $\mathcal{G}_{in}^{b_{out}}$ under $C_{A_{in},e_{out}}$ and $C_{B_{in},b_{out}}$.

2. For $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$, the edge $e = (a, b) \in E$ is satisfied in $\mathcal{G}$ under C_A and C_B (and, in particular, $sat_{\mathcal{G}}(a, C_A(a)) = true$).

Note that by the definition of the tree satisfiability constraints, if a vertex $a \in A$ checked assignments $f_{a_{out}, \xi_{out}^{(1)}, i_1}, \ldots, f_{a_{out}, \xi_{out}^{(s)}, i_s} \in \mathcal{D}_{1,dec}$ for $\xi_{out}^{(1)}, \ldots, \xi_{out}^{(s)} \in \Omega_{out}$, respectively, then there is an assignment $\sigma_{out} : U_{a_{out}} \to \mathbb{F}$ to the inner nodes of the satisfiability tree $T_{a_{out}}$ that is consistent with all these assignments. That is, for every $j \in [s]$, for every depth $h \in \{0, \ldots, \mathsf{depth}_{out} - 1\}$, if $u \in U_{a_{out}}$ is the ancestor of the leaf $\xi_{out}^{(j)}$ in depth h in the tree $T_{a_{out}}$, then $f_{a_{out}, \xi_{out}^{(j)}, i_j}(P_{a_{out}, \xi_{out}^{(j)}}(h)) = \sigma_{out}(u)$.

We define assignments $\sigma_{a_{out},1}, \sigma_{a_{out},2}, \ldots, \sigma_{a_{out}, l_{in}^*} : \Omega_{out} \to \mathcal{D}_{1,dec}$ using the following procedure:

for $i = 1, 2, \ldots, l_{in}^* + 1$ **do**

1. For every $\xi_{out} \in \Omega_{out}$, let the *uncovered assignments* for ξ_{out} be

$$L_{\xi_{out}} \doteq \{ f_{a_{out}, \xi_{out}, 1}, \ldots, f_{a_{out}, \xi_{out}, l'_{in}} \} - \{ \sigma_{a_{out}, 1}(\xi_{out}), \ldots, \sigma_{a_{out}, i-1}(\xi_{out}) \}$$

2. If there is no $a = \langle a_{out}, a_{in} \rangle \in A$ such that for at least δ fraction of the $\xi_{out} \in \Omega_{out}$, the vertex a checked an uncovered assignment for ξ_{out}, *halt*.

3. Otherwise, let $a \in A$ be such a vertex. For every $\xi_{out} \in \Omega_{out}$ such that a checked an uncovered assignment $f_{a_{out}, \xi_{out}, j} \in L_{\xi_{out}}$ for ξ_{out}, let $\sigma_{a_{out}, i}(\xi_{out}) \doteq f_{a_{out}, \xi_{out}, j}$. Complete $\sigma_{a_{out}, i}$ into an assignment $\Omega_{out} \to \mathcal{D}_{1,dec}$ such that $sat_{\mathcal{G}_{out}}(a_{out}, \sigma_{a_{out}, i}) = true$ [Note that this is possible for $\mathsf{depth}_{out} \le d'_1$].

Note that:

- When the algorithm ends, it is necessarily because it reached the *halt* instruction in Step 2:

  For every iteration $i = 1, \ldots, l_{in}^* + 1$, let $\mathbf{L}_i$ denote the average number of uncovered assignments $\frac{1}{|\Omega_{out}|} \cdot \sum_{\xi_{out} \in \Omega_{out}} |L_{\xi_{out}}|$ in the i'th iteration. We have that $\mathbf{L}_1 \le l'_{in}$. Moreover, for every iteration $i = 1, \ldots, l_{in}^* + 1$ in which the algorithm does not halt, we have that $\mathbf{L}_{i+1} \le \mathbf{L}_i - \delta \cdot 1$. Hence, if the algorithm reaches Step 2 in iteration $l_{in}^* + 1$, it holds that $\mathbf{L}_{l_{in}^* + 1} \le l'_{in} - \delta \cdot l_{in}^* \le 0$ (recall that $l_{in}^* \ge \frac{2b_1}{\delta^{b_2 + 2}}$, while $l'_{in} \le \frac{2b_1}{\delta^{b_2 + 1}}$). Therefore, in this iteration, for every $\xi_{out} \in \Omega_{out}$ it holds that $|L_{\xi_{out}}| = 0$, and the algorithm must halt.

- Assume that the algorithm halts in Step 2 in the i'th iteration. Pick uniformly at random an edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$ coming out of a_{out} in G_{out} and an edge $e_{in} = (a_{in}, b_{in}) \in E_{in}^{b_{out}}$. Set $e = (a, b) \in E$ for $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$. Let $\xi_{out} = label_{\mathcal{G}_{out}}(e_{out})$. Let us show that the probability that the edge e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, and e_{in} reads an element in $\{ f_{a_{out}, \xi_{out}, 1}, \ldots, f_{a_{out}, \xi_{out}, l'_{in}} \} - \{ \sigma_{a_{out}, 1}(\xi_{out}), \ldots, \sigma_{a_{out}, i-1}(\xi_{out}) \}$ in $\mathcal{G}_{in}^{b_{out}}$ under the assignments $C_{A_{in}, e_{out}}$ and $C_{B_{in}, b_{out}}$, is less than δ:

  Assume that this is not the case. For every $\xi_{out} \in \Omega_{out}$ there is the same number of edges $e_{out} = (a_{out}, b_{out}) \in E_{out}$ coming out of a_{out} in G_{out} with $label_{\mathcal{G}_{out}}(e_{out}) = \xi_{out}$. Hence,

114

in the i'th iteration, there must be a vertex $a = \langle a_{out}, a_{in} \rangle \in A$ that checked an uncovered assignment for at least δ fraction of the $\xi_{out} \in \Omega_{out}$. If this is the case, the algorithm does not halt.

Let $\sigma_{a_{out},1}, \ldots, \sigma_{a_{out},l_{in}^*} : \Omega_{out} \to \mathcal{D}_{1,dec}$ be the assignments the algorithm outputs (If there are less than l_{in}^* assignments, we pad the list arbitrarily with assignments $\sigma_{a_{out}} : \Omega_{out} \to \mathcal{D}_{1,dec}$ with $sat_{\mathcal{G}_{out}}(a_{out}, \sigma_{a_{out}}) = true$). By definition, for every $i \in [l_{in}^*]$, it holds that $sat_{\mathcal{G}_{out}}(a_{out}, \sigma_{a_{out},i}) = true$. Proposition 16.4 now follows from Proposition 16.3. $\qquad\square$

17 The Tree-Path Game

In this section we analyze a two-prover game that we call a *Tree-Path Game*. This analysis allows the composition of RM-RR and RM◊Had-LR construction algorithms.

A Tree-Path Game is defined by the following objects:

1. **Tree.** A rooted tree T on a set of nodes U. We denote the depth of T by d. The depth d should be thought of as some small constant. For $i = 0, \ldots, d$, we let $U_i \subseteq U$ denote the nodes in depth i. All the nodes in U_i have the same number of children, denoted k_i. We assume that the tree is directed from the root to the leaves.

2. **Alphabet.** A finite alphabet R, where nodes in U are assigned values from R.

3. **Code.** An encoding $E : R \to \Sigma^m$ for some alphabet Σ and length m. The encoding corresponds to a code with (relative) distance $1 - \epsilon$ for $0 < \epsilon < 1$.

In the game a verifier interacts with two provers: *the tree prover* $\mathcal{T}$ and *the path prover* $\mathcal{P}$. Both are asked about an assignment to the nodes U. Supposedly, both answer about the same assignment $\sigma : U \to R$.

The tree prover is given a position $i \in [m]$ in an encoding, and outputs for *every* node $u \in U$ a symbol in Σ. The symbol is supposedly the i'th symbol in the encoding $E(\sigma(u))$. We denote the answer of the prover by $\mathcal{T}(i) : U \to \Sigma$. We say that it is *consistent* with an assignment $\sigma : U \to R$, if indeed for every node $u \in U$, it holds that $\mathcal{T}(i)(u) = E(\sigma(u))_i$.

The path prover is given a leaf in the tree $u_d \in U_d$, and outputs assignments $r_0, \ldots, r_d \in R$ for the nodes $u_0, \ldots, u_d \in U$ on the path $u_0 \to \cdots \to u_d$ from the root to the leaf u_d. We denote the answer of the prover by $\mathcal{P}(u_d) : \{0, \ldots, d\} \to R$. We say that it is *consistent* with an assignment $\sigma : U \to R$, if indeed it holds that $\mathcal{P}(u_d)(j) = \sigma(u_j)$ for $j = 0, \ldots, d$.

The verifier in the Tree-Path game picks uniformly at random a question to the tree prover and a question to the path prover and checks their consistency. See Figure 12.

We will prove that for any strategy of the tree and path provers, there exists a short list of possible assignments to the tree $\sigma_1, \ldots, \sigma_l : U \to R$, such that almost surely whenever the provers pass the

Tree-Path$^{\mathcal{T},\mathcal{P}}$:

1. Pick uniformly at random a position $i \in [m]$.

2. Pick uniformly at random a leaf $u_d \in U_d$.

3. Let $u_0, \ldots, u_d \in U$ denote the nodes on the path from the root to u_d. The verifier asks for $\mathcal{T}(i)$ and $\mathcal{P}(u_d)$ and tests that the two provers are consistent on the assignments to the vertices $u_0, \ldots, u_d \in U$, namely, it checks the following equalities:

$$\mathcal{T}(i)(u_0) = E(\mathcal{P}(u_d)(0))_i, \; \cdots, \; \mathcal{T}(i)(u_d) = E(\mathcal{P}(u_d)(d))_i$$

Figure 12: Tree-Path Game Verifier

test, the answer of the path prover is consistent with one of $\sigma_1, \ldots, \sigma_l$. Note: we could have proved a similar assertion about the consistency of the tree prover with $\sigma_1, \ldots, \sigma_l$, but it is not necessary for us.

We use the following notation:

- $\mathsf{TP}^{\mathcal{T},\mathcal{P}}(i, u_d)$: For a position $i \in [m]$ and a leaf $u_d \in U_d$, we let $\mathsf{TP}^{\mathcal{T},\mathcal{P}}(i, u_d)$ be 1, if the verifier's test in the Tree-Path game passes when the verifier asks the tree prover $\mathcal{T}$ question i and the path prover $\mathcal{P}$ question u_d, and 0 otherwise.

- $\mathsf{con}_{\mathcal{P}}(u_d, \sigma)$: For a leaf $u_d \in U_d$ and an assignment $\sigma : U \to R$, we let $\mathsf{con}_{\mathcal{P}}(u_d, \sigma)$ be 1, if $\mathcal{P}(u_d)$ is consistent with σ, and 0 otherwise.

The first proposition shows that from any prover strategies we can extract an assignment $\sigma : U \to R$ to the nodes of the tree. The assignment is such that we can relate the consistency of the path prover with the assignment to the probability that the test passes.

We prove an even more general statement. In this statement, there is a weight function $w : U_d \to [0, 1]$ that assigns each leaf $u_d \in U_d$ a weight $w(u_d)$, and we consider the following:

- The *average success probability* of the test is

$$\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{\mathcal{T},\mathcal{P}}(i, u_d) \cdot w(u_d) \right]$$

- The *average consistency* of the path prover with an assignment $\sigma : U \to R$ is

$$\mathop{\mathbf{E}}_{u_d \in U_d} \left[\mathsf{con}_{\mathcal{P}}(u_d, \sigma) \cdot w(u_d) \right]$$

We relate the average success probability of the test and the average consistency of the path prover with the assignment. The relation we show rapidly deteriorates with the depth d. However, as we think of d as being a small constant, it does not bother us.

Proposition 17.1. *For any Tree-Path game as above, for any leaf weights $w : U_d \to [0, 1]$, for any prover strategies $\mathcal{T}$ and $\mathcal{P}$, there exists an assignment $\sigma : U \to R$, such that*

$$\mathop{\mathbf{E}}_{u_d \in U_d} \left[\mathsf{con}_{\mathcal{P}}(u_d, \sigma) \cdot w(u_d) \right] \geq \left(\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{\mathcal{T}, \mathcal{P}}(i, u_d) \cdot w(u_d) \right] \right)^{2^d} - (2^d - 1) \cdot \epsilon$$

By applying the proposition iteratively, we can find a short list of possible assignments to the tree $\sigma_1, \ldots, \sigma_l : U \to R$, such that almost surely whenever the provers pass the test, the answer of the path prover is consistent with one of $\sigma_1, \ldots, \sigma_l$:

Proposition 17.2. *Consider a Tree-Path game as above. Set $\delta_{min} \doteq 2 \cdot \epsilon^{\frac{1}{2^d}}$ and $l_{max}(\delta) \doteq \frac{2}{\delta^{2^d}}$. For every $\delta \geq \delta_{min}$, for any prover strategies $\mathcal{T}$ and $\mathcal{P}$, there exist $l \leq l_{max}(\delta)$ assignments $\sigma_1, \ldots, \sigma_l : U \to R$, such that the following holds:*

$$\mathop{\Pr}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{\mathcal{T}, \mathcal{P}}(i, u_d) \wedge \forall j \in [l], \ \mathsf{con}_{\mathcal{P}}(u_d, \sigma_j) = 0 \right] < \delta$$

Proof. Fix $\delta \geq \delta_{min}$. We construct assignments $\sigma_1, \ldots, \sigma_l : U \to R$ iteratively. Each time we use Proposition 17.1 to extract an assignment that is consistent with the path prover on a large fraction of the leaves. Then, we set the weights associated with these leaves to 0 to eliminate their contribution to the probability that the test passes, and move to the next iteration to extract another assignment:

1. **for** $u_d \in U_d$, set $w(u_d) \leftarrow 1$

2. $j \leftarrow 1$

3. **while** $\mathbf{E}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{\mathcal{T}, \mathcal{P}}(i, u_d) \cdot w(u_d) \right] \geq \delta$

   (a) Let $\sigma_j : U \to R$ be an assignment that satisfies

   $$\mathop{\mathbf{E}}_{u_d \in U_d} \left[\mathsf{con}_{\mathcal{P}}(u_d, \sigma_j) \cdot w(u_d) \right] \geq \frac{\delta^{2^d}}{2}$$

   [note that such exists by Proposition 17.1 and the choice of $\delta \geq \delta_{min}$]

   (b) $\hat{U}_{d,j} \leftarrow \{ u_d \in U_d \mid \mathsf{con}_{\mathcal{P}}(u_d, \sigma_j) \cdot w(u_d) = 1 \}$

   (c) **for** $u_d \in \hat{U}_{d,j}$, set $w(u_d) \leftarrow 0$

   (d) $j \leftarrow j + 1$

First note that in Step 3, the following holds:

$$\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{\mathcal{T}, \mathcal{P}}(i, u_d) \cdot w(u_d) \right] = \mathop{\Pr}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{\mathcal{T}, \mathcal{P}}(i, u_d) = 1 \wedge \forall 1 \leq j' \leq j - 1, \ \mathsf{con}_{\mathcal{P}}(u_d, \sigma_{j'}) = 0 \right]$$

Thus, when the procedure ends, the assignments $\sigma_1, \ldots, \sigma_{j-1}$ satisfy the statement of the proposition. It remains to argue that the number of assignments we constructed is at most $l_{max}(\delta) = \frac{2}{\delta^{2^d}}$. This follows since (i) in different iterations $j_1 \neq j_2$, we have $\hat{U}_{d,j_1} \cap \hat{U}_{d,j_2} = \phi$; and (ii) in every iteration j, we have $\left| \hat{U}_{d,j} \right| / |U_d| \geq \frac{\delta^{2^d}}{2}$. $\qquad\square$

Let us prove Proposition 17.1:

Proof. (of Proposition 17.1) The proof will be by induction on the depth d of the tree in the Tree-Path game. For $d = 0$, let T consist of a single node $U = \{u_0\}$. Take the assignment $\sigma : U \to R$ that is consistent with the answer of the path prover, i.e., $\sigma(u_0) = \mathcal{P}(u_0)(0)$. Then, $\mathrm{con}_{\mathcal{P}}(u_0, \sigma) \cdot w(u_0) = w(u_0)$, and we are done since $\mathbf{E}_{i \in [m]} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_0) \cdot w(u_0) \right]^{2^0} - (2^0 - 1) \cdot \epsilon \leq w(u_0)$.

Assume that the proposition holds for some natural number $d - 1$, and let us prove that it holds for d. Consider a Tree-Path game as above where the tree has depth d. Let $w : U_d \to [0, 1]$ give leaf weights. Fix prover strategies T and $\mathcal{P}$.

Observe the nodes in depth $d - 1$. Every such node $u_{d-1} \in U_{d-1}$ has k_{d-1} children $u_d \in U_d$ in depth d. We consider the sub-game, in which the question to the path prover is chosen among the children of u_{d-1} in the tree. Denote $u_{d-1} \to u_d$ when u_d is a child of u_{d-1} in the tree. The average success probability of this test is given by:

$$\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d : u_{d-1} \to u_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_d) \cdot w(u_d) \right]$$

Averaging over the $u_{d-1} \in U_{d-1}$ gives:

$$\mathop{\mathbf{E}}_{u_{d-1} \in U_{d-1}} \left[\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d : u_{d-1} \to u_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_d) \cdot w(u_d) \right] \right] = \mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_d) \cdot w(u_d) \right]$$

Or, equivalently,

$$\mathop{\mathbf{E}}_{u_{d-1} \in U_{d-1}, i \in [m]} \left[\mathop{\mathbf{E}}_{u_d \in U_d : u_{d-1} \to u_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_d) \cdot w(u_d) \right] \right] = \mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_d) \cdot w(u_d) \right]$$

Since for any random variable X it holds that $\mathbf{E}\left[X^2\right] \geq (\mathbf{E}\left[X\right])^2$, we have that

$$\mathop{\mathbf{E}}_{u_{d-1} \in U_{d-1}, i \in [m]} \left[\left(\mathop{\mathbf{E}}_{u_d \in U_d : u_{d-1} \to u_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_d) \cdot w(u_d) \right] \right)^2 \right] \geq \left(\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_d) \cdot w(u_d) \right] \right)^2$$

Using the notation $u_{d-1} \to u_d, u_d'$ to indicate that u_d and u_d' are both children of u_{d-1} (possibly the same child) in the tree, we get

$$\mathop{\mathbf{E}}_{u_{d-1} \in U_{d-1}, i \in [m]} \left[\mathop{\mathbf{E}}_{u_d, u_d' \in U_d : u_{d-1} \to u_d, u_d'} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_d) \cdot w(u_d) \cdot \mathsf{TP}^{T,\mathcal{P}}(i, u_d') \cdot w(u_d') \right] \right]$$

$$\geq \left(\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_d) \cdot w(u_d) \right] \right)^2$$

Or, equivalently,

$$\mathop{\mathbf{E}}_{u_{d-1} \in U_{d-1}} \left[\mathop{\mathbf{E}}_{u_d, u_d' \in U_d : u_{d-1} \to u_d, u_d'} \left[\mathop{\mathbf{E}}_{i \in [m]} \left[\mathsf{TP}^{T,\mathcal{P}}(i, u_d) \cdot w(u_d) \cdot \mathsf{TP}^{T,\mathcal{P}}(i, u_d') \cdot w(u_d') \right] \right] \right]$$

$$\geq \left(\mathop{\mathbf{E}}_{i\in[m],u_d\in U_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i,u_d) \cdot w(u_d) \right] \right)^2$$

For a path assignment $p : \{0,\ldots,d\} \to R$, we denote the assignment induced on depths 0 to $d-1$ by $p_{|d-1} : \{0,\ldots,d-1\} \to R$ (for $j=0,\ldots,d-1$, $p_{|d-1}(j) = p(j)$). We say that leaves $u_d, u_d' \in U_d$ that have the same father $u_{d-1} \in U_{d-1}$ in the tree T *agree*, if on questions u_d and u_d', the path prover answers the same on the common path from the root to u_{d-1}, i.e., $\mathcal{P}(u_d)_{|d-1} = \mathcal{P}(u_d')_{|d-1}$ (note that the equality is between two *functions*). We let $\mathsf{agr}_{\mathcal{P}}(u_d, u_d')$ be 1 if u_d and u_d' agree, and 0 otherwise.

By the distance of the code corresponding to E, for any leaves $u_d, u_d' \in U_d$ that have the same father $u_{d-1} \in U_{d-1}$ in the tree T but do not agree, i.e., $\mathsf{agr}_{\mathcal{P}}(u_d, u_d') = 0$, it holds that:

$$\mathop{\mathbf{E}}_{i\in[m]} \left[\mathsf{TP}^{T,\mathcal{P}}(i,u_d) \cdot \mathsf{TP}^{T,\mathcal{P}}(i,u_d') \right] \leq \epsilon$$

Hence,

$$\mathop{\mathbf{E}}_{u_{d-1}\in U_{d-1}} \left[\mathop{\mathbf{E}}_{u_d,u_d'\in U_d:u_{d-1}\to u_d,u_d'} \left[\mathop{\mathbf{E}}_{i\in[m]} \left[\mathsf{TP}^{T,\mathcal{P}}(i,u_d) \cdot w(u_d) \cdot \mathsf{TP}^{T,\mathcal{P}}(i,u_d') \cdot w(u_d') \cdot \mathsf{agr}_{\mathcal{P}}(u_d,u_d') \right] \right] \right]$$

$$\geq \left(\mathop{\mathbf{E}}_{i\in[m],u_d\in U_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i,u_d) \cdot w(u_d) \right] \right)^2 - \epsilon$$

Or, equivalently,

$$\mathop{\mathbf{E}}_{u_{d-1}\in U_{d-1}} \left[\mathop{\mathbf{E}}_{i\in[m],u_d,u_d'\in U_d:u_{d-1}\to u_d,u_d'} \left[\mathsf{TP}^{T,\mathcal{P}}(i,u_d) \cdot w(u_d) \cdot \mathsf{TP}^{T,\mathcal{P}}(i,u_d') \cdot w(u_d') \cdot \mathsf{agr}_{\mathcal{P}}(u_d,u_d') \right] \right]$$

$$\geq \left(\mathop{\mathbf{E}}_{i\in[m],u_d\in U_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i,u_d) \cdot w(u_d) \right] \right)^2 - \epsilon$$

Consider the tree T' of depth $d-1$ that is obtained from the tree T by discarding the nodes in depth d. Let $U' = \bigcup_{i=0}^{d-1} U_i$. Let $\mathcal{T}'$ be the tree prover induced by $\mathcal{T}$ for T'. For a path assignment $p : \{0,\ldots,d-1\} \to R$, for a position $i \in [m]$ and a node $u_{d-1} \in U_{d-1}$, let $\mathsf{TP}^{T',p}(i,u_{d-1})$ be 0 or 1, depending on whether the test of the Tree-Path game passes, when the tree is T', the tree prover is $\mathcal{T}'$ and the path prover answers p. Let $\mathsf{In}_{\mathcal{P},u_d}(p)$ be 1 if p is induced by the path assignment to u_d, i.e., $\mathcal{P}(u_d)_{|d-1} = p$, and 0 otherwise.

We have that

$$\mathop{\mathbf{E}}_{u_{d-1}\in U_{d-1}} \left[\sum_{p:\{0,\ldots,d-1\}\to R} \mathop{\mathbf{E}}_{i\in[m],u_d,u_d'\in U_d:u_{d-1}\to u_d,u_d'} \left[\mathsf{TP}^{T',p}(i,u_{d-1}) \cdot \mathsf{In}_{\mathcal{P},u_d}(p) \cdot w(u_d) \cdot \mathsf{In}_{\mathcal{P},u_d'}(p) \cdot w(u_d') \right] \right]$$

$$\geq \left(\mathop{\mathbf{E}}_{i\in[m],u_d\in U_d} \left[\mathsf{TP}^{T,\mathcal{P}}(i,u_d) \cdot w(u_d) \right] \right)^2 - \epsilon$$

119

Stated differently,

$$
\mathop{\mathbf{E}}_{u_{d-1}\in U_{d-1}}\left[\sum_{p:\{0,\ldots,d-1\}\to R}\mathop{\mathbf{E}}_{i\in[m]}\left[\mathsf{TP}^{\mathcal{T}',p}(i,u_{d-1})\right]\cdot\left(\mathop{\mathbf{E}}_{u_d\in U_d:u_{d-1}\to u_d}[\mathsf{In}_{\mathcal{P},u_d}(p)\cdot w(u_d)]\right)^2\right]
$$

$$
\geq\left(\mathop{\mathbf{E}}_{i\in[m],u_d\in U_d}\left[\mathsf{TP}^{\mathcal{T},\mathcal{P}}(i,u_d)\cdot w(u_d)\right]\right)^2-\epsilon
$$

We define a path prover $\mathcal{P}'$ for the Tree-Path game on T' as follows: For every node $u_{d-1}\in U_{d-1}$, let $\mathcal{P}'(u_{d-1})$ be an assignment $p:\{0,\ldots,d-1\}\to R$ that maximizes

$$
\mathop{\mathbf{E}}_{i\in[m]}\left[\mathsf{TP}^{\mathcal{T}',p}(i,u_{d-1})\right]\cdot\mathop{\mathbf{E}}_{u_d\in U_d:u_{d-1}\to u_d}[\mathsf{In}_{\mathcal{P},u_d}(p)\cdot w(u_d)]
$$

For $u_{d-1}\in U_{d-1}$, denote this maximum by $M_{u_{d-1}}$. Then, it follows from what we showed that

$$
\mathop{\mathbf{E}}_{u_{d-1}\in U_{d-1}}\left[M_{u_{d-1}}\cdot\sum_{p:\{0,\ldots,d-1\}\to R}\mathop{\mathbf{E}}_{u_d\in U_d:u_{d-1}\to u_d}[\mathsf{In}_{\mathcal{P},u_d}(p)\cdot w(u_d)]\right]
$$

$$
\geq\left(\mathop{\mathbf{E}}_{i\in[m],u_d\in U_d}\left[\mathsf{TP}^{\mathcal{T},\mathcal{P}}(i,u_d)\cdot w(u_d)\right]\right)^2-\epsilon
$$

Since for every $u_{d-1}\in U_{d-1}$ it holds that $\sum_{p:\{0,\ldots,d-1\}\to R}\mathbf{E}_{u_d\in U_d:u_{d-1}\to u_d}[\mathsf{In}_{\mathcal{P},u_d}(p)\cdot w(u_d)]\leq 1$, we have

$$
\mathop{\mathbf{E}}_{u_{d-1}\in U_{d-1}}\left[M_{u_{d-1}}\right]\geq\left(\mathop{\mathbf{E}}_{i\in[m],u_d\in U_d}\left[\mathsf{TP}^{\mathcal{T},\mathcal{P}}(i,u_d)\cdot w(u_d)\right]\right)^2-\epsilon
$$

Or, equivalently,

$$
\mathop{\mathbf{E}}_{i\in[m],u_{d-1}\in U_{d-1}}\left[\mathsf{TP}^{\mathcal{T}',\mathcal{P}'}(i,u_{d-1})\cdot\mathop{\mathbf{E}}_{u_d\in U_d:u_{d-1}\to u_d}[\mathsf{In}_{\mathcal{P},u_d}(\mathcal{P}'(u_{d-1}))\cdot w(u_d)]\right]
$$

$$
\geq\left(\mathop{\mathbf{E}}_{i\in[m],u_d\in U_d}\left[\mathsf{TP}^{\mathcal{T},\mathcal{P}}(i,u_d)\cdot w(u_d)\right]\right)^2-\epsilon \tag{3}
$$

Define a weight function $w':U_{d-1}\to[0,1]$ by assigning every $u_{d-1}\in U_{d-1}$ weight

$$
w'(u_{d-1})\doteq\mathop{\mathbf{E}}_{u_d\in U_d:u_{d-1}\to u_d}[\mathsf{In}_{\mathcal{P},u_d}(\mathcal{P}'(u_{d-1}))\cdot w(u_d)]
$$

By the induction hypothesis on the Tree-Path game on the tree T' of depth $d-1$, we get that there exists an assignment $\sigma':U'\to R$, such that

$$
\mathop{\mathbf{E}}_{u_{d-1}\in U_{d-1}}[\mathsf{con}_{\mathcal{P}'}(u_{d-1},\sigma')\cdot w'(u_{d-1})]
$$

$$
\geq\left(\mathop{\mathbf{E}}_{i\in[m],u_{d-1}\in U_{d-1}}\left[\mathsf{TP}^{\mathcal{T}',\mathcal{P}'}(i,u_{d-1})\cdot w'(u_{d-1})\right]\right)^{2^{d-1}}-(2^{d-1}-1)\cdot\epsilon \tag{4}
$$

Let $\sigma : U \to R$ be the assignment that identifies with σ' on U' and assigns each $u_d \in U_d$ the value $\mathcal{P}(u_d)(d)$. Then,

$$\mathop{\mathbf{E}}_{u_d \in U_d} \left[\mathsf{con}_{\mathcal{P}}(u_d, \sigma) \cdot w(u_d) \right] \geq \mathop{\mathbf{E}}_{u_{d-1} \in U_{d-1}} \left[\mathsf{con}_{\mathcal{P}'}(u_{d-1}, \sigma') \cdot w'(u_{d-1}) \right] \tag{5}$$

Let us lower bound the right hand side of inequality (4). By inequality (3),

$$\left(\mathop{\mathbf{E}}_{i \in [m], u_{d-1} \in U_{d-1}} \left[\mathsf{TP}^{\mathcal{T}', \mathcal{P}'}(i, u_{d-1}) \cdot w'(u_{d-1}) \right] \right)^{2^{d-1}} - (2^{d-1} - 1) \cdot \epsilon$$

$$\geq \left(\left(\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{\mathcal{T}, \mathcal{P}}(i, u_d) \cdot w(u_d) \right] \right)^2 - \epsilon \right)^{2^{d-1}} - (2^{d-1} - 1) \cdot \epsilon$$

$$\geq \left(\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{\mathcal{T}, \mathcal{P}}(i, u_d) \cdot w(u_d) \right] \right)^{2^d} - 2^{d-1} \cdot \epsilon - (2^{d-1} - 1) \cdot \epsilon$$

$$= \left(\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{\mathcal{T}, \mathcal{P}}(i, u_d) \cdot w(u_d) \right] \right)^{2^d} - (2^d - 1) \cdot \epsilon$$

Therefore, from inequality (4) and equality (5), we get

$$\mathop{\mathbf{E}}_{u_d \in U_d} \left[\mathsf{con}_{\mathcal{P}}(u_d, \sigma) \cdot w(u_d) \right] \geq \left(\mathop{\mathbf{E}}_{i \in [m], u_d \in U_d} \left[\mathsf{TP}^{\mathcal{T}, \mathcal{P}}(i, u_d) \cdot w(u_d) \right] \right)^{2^d} - (2^d - 1) \cdot \epsilon$$

The inductive claim follows. $\qquad\square$

18 Composition of Reed-Muller Right Reader and RM⋄Had Left Reader Construction Algorithms

In this section we show how to compose RM-RR and RM⋄Had-LR construction algorithms, thus proving Lemma 8.9.

1. Let $\mathcal{D} = \langle \mathbb{F}^m, \mathbb{F}, \mathcal{D}_{enc}, \mathcal{D}_{dec} \rangle$ be a Reed-Muller domain defined by a finite field $\mathbb{F}$, a dimension m, an encoding degree d and a decoding degree d'.

2. Let $\mathcal{D}_1 = \langle \mathbb{F}^w, \mathbb{F}, \mathcal{D}_{1,enc}, \mathcal{D}_{1,dec} \rangle$ be a Reed-Muller domain defined by the field $\mathbb{F}$, a dimension w, an encoding degree d_1 and a decoding degree d'_1.

3. Let $\mathcal{D}^\diamond = \langle \mathbb{F}^m \times \mathbb{L}^\tau, \mathbb{L}, \mathcal{D}^\diamond_{enc}, \mathcal{D}^\diamond_{dec} \rangle$ be a RM⋄Had domain associated with $\mathcal{D}$, where $\mathbb{L}$ is a subfield of $\mathbb{F}$ and the extension degree is $\tau = [\mathbb{F} : \mathbb{L}]$.

4. Let $\mathcal{D}^\diamond_1 = \langle \mathbb{F}^w \times \mathbb{L}^\tau, \mathbb{L}, \mathcal{D}^\diamond_{1,enc}, \mathcal{D}^\diamond_{1,dec} \rangle$ be a RM⋄Had domain associated with $\mathcal{D}_1$.

Assume that we have outer and inner construction algorithms as follows:

- $\mathcal{A}_{out}$: $(\mathcal{D}, k, N)$-RM-RR construction algorithm with structural parameters $(\mathsf{size}_{out}, \mathsf{block}_{out}, \mathsf{degleft}_{out}, \mathsf{degright}_{out}, \mathsf{depth}_{out})$ reducing $\mathcal{D} \mapsto \mathcal{D}_1$, where the depth depth_{out} is constant and smaller than d_1'.

- $\mathcal{A}_{in}$: $(\mathcal{D}_1^\diamond, \mathsf{degleft}_{out} \cdot k + \mathsf{depth}_{out} + 1, |\mathbb{F}|^{w+1})$-RM$\diamond$Had-LR construction algorithm with structural parameters $(\mathsf{size}_{in}, \mathsf{block}_{in}, \mathsf{degleft}_{in}, \mathsf{degright}_{in})$ that is uniform in the tuple association and in the encoding and list decoding. The algorithm outputs right regular RM$\diamond$Had-LRs.

We design a composed algorithm $\mathcal{A}$. The algorithm $\mathcal{A}$ is a $(\mathcal{D}^\diamond, k, N)$-construction algorithm that outputs edge reading bipartite locally decode/reject codes. The algorithm has structural parameters $(\mathsf{size}, \mathsf{block}, \mathsf{degleft}, \mathsf{degright})$ for $\mathsf{size} \leq \mathsf{size}_{out} \cdot \mathsf{size}_{in}$, $\mathsf{block} \leq \mathsf{degleft}_{out} \cdot \mathsf{block}_{in}$, $\mathsf{degleft} \leq \mathsf{degleft}_{out} \cdot \mathsf{degleft}_{in}$, $\mathsf{degright} \leq \mathsf{degright}_{out} \cdot \mathsf{degright}_{in}$:

Assume that the input to the construction algorithm $\mathcal{A}$ is a collection of k-tuples of points $\langle\langle \vec{x}_{i,1}, \vec{y}_{i,1}\rangle, \ldots, \langle \vec{x}_{i,k}, \vec{y}_{i,k}\rangle\rangle \in (\mathbb{F}^m \times \mathbb{L}^\tau)^k$ for $i \in [N]$. The construction algorithm constructs an edge reading bipartite locally decode/reject code

$$\mathcal{G} = \langle G = (A, B, E), \Omega, \Sigma_A, \Sigma_B, sat_\mathcal{G}, label_\mathcal{G}, proj_\mathcal{G}, tup_\mathcal{G}, eval_\mathcal{G}\rangle$$

as follows:

1. **Outer construction.** Invoke construction algorithm $\mathcal{A}_{out}$ on the input:

$$\langle \vec{x}_{1,1}, \ldots, \vec{x}_{1,k}\rangle, \ldots, \langle \vec{x}_{N,1}, \ldots, \vec{x}_{N,k}\rangle \in (\mathbb{F}^m)^k$$

Obtain an RM-RR

$$\mathcal{G}_{out} = \langle G_{out}, V_{out} = B_{out}, \Omega_{out}, \Sigma_{A_{out}}, \mathcal{D}_1, sat_{\mathcal{G}_{out}}, label_{\mathcal{G}_{out}}, proj_{\mathcal{G}_{out}}, tup_{\mathcal{G}_{out}}, eval_{\mathcal{G}_{out}}\rangle$$

where $G_{out} = (A_{out}, B_{out}, E_{out})$ and $\Sigma_{A_{out}} = \langle \Omega_{out}, \mathcal{D}_{1,dec}, \Sigma_{A_{out},enc}, \Sigma_{A_{out},dec}\rangle$.

2. **Queried points.** Set $k' \doteq (\mathsf{degleft}_{out} \cdot k + \mathsf{depth}_{out} + 1)$. For a vertex-label pair $I = \langle a_{out}, \xi_{out}\rangle \in A_{out} \times \Omega_{out}$ we define a collection of size $|\mathbb{F}|^{w+1}$ of k'-tuples of points in $\mathbb{F}^w \times \mathbb{L}^\tau$. We call these points *queried points*. Each k'-tuple is indexed by a pair $\langle \vec{x}, \vec{y}\rangle \in \mathbb{F}^w \times \mathbb{L}^\tau$. The tuple consists of the following k' points in $\mathbb{F}^w \times \mathbb{L}^\tau$:

   - *Vertex queried points ($\mathsf{degleft}_{out} \cdot k$ points; k points for every neighbor of a_{out}):* These points depend only on the vertex a_{out}. Let $e_{out} = (a_{out}, b_{out}) \in E_{out}$ be an edge coming out of a_{out}. The vertex queried points associated with this edge are as follows. Assume that the tuple associated with b_{out} is $tup_{\mathcal{G}_{out}}(b_{out}) = \langle \vec{x}_{i,1}, \ldots, \vec{x}_{i,k}\rangle$ for $i \in [N]$. If $\vec{p}_1, \ldots, \vec{p}_k \in \mathbb{F}^w$ are the points such that for every vertex $b_{out} \in B_{out}$ and assignment $\sigma_{b_{out}} \in \mathcal{D}_{1,dec}$ it holds that $eval_{\mathcal{G}_{out}}(b_{out}, \sigma_{b_{out}}) = \langle \sigma_{b_{out}}(\vec{p}_1), \ldots, \sigma_{b_{out}}(\vec{p}_k)\rangle$ (such points exist by the definition of RM-RRs), then $\langle \vec{p}_1, \vec{y}_{i,1}\rangle, \ldots, \langle \vec{p}_k, \vec{y}_{i,k}\rangle \in \mathbb{F}^w \times \mathbb{L}^\tau$ are the vertex queried points of $\langle a_{out}, \xi_{out}\rangle$ associated with e_{out}. Note that these points do not depend on $\langle \vec{x}, \vec{y}\rangle$ or on ξ_{out}.

- *Path queried points (*depth$_{\text{out}}$ *points):* Recall that for every vertex $a_{out} \in A_{out}$ there are satisfiability constraints in $\mathcal{G}_{out}$ given by a tree $T_{a_{out}}$ whose leaves are the elements in Ω_{out} and by ancestors point specification functions $\{P_{a_{out},\xi_{out}}\}_{\xi_{out} \in \Omega_{out}}$ (see the discussion in Section 7.5). A pair $\langle \vec{p}, \vec{y} \rangle \in \mathbb{F}^w \times \mathbb{L}^\tau$ is a path queried point of $\langle a_{out}, \xi_{out} \rangle$, if $\vec{p}$ is one of the points along the path from ξ_{out} to the root of the satisfiability tree $T_{a_{out}}$, i.e., there is a depth $i \in \{0, \ldots, \text{depth}_{\text{out}} - 1\}$ such that $P_{a_{out},\xi_{out}}(i) = \vec{p}$.
- *Random point (1 point):* the point $\langle \vec{x}, \vec{y} \rangle \in \mathbb{F}^w \times \mathbb{L}^\tau$.

We assume that each k'-tuple is ordered as above.

3. **Inner construction.** Let $I = \langle a_{out}, \xi_{out} \rangle \in A_{out} \times \Omega_{out}$. We define a RM◇Had-LR whose purpose is to read the queried points of I.

The algorithm $\mathcal{A}_{in}$ is uniform in the tuple association, and, in particular, uniform in structure. Let A_{in}, B_{in}, Ω_{in}, $\Sigma_{A_{in}}$ and $\Sigma_{B_{in}}$ be such that $\mathcal{A}_{in}$ is uniform in structure $\langle A_{in}, B_{in}, V_{in} = A_{in}, \Omega_{in}, \Sigma_{A_{in}}, \Sigma_{B_{in}} \rangle$.

Invoke $\mathcal{A}_{in}$ on the collection of queried points of I. Obtain the RM◇Had-LR $\mathcal{G}^I_{in}$:

$$\mathcal{G}^I_{in} = \langle G^I_{in}, V_{in}, \Omega_{in}, \Sigma_{A_{in}}, \Sigma_{B_{in}}, sat_{\mathcal{G}^I_{in}} \equiv true, label_{\mathcal{G}^I_{in}}, proj_{\mathcal{G}^I_{in}}, tup_{\mathcal{G}^I_{in}}, eval_{\mathcal{G}^I_{in}} \rangle$$

where $G^I_{in} = (A_{in}, B_{in}, E^I_{in})$. Let $tupi_{in} : A_{in} \to \mathbb{F}^w \times \mathbb{L}^\tau$ be the uniform tuple associator (where we use the indexing introduced for the queried points above).

4. **Composed graph.** To construct the composed graph G, for each vertex $a_{out} \in A_{out}$ we produce a copy of A_{in}, and for each vertex $b_{out} \in B_{out}$, we produce a copy of B_{in}. That is, we take:

$$A \doteq \{\langle a_{out}, a_{in} \rangle \mid a_{out} \in A_{out} \wedge a_{in} \in A_{in}\} \quad , \quad B \doteq \{\langle b_{out}, b_{in} \rangle \mid b_{out} \in B_{out} \wedge b_{in} \in B_{in}\}$$

For every edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$ with label $\xi_{out} = label_{\mathcal{G}_{out}}(e_{out})$ and an edge $e_{in} = (a_{in}, b_{in}) \in E^I_{in}$ where $I = \langle a_{out}, \xi_{out} \rangle$, we put an edge $e \in E$ between $\langle a_{out}, a_{in} \rangle \in A$ and $\langle b_{out}, b_{in} \rangle \in B$.

Note that the composed graph is left regular with left degree degleft$_{\text{out}} \cdot$ degleft$_{\text{in}}$ and right regular with right degree degright$_{\text{out}} \cdot$ degright$_{\text{in}}$. The size of the composed graph is less than size$_{\text{out}} \cdot$ size$_{\text{in}}$.

5. **Labels.** We let $\Omega \doteq \Omega_{out} \times \Omega_{in}$. If an edge $e \in E$ corresponds to the outer edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$, labeled by $\xi_{out} = label_{\mathcal{G}_{out}}(e_{out})$, and the inner edge $e_{in} \in E^I_{in}$, where $I = \langle a_{out}, \xi_{out} \rangle$, then $label_G(e)$ is the pair $\langle \xi_{out}, \xi_{in} \rangle \in \Omega$, where ξ_{in} is the label of the inner edge e_{in} in $\mathcal{G}^I_{in}$, i.e., $\xi_{in} = label_{\mathcal{G}^I_{in}}(e_{in})$.

6. **Alphabets.** We let $\Sigma_A = \langle \Omega_{out}, \Sigma_{A_{in},dec}, \Sigma_{A,enc}, \Sigma_{A,dec} \rangle$, where the encoded domain consists of all possible functions from Ω_{out} to the encoded domain of $\Sigma_{A_{in}}$, i.e.,

$$\Sigma_{A,enc} \doteq \{f \mid f : \Omega_{out} \to \Sigma_{A_{in},enc}\}$$

and the decoded domain consists of all possible functions from Ω_{out} to the decoded domain of $\Sigma_{A_{in}}$, i.e.,

$$\Sigma_{A,dec} \doteq \{f \mid f : \Omega_{out} \to \Sigma_{A_{in},dec}\}$$

We let $\Sigma_B \doteq \Sigma_{B_{in}}$. The projection $proj_{\mathcal{G}}$ is defined in the natural way, by assigning every vertex $a = \langle a_{out}, a_{in}\rangle \in A$, assignment $\sigma_a : \Omega_{out} \to \Sigma_{A_{in},dec}$ and label $\xi = \langle \xi_{out}, \xi_{in}\rangle \in \Omega$, where $I = \langle a_{out}, \xi_{out}\rangle$, the projection

$$proj_{\mathcal{G}}(a, \sigma_a, \xi) \doteq proj_{\mathcal{G}_{in}^I}(a_{in}, \sigma_a(\xi_{out}), \xi_{in})$$

Note that the block length is at most $\mathsf{degleft}_{out} \cdot \mathsf{block}_{in}$ (note that we can assume without loss of generality that $|\Omega_{out}| \leq \mathsf{degleft}_{out}$).

7. **Evaluation.** Let $e = (a, b) \in E$ be an edge, where $a = \langle a_{out}, a_{in}\rangle$ and $b = \langle b_{out}, b_{in}\rangle$. Let $e_{out} = (a_{out}, b_{out})$ and let $\xi_{out} = label_{\mathcal{G}_{out}}(e_{out})$ be the label of the outer edge.

   Suppose that $tup_{\mathcal{G}_{out}}(b_{out}) = \langle \vec{x}_{i,1}, \ldots, \vec{x}_{i,k}\rangle \in (\mathbb{F}^m)^k$ for $i \in [N]$ (For simplicity, we associate every b_{out} with a unique $i \in [N]$). We set $tup_{\mathcal{G}}(e) \doteq \langle\langle \vec{x}_{i,1}, \vec{y}_{i,1}\rangle, \ldots, \langle \vec{x}_{i,k}, \vec{y}_{i,k}\rangle\rangle \in (\mathbb{F}^m \times \mathbb{L}^\tau)^k$. Note that each tuple is associated with the same number of edges.

   Let $\sigma_a : \Omega_{out} \to \Sigma_{A_{in},dec}$ and $I = \langle a_{out}, \xi_{out}\rangle$. We let $eval_{\mathcal{G}}(e, \sigma_a)$ be $eval_{\mathcal{G}_{in}^I}(a_{in}, \sigma_a(\xi_{out}))$ truncated to the k positions corresponding to the k vertex queried points associated with e_{out}.

8. **Tree satisfiability constraints.** Recall the definition of tree satisfiability constraints appearing in Section 7.5.

   Let $a = \langle a_{out}, a_{in}\rangle \in A$. Let $\sigma_a : \Omega_{out} \to \Sigma_{A_{in},dec}$ be an assignment for a. We define a satisfiability constraint $sat_{\mathcal{G}}(a, \sigma_a)$, whose purpose is to check the tree satisfiability constraints of a_{out} in $\mathcal{G}_{out}$ (recall that the inner reader has no satisfiability constraints). The check focuses on one position in the Hadamard encoding of the evaluation in each of the nodes. The position is determined by a_{in} (here we use the uniformity in the tuple association).

   Let the tree satisfiability constraints of a_{out} in $\mathcal{G}_{out}$ be given by the tree $T_{a_{out}} = (U_{a_{out}} \cup \Omega_{out}, E_{a_{out}})$ and the ancestors point specification functions $\{P_{a_{out},\xi_{out}}\}_{\xi_{out}\in\Omega_{out}}$.

   We say that σ_a satisfies a (i.e., $sat_{\mathcal{G}}(a, \sigma_a) = true$), if there exists an assignment of elements in $\mathbb{L}$ to the inner nodes of the tree $\sigma : U_{a_{out}} \to \mathbb{L}$, such that the following holds. Let $u \in U_{a_{out}}$ be a node in the tree $T_{a_{out}}$. Assume that u is in depth $h \in \{0, \ldots, \mathsf{depth}_{out} - 1\}$ in the tree. Let $\xi_{out} \in \Omega_{out}$ be a leaf in tree which is a descendent of u. Let $\vec{p} = P_{a_{out},\xi_{out}}(h) \in \mathbb{F}^w$ be the point specified for u in the tree. Let $\vec{y} \in \mathbb{L}^\tau$ be the position in the Hadamard encoding associated with a_{in} (i.e., is the second component of the pair $tupi_{in}(a_{in})$).

   Then, $\langle \vec{p}, \vec{y}\rangle$ is a path queried point of $I = \langle a_{out}, \xi_{out}\rangle$. Suppose it is the j'th queried point for $j \in [k']$. It should hold that $eval_{\mathcal{G}_{in}^I}(a_{in}, \sigma_a(\xi_{out}))_j = \sigma(u)$.

18.1 Analysis

Lemma 18.1 (Composition). *Let $0 < \delta_{min,out}, \delta_{min,in} < 1$. Let $l_{max,out}, l_{max,in} : (0,1) \to \mathbb{R}^+$ be decreasing functions. Assume that $l_{max,in}(\delta), l_{max,out}(\delta) \leq \delta^{-O(1)}$. For a sufficiently small constant $c > 0$, set*

$$\delta_{min} \doteq \max\left\{ \delta^c_{min,in}, \delta^c_{min,out}, \left(\frac{d'}{|\mathbb{F}|}\right)^c, \left(\frac{1}{|\mathbb{L}|}\right)^c \right\}$$

Then, if $\mathcal{A}_{out}$ outputs $(\delta_{min,out}, l_{max,out})$-RM-RRs, and $\mathcal{A}_{in}$ outputs $(\delta_{min,in}, l_{max,in})$-RM$\diamond$Had-LRs, then $\mathcal{A}$ outputs (δ_{min}, l_{max})-edge reading bipartite locally decode/reject codes, for some $l_{max}(\delta) \leq \delta^{-O(1)}$.

Proof. We will prove encoding and list decoding:

Encoding. Let $f \in \mathcal{D}_{enc}$. We efficiently construct assignments $C_A : A \to \Sigma_{A,enc}$ and $C_B : B \to \Sigma_{B,enc}$ as follows: Let $C_{A_{out}} : A_{out} \to \Sigma_{A_{out},enc}$ and $C_{B_{out}} : B_{out} \to \mathcal{D}_{1,enc}$ be the assignments for $\mathcal{G}_{out}$ following from the encoding property for f.

By the uniformity in encoding (and list decoding) of $\mathcal{A}_{in}$, for every $b_{out} \in B_{out}$ we can efficiently construct an assignment $C_{B_{in},b_{out}} : B_{in} \to \Sigma_{B_{in},enc}$ for the concatenation of the Reed-Muller codeword $C_{B_{out}}(b_{out})$ with the Hadamard encoding over $\mathbb{L}$. For every $b = \langle b_{out}, b_{in} \rangle \in B$, let $C_B(b) \doteq C_{B_{in},b_{out}}(b_{in})$.

For $I = \langle a_{out}, \xi_{out} \rangle \in A_{out} \times \Omega_{out}$, let $C_{A_{in},I} : A_{in} \to \Sigma_{A_{in},enc}$ be the assignment for A_{in} in $\mathcal{G}^I_{in}$ following from the encoding property for the concatenation of $proj_{\mathcal{G}_{out}}(a_{out}, C_{A_{out}}(a_{out}), \xi_{out}) \in \mathcal{D}_{1,enc}$ with the Hadamard encoding over $\mathbb{L}$.

Let $a = \langle a_{out}, a_{in} \rangle \in A$. Then, $C_A(a)$ is taken to be the function $\sigma_a : \Omega_{out} \to \Sigma_{A_{in},enc}$ defined as follows: For $\xi_{out} \in \Omega_{out}$, let $I = \langle a_{out}, \xi_{out} \rangle$ and define $\sigma_a(\xi_{out}) \doteq C_{A_{in},I}(a_{in})$.

Let $e = (a, b) \in E$ for $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$. Denote the label of e by $label(e) = \xi = \langle \xi_{out}, \xi_{in} \rangle \in \Omega$. Let $I = \langle a_{out}, \xi_{out} \rangle \in A_{out} \times \Omega_{out}$. Let $e_{out} = (a_{out}, b_{out}) \in E_{out}$ and $e_{in} = (a_{in}, b_{in}) \in E^I_{in}$. We have the following properties:

Reading. Assume that $tup_{\mathcal{G}}(e) = \langle \langle \vec{x}_{i,1}, \vec{y}_{i,1} \rangle, \ldots, \langle \vec{x}_{i,k}, \vec{y}_{i,k} \rangle \rangle$ for $i \in [N]$.

We have that $eval_{\mathcal{G}}(e, C_A(a))$ is the k positions in $eval_{\mathcal{G}^I_{in}}(a_{in}, C_{A_{in},I}(a_{in}))$ corresponding to the k vertex queried points associated with e_{out}. The edge e_{in} reads the concatenation of $C_{B_{out}}(b_{out})$ with the Hadamard encoding over $\mathbb{L}$ in $\mathcal{G}^I_{in}$ under $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$. Thus, for every $j \in [k]$ it holds that $eval_{\mathcal{G}}(e, C_A(a))_j$ is the $\vec{y}_{i,j}$ symbol of $eval_{\mathcal{G}_{out}}(b_{out}, C_{B_{out}}(b_{out}))_j$ with the Hadamard encoding over $\mathbb{L}$. Since e_{out} reads f in $\mathcal{G}_{out}$ under $C_{A_{out}}$ and $C_{B_{out}}$, also e reads the concatenation of f with the Hadamard encoding over $\mathbb{L}$ in $\mathcal{G}$ under C_A and C_B.

Projection. We have projection:

$$proj_{\mathcal{G}}(a, C_A(a), \xi) = proj_{\mathcal{G}^I_{in}}(a_{in}, C_{A_{in},I}(a_{in}), \xi_{in}) = C_{B_{in},b_{out}}(b_{in}) = C_B(b)$$

Let $a = \langle a_{out}, a_{in} \rangle \in A$, and let us show that $sat_{\mathcal{G}}(a, C_A(a)) = true$. Let the position in the Hadamard encoding associated with a_{in} be $\vec{y} = (y_1, \ldots, y_\tau) \in \mathbb{L}^\tau$ (this is the second component of the pair $tupi_{in}(a_{in})$).

Satisfaction. We define an assignment $\sigma : U_{a_{out}} \to \mathbb{L}$ of subfield elements to the inner nodes of $T_{a_{out}}$: Let $\sigma' : U_{a_{out}} \to \mathbb{F}$ be the satisfying assignment to the inner nodes of the satisfiability tree $T_{a_{out}}$ of a_{out} in $\mathcal{G}_{out}$ as follows from $sat_{\mathcal{G}_{out}}(a_{out}, C_{A_{out}}(a_{out})) = true$. Recall that we identify $\mathbb{F}$ with $\mathbb{L}^\tau$. For every node $u \in U_{a_{out}}$ let $\sigma(u)$ be the symbol of the Hadamard encoding of $\sigma'(u)$ determined by a_{in}. That is, if $\sigma'(u) = (\sigma'_1, \ldots, \sigma'_\tau) \in \mathbb{L}^\tau$, then $\sigma(u) \doteq \sum_{i=1}^\tau y_i \cdot \sigma'_i$.

Next we show that this assignment is indeed satisfying. Let $u \in U_{a_{out}}$ be a node in the tree $T_{a_{out}}$. Assume that u is in depth $h \in \{0, \ldots, \mathsf{depth}_{out} - 1\}$ in the tree. Let $\xi_{out} \in \Omega_{out}$ be a leaf in the tree which is a descendant of u. Let $Q_{\xi_{out}} \equiv C_{A_{out}}(a_{out})(\xi_{out}) \in \mathcal{D}_{1,dec}$ be the assignment for this leaf in the outer RM-RR $\mathcal{G}_{out}$.

Let $\vec{p} = P_{a_{out}, \xi_{out}}(h) \in \mathbb{F}^w$ be the point specified for u in the tree. Let $\vec{y} \in \mathbb{L}^\tau$ be the second component of the pair $tupi_{in}(a_{in})$. Assume that $\langle \vec{p}, \vec{y} \rangle$ is the j'th queried point of $I = \langle a_{out}, \xi_{out} \rangle$ for $j \in [k']$. Then, the inner reader indeed evaluates by $eval_{\mathcal{G}_{in}^I}(a_{in}, C_A(a)(\xi_{out}))_j$ the position $\vec{y}$ of the Hadamard encoding over $\mathbb{L}$ of $Q_{\xi_{out}}(\vec{p})$.

List decoding. Let $C_B : B \to \Sigma_{B_{in}, dec}$. Let $\delta < 1$. In the course of the proof we will need various lower bounds on δ. All these lower bounds will be quantities that are $\delta_{min,in}^{\Omega(1)}$, $\delta_{min,out}^{\Omega(1)}$, $\left(\frac{d'}{|\mathbb{F}|} \right)^{\Omega(1)}$ or $\left(\frac{1}{|\mathbb{L}|} \right)^{\Omega(1)}$. We will set δ_{min} as to satisfy all these lower bounds.

We use the list decoding properties of the inner and outer constructions to define a list decoding for the composed construction.

We will set δ_{in} in the sequel in a way that $\delta_{in} \geq \delta^{O(1)}$ would hold. Set δ_{min} such that $\delta_{in} \geq \delta_{min,in}$. Set $l_{in} \doteq \lfloor l_{max,in}(\delta_{in}) \rfloor \leq \delta^{-O(1)}$. Let $b_{out} \in B_{out}$. Let $C_{B_{in}, b_{out}} : B_{in} \to \Sigma_{B_{in}, dec}$ be the assignment induced by C_B defined by letting every $b_{in} \in B_{in}$ be assigned $C_B(\langle b_{out}, b_{in} \rangle)$. Let $f_{b_{out},1}, \ldots, f_{b_{out},l_{in}} \in \mathcal{D}_{1,dec}^\diamond$ be the list decoding guaranteed by the uniform list decoding property of $\mathcal{A}_{in}$ for the assignment $C_{B_{in}, b_{out}}$ and confidence parameter δ_{in} (we pad the list arbitrarily if there are less than l_{in} elements in the list decoding). Define l_{in} assignments $C_{B_{out},1}, \ldots, C_{B_{out},l_{in}} : B_{out} \to \mathcal{D}_{1,dec}$ by assigning, for $i \in [l_{in}]$, every vertex $b_{out} \in B_{out}$ to the Reed-Muller codeword associated with $f_{b_{out},i}$.

We will set δ_{out} in the sequel in a way that $\delta_{out} \geq \delta^{O(1)}$ would hold, and set δ_{min} such that $\delta_{out} \geq \delta_{min,out}$. Set $l_{out} \doteq \lfloor l_{max,out}(\delta_{out}) \rfloor \leq \delta^{-O(1)}$. For every $i \in [l_{in}]$, let $f_{i,1}, \ldots, f_{i,l_{out}} \in \mathcal{D}_{dec}$ be the list decoding guaranteed by the property of the RM-RR $\mathcal{G}_{out}$ for the assignment $C_{B_{out},i}$ and confidence parameter δ_{out} (we pad the list arbitrarily if there are less than l_{out} elements in the list decoding). In total, we defined a list decoding of size $l_{in} \cdot l_{out} \leq \delta^{-O(1)}$.

Fix an assignment $C_A : A \to \Sigma_{A,dec}$. For every $I = \langle a_{out}, \xi_{out} \rangle \in A_{out} \times \Omega_{out}$, the assignment C_A induces an assignment $C_{A_{in},I} : A_{in} \to \Sigma_{A_{in},dec}$ to A_{in} in $\mathcal{G}_{in}^I$: for every vertex $a_{in} \in A_{in}$, take

$C_{A_{in},I}(a_{in}) \doteq C_A(\langle a_{out}, a_{in} \rangle)(\xi_{out})$.

For some $l_{in}^* \leq \delta^{-O(1)}$ defined later, we will construct assignments $C_{A_{out},1}, \ldots, C_{A_{out},l_{in}^*}$: $A_{out} \to \Sigma_{A_{out},dec}$, such that the following holds:

Proposition 18.2 (Target outer assignments). *Pick uniformly at random an edge $e = (a, b) \in E$. Let $a = \langle a_{out}, a_{in} \rangle \in A$, $b = \langle b_{out}, b_{in} \rangle \in B$, $e_{out} = (a_{out}, b_{out}) \in E_{out}$, $\xi_{out} = label_{\mathcal{G}_{out}}(e_{out})$, $I = \langle a_{out}, \xi_{out} \rangle$ and $e_{in} = (a_{in}, b_{in}) \in E_{in}^I$. With probability at least $1 - O(\delta)$:*

Either the edge e is not satisfied in $\mathcal{G}$ under the assignments C_A and C_B, or there are $i_0 \in [l_{in}]$ and $j_0 \in [l_{in}^*]$, for which: (i) the edge e_{in} reads the concatenation of $C_{B_{out},i_0}(b_{out})$ with the Hadamard encoding over $\mathbb{L}$ in $\mathcal{G}_{in}^I$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$; (ii) the edge e_{out} is satisfied in $\mathcal{G}_{out}$ under the assignments C_{A_{out},j_0} and C_{B_{out},i_0}.

We set $\delta_{out} \geq \delta^{O(1)}$ such that $\delta_{out} \cdot l_{in} \cdot l_{in}^* = O(\delta)$. Note that once we prove Proposition 18.2, we are done: Let us use the notation of the proposition. The edge e_{out} is uniformly distributed in E_{out}. Thus, by the list decoding property of the outer construction $\mathcal{G}_{out}$, for every $i_0 \in [l_{in}]$ and $j_0 \in [l_{in}^*]$, the probability of the following event is at most $O(\delta_{out})$: (ii) holds, but not (iii) e_{out} reads one of $f_{i_0,1}, \ldots, f_{i_0,l_{out}}$ in $\mathcal{G}_{out}$ under the assignments C_{A_{out},j_0} and C_{B_{out},i_0}. Hence, the probability that this event happens for *some* $i_0 \in [l_{in}]$ and $j_0 \in [l_{in}^*]$ is at most $O(\delta_{out} \cdot l_{in} \cdot l_{in}^*) = O(\delta)$. Moreover, whenever both (i) and (iii) hold, in $\mathcal{G}$ under C_A and C_B, the edge e reads the concatenation of one of $f_{i_0,1}, \ldots, f_{i_0,l_{out}}$ with the Hadamard encoding over $\mathbb{L}$.

Constructing the target outer assignments. For every vertex $a_{out} \in A_{out}$, we construct l_{in}^* assignments $\sigma_{a_{out},1}, \ldots, \sigma_{a_{out},l_{in}^*} : \Omega_{out} \to \mathcal{D}_{1,dec}^\diamond$ to a_{out}. We identify an assignment $\sigma : \Omega_{out} \to \mathcal{D}_{1,dec}^\diamond$ with an assignment $\Omega_{out} \to \mathcal{D}_{1,dec}$ that maps each ξ_{out} to the Reed-Muller codeword corresponding to $\sigma(\xi_{out})$.

The assignments $C_{A_{out},1}, \ldots, C_{A_{out},l_{in}^*} : A_{out} \to \Sigma_{A_{out},dec}$ are defined for every $i \in [l_{in}^*]$ by assigning each $a_{out} \in A_{out}$ the function $\Omega_{out} \to \mathcal{D}_{1,dec}$ identified with $\sigma_{a_{out},i}$.

For every vertex $a_{out} \in A_{out}$ we construct the assignments $\sigma_{a_{out},1}, \ldots, \sigma_{a_{out},l_{in}^*}$ in three steps:

1. *Projection step.* Set $l_{in}' \doteq \lfloor \frac{1}{\delta} \cdot l_{max,in}(\delta^2) \rfloor \leq \delta^{-O(1)}$. For every vertex $a_{out} \in A_{out}$, for each label $\xi_{out} \in \Omega_{out}$, we construct a list $f_{a_{out},\xi_{out},1}, \ldots, f_{a_{out},\xi_{out},l_{in}'} \in \mathcal{D}_{1,dec}^\diamond$ of candidates for ξ_{out}. The list satisfies the following property:

   Proposition 18.3. *Let $a_{out} \in A_{out}$. Let $e_{out} = (a_{out}, b_{out}) \in E_{out}$ be an edge coming out of a_{out} that has label $label_{\mathcal{G}_{out}}(e_{out}) = \xi_{out}$. Let $I = \langle a_{out}, \xi_{out} \rangle \in A_{out} \times \Omega_{out}$.*

   When picking uniformly at random an edge $e_{in} = (a_{in}, b_{in}) \in E_{in}^I$ and setting $e = (a, b) \in E$ for $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$, the probability that the following holds is at most $O(\delta)$:

   The edge e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, but e_{in} does not read an element in $\{f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}\} \cap \{f_{a_{out},\xi_{out},1}, \ldots, f_{a_{out},\xi_{out},l_{in}'}\}$ in $\mathcal{G}_{in}^I$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$.

For every $i \in [l'_{in}]$, for every $\xi_{out} \in \Omega_{out}$, define $\sigma_{a_{out},i}(\xi_{out}) \doteq f_{a_{out},\xi_{out},i}$.

2. *Satisfaction step.* Using our analysis for the Tree-Path game, for every vertex $a_{out} \in A_{out}$ we construct the l^*_{in} assignments $\sigma_{a_{out},1}, \ldots, \sigma_{a_{out},l^*_{in}} : \Omega_{out} \to \mathcal{D}^\diamond_{1,dec}$. The assignments correspond to satisfying assignments for a_{out} in $\mathcal{G}_{out}$. In addition, the property of the assignments from the previous step would still hold for them:

Proposition 18.4. *Let $a_{out} \in A_{out}$.*

(a) *For every $i \in [l^*_{in}]$, it holds that $sat_{\mathcal{G}_{out}}(a_{out}, \sigma_{a_{out},i}) = true$ (recall that we identify $\sigma_{a_{out},i}$ with an assignment $\Omega_{out} \to \mathcal{D}_{1,dec}$).*

(b) *Pick uniformly at random an edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$ coming out of a_{out} in $\mathcal{G}_{out}$. Denote its label by $\xi_{out} = label_{\mathcal{G}_{out}}(e_{out})$. Denote $I = \langle a_{out}, \xi_{out} \rangle$. Pick uniformly and independently at random an edge $e_{in} = (a_{in}, b_{in}) \in E^I_{in}$. Set $e = (a, b) \in E$ for $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$. The probability that the following holds is at most $O(\delta)$:*

The edge e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, but e_{in} does not read an element in $\{f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}\} \cap \{\sigma_{a_{out},1}(\xi_{out}), \ldots, \sigma_{a_{out},l^*_{in}}(\xi_{out})\}$ in $\mathcal{G}^I_{in}$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$.

When (in the notation of Proposition 18.4), the edge e_{in} reads an element in $\{f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}\} \cap \{\sigma_{a_{out},1}(\xi_{out}), \ldots, \sigma_{a_{out},l^*_{in}}(\xi_{out})\}$ in $\mathcal{G}^I_{in}$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$, it holds that for some $i_0 \in [l_{in}]$ and $j_0 \in [l^*_{in}]$, the edge e_{in} reads the concatenation of $C_{B_{out},i_0}(b_{out})$ with the Hadamard encoding over $\mathbb{L}$ in $\mathcal{G}^I_{in}$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$, and that the edge e_{out} is satisfied in $\mathcal{G}_{out}$ under C_{A_{out},j_0} and C_{B_{out},i_0}. Thus, once we prove Proposition 18.4, Proposition 18.2 is proved as well, noticing that when $e = (\langle a_{out}, a_{in} \rangle, \langle b_{out}, b_{in} \rangle)$ is uniformly distributed in E, we also have that $e_{out} = (a_{out}, b_{out})$ is uniformly distributed among the edges coming out of a_{out} in $\mathcal{G}_{out}$, and the edge $e_{in} = (a_{in}, b_{in})$ is uniformly distributed in E^I_{in} for $I = \langle a_{out}, \xi_{out} \rangle$.

Let us turn to the construction.

The projection step (Proof of Proposition 18.3). We will apply Proposition 6.11 that shows a list decoding for assignments to the left side of a reader and use the uniformly distributed position read by the inner reader.

Let $a_{out} \in A_{out}$. Let $e_{out} = (a_{out}, b_{out}) \in E_{out}$ be an edge coming out of a_{out} that has label $label_{\mathcal{G}_{out}}(e_{out}) = \xi_{out}$. Let $I = \langle a_{out}, \xi_{out} \rangle \in A_{out} \times \Omega_{out}$. Choose δ_{min} such that $\delta \geq \sqrt{\delta_{min,in}}$. Let $f_{a_{out},\xi_{out},1}, \ldots, f_{a_{out},\xi_{out},l'_{in}} \in \mathcal{D}^\diamond_{1,dec}$ be the elements following from Proposition 6.11 for the assignment $C_{A_{in},I} : A_{in} \to \Sigma_{A_{in},dec}$ and the parameter δ.

Pick uniformly at random an edge $e_{in} = (a_{in}, b_{in}) \in E^I_{in}$ and set $e = (a, b) \in E$ for $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$.

Let us bound the probability that the following bad events happen by $O(\delta)$ and be done:

- **BAD$_1$**: The edge e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, but e_{in} is not satisfied or does not read one of $f_{a_{out},\xi_{out},1}, \ldots, f_{a_{out},\xi_{out},l'_{in}}$ in $\mathcal{G}^I_{in}$ under $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$.

- **BAD$_2$**: The edge e_{in} is satisfied and reads an element from $\{f_{a_{out},\xi_{out},1}, \ldots, f_{a_{out},\xi_{out},l'_{in}}\} - \{f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}\}$ in $\mathcal{G}^I_{in}$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$.

Bounding BAD_1. When e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, we have that e_{in} is satisfied in $\mathcal{G}^I_{in}$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$. The bound follows from Proposition 6.11.

Bounding BAD_2. Let $i_0 \in [l_{in}]$ and $j_0 \in [l'_{in}]$ be such that $f_{a_{out},\xi_{out},j_0} \neq f_{b_{out},i_0}$. The probability that e_{in} reads $f_{a_{out},\xi_{out},j_0}$ and f_{b_{out},i_0} in $\mathcal{G}^I_{in}$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$ is at most $\frac{d'}{|\mathbb{F}|} + \frac{1}{|\mathbb{L}|}$ (since when e_{in} is uniformly distributed in E^I_{in} also the last pair in $tup_{\mathcal{G}^I_{in}}(a_{in})$ is uniformly distributed in $\mathbb{F}^w \times \mathbb{L}^\tau$, and by the distance property of the concatenation of the Reed-Muller code and the Hadamard code). Thus, for every $j_0 \in [l'_{in}]$ such that $f_{a_{out},\xi_{out},j_0} \notin \{f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}\}$, the probability that e_{in} reads $f_{a_{out},\xi_{out},j_0}$, as well as one of $f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}$, in $\mathcal{G}^I_{in}$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$ is at most $l_{in} \cdot \left(\frac{d'}{|\mathbb{F}|} + \frac{1}{|\mathbb{L}|}\right)$.

The probability that e_{in} is satisfied but does not read one of $f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}$ in $\mathcal{G}^I_{in}$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$ is at most $O(\delta_{in})$. In particular, for every $j_0 \in [l'_{in}]$, the probability that e_{in} is satisfied, reads $f_{a_{out},\xi_{out},j_0}$, but does not read one of $f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}$ in $\mathcal{G}^I_{in}$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$ is at most $O(\delta_{in})$.

Hence, the probability that for some $j_0 \in [l'_{in}]$ such that $f_{a_{out},\xi_{out},j_0} \notin \{f_{b_{out},1}, \ldots, f_{b_{out},l_{in}}\}$, the edge e_{in} is satisfied and reads $f_{a_{out},\xi_{out},j_0}$ in $\mathcal{G}^I_{in}$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$, is at most $l'_{in} \cdot \left(l_{in} \cdot \left(\frac{d'}{|\mathbb{F}|} + \frac{1}{|\mathbb{L}|}\right) + O(\delta_{in})\right)$. We set δ_{in} such that $l'_{in} \cdot \delta_{in} \leq \delta$ (this also fixes l_{in}). We set δ_{min} such that $l'_{in} \cdot l_{in} \cdot \left(\frac{d'}{|\mathbb{F}|} + \frac{1}{|\mathbb{L}|}\right) \leq \delta$.

The satisfaction step (Proof of Proposition 18.4). Let $a_{out} \in A_{out}$. Let us define a Tree-Path game corresponding to the constraints on the edges connecting vertices $\langle a_{out}, a_{in} \rangle \in A$ to their neighbors in G.

1. **Tree.** The tree T is $T_{a_{out}} = (U_{a_{out}} \cup \Omega_{out}, E_{a_{out}})$ from the tree satisfiability constraints of a_{out} in $\mathcal{G}_{out}$.

2. **Alphabet.** The range R of the assignments to the nodes of the tree is the field $\mathbb{F}$.

3. **Code.** The encoding of the assignments to the nodes is a repetition of a Hadamard encoding $E : \mathbb{F} \to \mathbb{L}^{|A_{in}|}$ defined as follows: Let $t \in \mathbb{F}$. Let us identify the $|A_{in}|$ positions in $E(t)$ with the vertices in A_{in} and focus on a position $a_{in} \in A_{in}$. Assume that a_{in} is associated with position $\vec{y} \in \mathbb{L}^\tau$ in the Hadamard encoding (this is the case if $\vec{y}$ is the second component in the pair $tupi_{in}(a_{in})$). Then the symbol of $E(t)$ in position a_{in} is the symbol of the Hadamard

encoding of t in position $\vec{y}$, namely $\phi(t)(\vec{y})$ (where ϕ is as in the definition of the domain corresponding to concatenation of Reed-Muller and Hadamard). Note that the code has relative distance $1 - \frac{1}{|\mathbb{L}|}$.

Fix $j \in [l'_{in}]$. Let us define the strategies for the tree prover and the path prover induced by the assignment C_A and the assignment $\sigma_{a_{out},j}$ respectively:

1. **Tree prover.** Assume that the tree prover $\mathcal{T}$ is given a vertex $a_{in} \in A_{in}$ corresponding to a position in the encoding. Denote $a = \langle a_{out}, a_{in} \rangle$.

   - If $sat_{\mathcal{G}}(a, C_A(a)) = false$, then $\mathcal{T}$ outputs an arbitrary assignment to the nodes of the tree $\sigma : U_{a_{out}} \cup \Omega_{out} \to \mathbb{L}$.

   - If $sat_{\mathcal{G}}(a, C_A(a)) = true$, then there is an implied assignment to the inner nodes $\sigma_1 : U_{a_{out}} \to \mathbb{L}$. The tree prover $\mathcal{T}$ outputs an assignment $\sigma : U_{a_{out}} \cup \Omega_{out} \to \mathbb{L}$ that identifies with σ_1 on $U_{a_{out}}$, and assigns 0 to the elements in Ω_{out}.

2. **Path prover.** Let $\{P_{a_{out},\xi_{out}}\}_{\xi_{out} \in \Omega_{out}}$ be the ancestors point specification functions of the tree satisfiability constraints of a_{out} in $\mathcal{G}_{out}$. Fix a leaf $\xi_{out} \in \Omega_{out}$. We say that an assignment $\sigma : \{0, \ldots, \text{depth}_{out}\} \to \mathbb{F}$ is *consistent* with an assignment $\sigma_{a_{out},j}(\xi_{out})$ to ξ_{out}, if the following holds:

   - For $0 \le h \le \text{depth}_{out} - 1$, let $f_{\xi_{out}} \in \mathcal{D}_{1,dec}$ be the Reed-Muller codeword corresponding to $\sigma_{a_{out},j}(\xi_{out}) \in \mathcal{D}^{\diamond}_{1,dec}$. Let $\vec{p} \in \mathbb{F}^w$ be the point associated with the node in depth h, i.e., $\vec{p} \doteq P_{a_{out},\xi_{out}}(h)$. Then, it should hold that $\sigma(h)$ is the evaluation of $f_{\xi_{out}}$ on $\vec{p}$. I.e., $\sigma(h) = f_{\xi_{out}}(\vec{p})$.

   - For $h = \text{depth}_{out}$, it should hold that $\sigma(h) = 0$.

   Assume that the path prover $\mathcal{P}$ is given a leaf $\xi_{out} \in \Omega_{out}$. Then, $\mathcal{P}$ outputs the assignment $\sigma : \{0, \ldots, \text{depth}_{out}\} \to \mathbb{F}$ that is consistent with $\sigma_{a_{out},j}(\xi_{out})$.

Set $\delta_T \doteq \frac{\delta}{l'_{in}} \ge \delta^{O(1)}$. We set δ_{min} such that $\delta_T \ge 2 \cdot \left(\frac{1}{|\mathbb{L}|}\right)^{1/2^{\text{depth}_{out}}}$ (recall that depth_{out} is constant). Let $\sigma_{j,1}, \ldots, \sigma_{j,l} : U_{a_{out}} \cup \Omega_{out} \to \mathbb{F}$ be the $l \le \frac{2}{(\delta_T)^{2^{\text{depth}_{out}}}} \le \delta^{-O(1)}$ assignments to the nodes of the tree guaranteed for δ_T and the strategies of $\mathcal{T}$ and $\mathcal{P}$ by Proposition 17.2. For every $\iota \in [l]$, let $\sigma_{a_{out},j,\iota} : \Omega_{out} \to \mathcal{D}^{\diamond}_{1,dec}$ be defined as follows: let $\xi_{out} \in \Omega_{out}$.

- If $\sigma_{a_{out},j}(\xi_{out})$ is consistent with $\sigma_{j,\iota}$ (that is, consistent with an assignment $\{0, \ldots, \text{depth}_{out}\} \to \mathbb{F}$ that is consistent with $\sigma_{j,\iota}$), let $\sigma_{a_{out},j,\iota}(\xi_{out}) \doteq \sigma_{a_{out},j}(\xi_{out})$.

- Otherwise, let $\sigma_{a_{out},j,\iota}(\xi_{out})$ be an arbitrary element in $\mathcal{D}^{\diamond}_{1,dec}$ that is consistent with $\sigma_{j,\iota}$ (note that such exists since $\text{depth}_{out} \le d'_1$).

For every $\iota \in [l]$, it holds that $sat_{\mathcal{G}_{out}}(a_{out}, \sigma_{a_{out},j,\iota}) = true$ (where we identify $\sigma_{a_{out},j,\iota}$ with an assignment $\Omega_{out} \to \mathcal{D}_{1,dec}$).

Pick uniformly at random an edge $e_{out} = (a_{out}, b_{out}) \in E_{out}$ coming out of a_{out} in $\mathcal{G}_{out}$. Denote its label by $\xi_{out} = label_{\mathcal{G}_{out}}(e_{out})$. Denote $I = \langle a_{out}, \xi_{out} \rangle$. Pick uniformly and independently at random an edge $e_{in} = (a_{in}, b_{in}) \in E_{in}^I$. Set $e = (a, b) \in E$ for $a = \langle a_{out}, a_{in} \rangle \in A$ and $b = \langle b_{out}, b_{in} \rangle \in B$. Note that ξ_{out} is uniformly distributed in Ω_{out} and that a_{in} is uniformly distributed in A_{in}.

When the edge e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B and the edge e_{in} reads $\sigma_{a_{out},j}(\xi_{out})$ in $\mathcal{G}_{in}^I$ under $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$, it holds that the Tree-Path verifier accepts when the tree prover $\mathcal{T}$ gets the position a_{in} and the path prover $\mathcal{P}$ gets the leaf ξ_{out}. In addition, when the answer of $\mathcal{P}$ is consistent with $\sigma_{j,\iota}$ for $\iota \in [l]$, it holds that e_{in} reads $\sigma_{a_{out},j,\iota} = \sigma_{a_{out},j}$ in $\mathcal{G}_{in}^I$ under the assignments $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$. Hence, by Proposition 17.2, the probability that e is satisfied in $\mathcal{G}$ under the assignments C_A and C_B, and e_{in} reads $\sigma_{a_{out},j}(\xi_{out})$ in $\mathcal{G}_{in}^I$ under $C_{A_{in},I}$ and $C_{B_{in},b_{out}}$, but for all $\iota \in [l]$, either $\sigma_{a_{out},j,\iota}(\xi_{out}) \neq \sigma_{a_{out},j}(\xi_{out})$, or the edge e_{in} does not read $\sigma_{a_{out},j,\iota}(\xi_{out}) = \sigma_{a_{out},j}(\xi_{out})$ in $\mathcal{G}_{in}^I$ under the same assignments, is at most $O(\delta_T)$. The probability that this happens for some $j \in [l'_{in}]$ is at most $O(\delta)$.

Let $l_{in}^* \leq \delta^{-O(1)}$ be the total number of assignments we defined. Proposition 18.4 follows from Proposition 18.3. □

Acknowledgments

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[2] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[3] S. Arora and M. Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[4] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 21–32, 1991.

[5] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[6] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.

[7] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactove proofs: How to remove the intractability assumption. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 113 – 131, 1988.

[8] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

[9] E. Ben-Sasson and M. Sudan. Simple PCPs with poly-log rate and query complexity. In *Proc. 37th ACM Symp. on Theory of Computing*, pages 266–275, 2005.

[10] E. Ben-Sasson, M. Sudan, S. P. Vadhan, and A. Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *Proc. 34th ACM Symp. on Theory of Computing*, pages 612–621, 2003.

[11] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.

[12] I. Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

[13] I. Dinur, E. Fischer, G. Kindler, R. Raz, and S. Safra. PCP characterizations of NP: Towards a polynomially-small error-probability. In *Proc. 31st ACM Symp. on Theory of Computing*, pages 29–40, 1999.

[14] I. Dinur and O. Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM Journal on Computing*, 36(4):975–1024, 2006.

[15] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.

[16] U. Feige and J. Kilian. Impossibility results for recycling random bits in two-prover proof systems. In *Proc. 27th ACM Symp. on Theory of Computing*, pages 457–468, 1995.

[17] O. Goldreich and M. Sudan. Locally testable codes and PCPs of almost-linear length. *Journal of the ACM*, 53(4):558–655, 2006.

[18] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[19] J. Håstad and A. Wigderson. Simple analysis of graph tests for linearity and PCP. *Random Structures and Algorithms*, 22(2):139–160, 2003.

[20] T. Holenstein. Parallel repetition: Simplifications and the no-signaling case. In *Proc. 39th ACM Symp. on Theory of Computing*, pages 411–419, 2007.

[21] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proc. 32nd ACM Symp. on Theory of Computing*, pages 80–86, 2000.

[22] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *Proc. 35th ACM Symp. on Theory of Computing*, pages 106–115, 2003.

[23] S. Khot. On the power of unique 2-prover 1-round games. In *Proc. 34th ACM Symp. on Theory of Computing*, pages 767–775, 2002.

[24] S. Khot. Guest column: inapproximability results via long code based PCPs. *SIGACT News*, 36(2):25–42, 2005.

[25] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *IEEE Symposium on Foundations of Computer Science*, pages 2–10, 1990.

[26] D. Moshkovitz and R. Raz. Sub-constant error probabilistically checkable proof of almost-linear size. Technical Report TR07-026, Electronic Colloquium on Computational Complexity, 2007.

[27] D. Moshkovitz and R. Raz. Sub-constant error low degree test of almost-linear size. *SIAM Journal on Computing*, 38(1):140–180, 2008.

[28] A. Rao. Parallel repetition in projection games and a concentration bound. In *Proc. 40th ACM Symp. on Theory of Computing*, 2008.

[29] R. Raz. A parallel repetition theorem. In *SIAM Journal on Computing*, volume 27, pages 763–803, 1998.

[30] R. Raz and S. Safra. A sub-constant error-probability low-degree test and a sub-constant error-probability PCP characterization of NP. In *Proc. 29th ACM Symp. on Theory of Computing*, pages 475–484, 1997.

[31] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

[32] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proc. 32nd ACM Symp. on Theory of Computing*, pages 191–199, 2000.

[33] M. Sudan, L. Trevisan, and S. P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[34] M. Szegedy. Many-valued logics and holographic proofs. In J. Weidermann, P. van Emde Boas, and M. Nielsen, editors, *Automata, Languages and Programming, 26th International Colloquium, ICALP 2007. Lecture notes in Computer Science*, pages 676–686. Springer-Verlag, 1999.

[35] S. Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM*, 55(1):1–16, 2008.

A Explicit Construction of Expanders

We will use the following lemma as our starting point. It gives an explicit construction of expanders with constant degree and constant eigenvalue. For a proof see, for example, Corollary 2.4 in [12]:

Lemma A.1. *There are constants $\Delta_0 > 1$ and $0 < \lambda_0 < \Delta_0$, such that given a natural number n, one can construct in time polynomial in n a Δ_0-regular graph $G = (V, E)$ with $|V| = n$ whose adjacency matrix has second largest eigenvalue (in absolute value) λ_0.*

By raising the graph obtained from Lemma A.1 to a sufficiently large power r, we can get Lemma 5.2:

Corollary 20 (Restatement of Lemma 5.2). *There is a constant $\alpha < 1$ and a function $T : \mathbb{N} \to \mathbb{N}^+$ with $T(\Delta) = \Theta(\Delta)$, such that given two natural numbers n and Δ, one can find in time polynomial in n and in Δ an undirected (multi-)graph $G = (V, E)$ with $|V| = n$, which is $T(\Delta)$-regular and whose adjacency matrix has second largest eigenvalue (in absolute value) $\lambda \leq (T(\Delta))^\alpha$.*

Proof. We use the notation of Lemma A.1. Take $\alpha \doteq \frac{\log \lambda_0}{\log \Delta_0}$. Let $T : \mathbb{N} \to \mathbb{N}^+$ be the function that maps every natural $\Delta < \Delta_0$ to Δ_0 and every natural $\Delta \geq \Delta_0$ to Δ_0^r, where $r \geq 1$ is the natural number that satisfies $\Delta_0^r \leq \Delta < \Delta_0^{r+1}$. Note that $T(\Delta) = \Theta(\Delta)$.

Given natural numbers n and Δ, let $G_0 = (V, E_0)$ be the Δ_0-regular graph with $|V| = n$ obtained from Lemma A.1. Set $r \geq 1$ to be the natural number $\log_{\Delta_0}(T(\Delta))$. Raise G_0 to the power r to obtain a $T(\Delta)$-regular undirected (multi-)graph $G = (V, E)$ (i.e., let G be the (multi-)graph corresponding to raising the adjacency matrix of G_0 to the r'th power). Then, the second largest eigenvalue (in absolute value) of G's adjacency matrix is $\lambda_0^r = (\Delta_0^r)^\alpha = (T(\Delta))^\alpha$. Note that G can be computed in time polynomial in n and in Δ. $\qquad\square$

B Low Degree Testing

The test presented in [27] differs from the variant stated in Subsection 10.1.

Specifically, the test of [27] assumes access to a *deterministic* oracle $\mathcal{A}_0$, assigning polynomials of degree at most d' to three-dimensional *subspaces* in $\mathbb{F}^m$ (and not to their basis representation $\vec{z}, \vec{y}_1, \vec{y}_2$). Supposedly, the polynomials are the restrictions of the tested function f to the subspaces. The notational convention is that for an affine subspace $s \subseteq \mathbb{F}^m$, for a point $\vec{z} \in s$, the polynomial assigned to s evaluated on $\vec{z}$ is denoted $\mathcal{A}_0(s)(\vec{z})$ (this way there is no need to specify the representation of s).

For a function $f : \mathbb{F}^m \to \mathbb{F}$, the test of [27] is as in Figure 13.

We consider a different (yet equivalent) formulation, given in Figure 14.

The following follows from what was shown in [27, 26] (The theorem was essentially proved in [27]. In [26] it was shown that the list decoding does not depend on $\mathcal{A}_0$):

Theorem 21 (Analysis of low degree test, [27, 26]). *For $\delta \geq m \left(\sqrt[8]{\frac{1}{|\mathbb{K}|}} + \sqrt[4]{\frac{md'}{|\mathbb{F}|}} \right)$, for any function $f : \mathbb{F}^m \to \mathbb{F}$, there are $l \leq \frac{2}{\delta}$ polynomials $Q_1, \ldots, Q_l : \mathbb{F}^m \to \mathbb{F}$ of degree at most d', such that*

$LDT_{original}^{f,\mathcal{A}_0}$:

1. Pick uniformly at random three vectors $\langle \vec{z}_0, \vec{y}_1, \vec{y}_2 \rangle \in \mathbb{F}^m \times \mathbb{K}^m \times \mathbb{K}^m$. If $\vec{z}_0, \vec{y}_1, \vec{y}_2$ are linearly dependent, *accept*. Otherwise, let s denote the three-dimensional linear subspace spanned by $\vec{z}_0, \vec{y}_1, \vec{y}_2$.

2. If $\mathcal{A}_0(s)(\vec{z}_0) = f(\vec{z}_0)$, *accept*. Otherwise, *reject*.

Figure 13: The low degree tester of [27]

$LDT_{equiv}^{f,\mathcal{A}_0}$:

1. Pick uniformly at random three vectors $\langle \vec{z}, \vec{y}_1, \vec{y}_2 \rangle \in \mathbb{F}^m \times \mathbb{K}^m \times \mathbb{K}^m$. If $\vec{z}, \vec{y}_1, \vec{y}_2$ are linearly dependent, *accept*. Otherwise, let s denote the three-dimensional linear subspace spanned by $\vec{z}, \vec{y}_1, \vec{y}_2$.

2. Pick uniformly at random $t_0 \neq 0, t_1, t_2 \in \mathbb{F}$. Set $\vec{z}_0 = t_0\vec{z} + t_1\vec{y}_1 + t_2\vec{y}_2$. If $\mathcal{A}_0(s)(\vec{z}_0) = f(\vec{z}_0)$, *accept*. Otherwise, *reject*.

Figure 14: The low degree tester of [27]; different formulation.

for every oracle $\mathcal{A}_0$, the following holds: the probability over the randomness of the tester that $LDT_{equiv}^{f,\mathcal{A}_0}$ accepts, although $f(\vec{z}_0) \notin \{Q_1(\vec{z}_0), \ldots, Q_l(\vec{z}_0)\}$, is at most $O(\delta)$.

Let us show that Theorem 18 from Subsection 10.1 indeed follows from Theorem 21. Recall that in the tester of Theorem 18, we let the answers of the oracle $\mathcal{A}$ depend on the basis representation of s and on additional randomness.

Assume that for a function $f : \mathbb{F}^m \to \mathbb{F}$ and for an oracle $\mathcal{A}$, the probability, over the randomness of $\mathcal{A}$ and the randomness of the tester, that the tester $LDT^{f,\mathcal{A}}$ from Subsection 10.1 accepts, although $f(\vec{z}_0) \notin \{Q_1(\vec{z}_0), \ldots, Q_l(\vec{z}_0)\}$, is $0 \leq \delta' \leq 1$.

Let us probabilistically construct an oracle $\mathcal{A}_0$ for $LDT_{equiv}^{f,\mathcal{A}_0}$. For every subspace s such that there is a positive probability for $s = span\{\vec{z}, \vec{y}_1, \vec{y}_2\}$ in $LDT^{f,\mathcal{A}}$, pick at random a representation $\langle \vec{z}, \vec{y}_1, \vec{y}_2 \rangle \in \mathbb{F}^m \times \mathbb{K}^m \times \mathbb{K}^m$, and a polynomial of degree at most d', according to the distribution of $LDT^{f,\mathcal{A}}$, conditioned on $s = span\{\vec{z}, \vec{y}_1, \vec{y}_2\}$. Let $\mathcal{A}_0(s)$ be this polynomial.

Then, the expectation (over the randomness in the construction of $\mathcal{A}_0$) of the probability, over the randomness of the tester, that the tester $LDT_{equiv}^{f,\mathcal{A}_0}$ accepts, although $f(\vec{z}_0) \notin \{Q_1(\vec{z}_0), \ldots, Q_l(\vec{z}_0)\}$, is at least $\delta' - O(\frac{1}{|\mathbb{F}|})$. Hence, there exists an oracle $\mathcal{A}_0$, such that the probability that the tester $LDT_{equiv}^{f,\mathcal{A}_0}$ accepts, although $f(\vec{z}_0) \notin \{Q_1(\vec{z}_0), \ldots, Q_l(\vec{z}_0)\}$, is at least $\delta' - O(\frac{1}{|\mathbb{F}|})$.

Theorem 18 follows, noticing that for the choice of parameters made in the construction, it holds that $\frac{1}{|\mathbb{F}|} \leq \delta$.

135

C   Linearity Testing

In this section we show how the linearity testing theorem we stated in Section 11.1 follows from the analysis of the Blum-Luby-Rubinfeld linearity test [11].

We test a folded function $\widetilde{f} : R \to \mathbb{F}$, where $R \subseteq \mathbb{F}^m$ is a set of representatives needed for folding, as in Section 11.1. The folded function defines a function $f : \mathbb{F}^m \to \mathbb{F}$ that respects scalar multiplication. The Blum-Luby-Rubinfeld test is as in Figure 15.

$BLRLinTest^{\widetilde{f}}$:

1. Pick uniformly at random $\vec{z}, \vec{y} \in \mathbb{F}^m$.

2. If $f(\vec{z} + \vec{y}) = f(\vec{z}) + f(\vec{y})$, *accept*. Otherwise, *reject*.

Figure 15: Blum-Luby-Rubinfeld (BLR) linearity test

Define the agreement of a function $\widetilde{f} : R \to \mathbb{F}$ with a linear function by

$$Lin^{\widetilde{f}} \doteq \max_{\vec{a} \in \mathbb{F}^m} \left\{ \Pr_{\vec{z} \in \mathbb{F}^m} \left[f(\vec{z}) = \sum_{i=1}^m a_i z_i \right] \right\}$$

It was shown in [19] that large acceptance probability of the BLR linearity test implies large agreement of the function with a linear function. The same analysis also allows to derive a converse result: a large agreement of the function with a linear function implies a (relatively) large acceptance probability of the BLR tester:

Theorem 22 (Analysis of linearity test [19]). *Let $\widetilde{f} : R \to \mathbb{F}$.*

$$\left(Lin^{\widetilde{f}} \right)^3 \le \Pr \left[BLRLinTest^{\widetilde{f}} \text{ accepts} \right] \le Lin^{\widetilde{f}}$$

For convenience we repeat below the linearity test from Section 11.1.

$LinTest^{\widetilde{f}, \mathcal{A}}$:

1. Pick uniformly at random two vectors $\langle \vec{z}, \vec{y} \rangle \in \mathbb{F}^m \times \mathbb{F}^m$. Using the oracle access to $\mathcal{A}$, obtain a bi-variate linear function $l^*(\mathbf{t_1}, \mathbf{t_2})$ over $\mathbb{F}$ for $\langle \vec{z}, \vec{y} \rangle$ [l^* *is supposedly the restriction* $f(\mathbf{t_1}\vec{z} + \mathbf{t_2}\vec{y})$].

2. Pick uniformly at random $t_1, t_2 \in \mathbb{F}$. Set $\vec{x}_0 = t_1\vec{z} + t_2\vec{y}$. If indeed $l^*(t_1, t_2) = f(\vec{x}_0)$, *accept*. Otherwise, *reject*.

Figure 16: Linearity Tester (Projection form) – a copy of Figure 9

From Theorem 22, we conclude that large acceptance probability of the linearity test in the projection form implies large agreement of the function with a linear function:

Corollary 23. *Let $\widetilde{f} : R \to \mathbb{F}$ and let $\mathcal{A}$ be a probabilistic oracle. Then,*

$$Lin^{\widetilde{f}} \geq \left(\Pr \left[LinTest^{\widetilde{f},\mathcal{A}} \text{ accepts} \right] \right)^3 - O \left(\frac{1}{|\mathbb{F}|} \right)$$

Proof. For $\vec{z}, \vec{y} \in \mathbb{F}^m$, let the maximal agreement of f with a linear function within the subspace spanned by $\vec{z}$ and $\vec{y}$ be

$$Lin^{\widetilde{f}}_{\vec{z},\vec{y}} \doteq \max_{\vec{a} \in \mathbb{F}^2} \left\{ \Pr_{\vec{t} \in \mathbb{F}^2} \left[f(t_1\vec{z} + t_2\vec{y}) = a_1 t_1 + a_2 t_2 \right] \right\}$$

Note that

$$\Pr \left[LinTest^{\widetilde{f},\mathcal{A}} \text{ accepts} \right] \leq \mathop{\mathbf{E}}_{\vec{z},\vec{y} \in \mathbb{F}^m} \left[Lin^{\widetilde{f}}_{\vec{z},\vec{y}} \right] \tag{6}$$

Let $R_2 \subseteq \mathbb{F}^2$ be a set of representatives for the equivalence ratio $\sim$ on $\mathbb{F}^2$ (as defined in Section 11.1). For $\vec{z}, \vec{y} \in \mathbb{F}^m$, define $\widetilde{f}_{\vec{z},\vec{y}} : R_2 \to \mathbb{F}$ to be the following restriction of f

$$\widetilde{f}_{\vec{z},\vec{y}}(t_1, t_2) \doteq f(t_1\vec{z} + t_2\vec{y})$$

Let $f_{\vec{z},\vec{y}} : \mathbb{F}^2 \to \mathbb{F}$ be the function defined by $\widetilde{f}_{\vec{z},\vec{y}}$. Note that for every $t_1, t_2 \in \mathbb{F}$, it holds that

$$f_{\vec{z},\vec{y}}(t_1, t_2) = f(t_1\vec{z} + t_2\vec{y})$$

Hence,

$$\Pr \left[BLRLinTest^{\widetilde{f}} \text{ accepts} \right] \geq \mathop{\mathbf{E}}_{\vec{z},\vec{y} \in \mathbb{F}^m} \left[\Pr \left[BLRLinTest^{\widetilde{f}_{\vec{z},\vec{y}}} \text{ accepts} \right] \right] - O \left(\frac{1}{|\mathbb{F}|} \right) \tag{7}$$

In addition, inequality 6 gives

$$\Pr \left[LinTest^{\widetilde{f},\mathcal{A}} \text{ accepts} \right] \leq \mathop{\mathbf{E}}_{\vec{z},\vec{y} \in \mathbb{F}^m} \left[Lin^{\widetilde{f}_{\vec{z},\vec{y}}} \right] \tag{8}$$

By Theorem 22,

$$\mathop{\mathbf{E}}_{\vec{z},\vec{y} \in \mathbb{F}^m} \left[\Pr \left[BLRLinTest^{\widetilde{f}_{\vec{z},\vec{y}}} \text{ accepts} \right] \right] \geq \mathop{\mathbf{E}}_{\vec{z},\vec{y} \in \mathbb{F}^m} \left[\left(Lin^{\widetilde{f}_{\vec{z},\vec{y}}} \right)^3 \right]$$

By Jensen's inequality,

$$\mathop{\mathbf{E}}_{\vec{z},\vec{y} \in \mathbb{F}^m} \left[\Pr \left[BLRLinTest^{\widetilde{f}_{\vec{z},\vec{y}}} \text{ accepts} \right] \right] \geq \left(\mathop{\mathbf{E}}_{\vec{z},\vec{y} \in \mathbb{F}^m} \left[Lin^{\widetilde{f}_{\vec{z},\vec{y}}} \right] \right)^3$$

Thus, applying Theorem 22 again and using inequalities 7 and 8, we get:

$$Lin^{\widetilde{f}} \geq \Pr\left[BLRLinTest^{\widetilde{f}} \text{ accepts}\right]$$

$$\geq \underset{\vec{z},\vec{y}\in\mathbb{F}^m}{\mathbf{E}}\left[\Pr\left[BLRLinTest^{\widetilde{f}_{\vec{z},\vec{y}}} \text{ accepts}\right]\right] - O\left(\frac{1}{|\mathbb{F}|}\right)$$

$$\geq \left(\underset{\vec{z},\vec{y}\in\mathbb{F}^m}{\mathbf{E}}\left[Lin^{\widetilde{f}_{\vec{z},\vec{y}}}\right]\right)^3 - O\left(\frac{1}{|\mathbb{F}|}\right)$$

$$\geq \left(\Pr\left[LinTest^{\widetilde{f},\mathcal{A}} \text{ accepts}\right]\right)^3 - O\left(\frac{1}{|\mathbb{F}|}\right)$$

$\square$

We use the list decoding version of this theorem:

Corollary 24 (Restatement of Theorem 19). *There are some natural m_0 and F_0, such that for every $m \geq m_0$ and prime finite field $\mathbb{F}$ with $|\mathbb{F}| \geq F_0$, the following holds. Let $R \subseteq \mathbb{F}^m$ be the set of representatives needed for folding as in Section 11.1.*

For $\delta \geq 2\sqrt[6]{\frac{1}{|\mathbb{F}|}}$, for any function $\widetilde{f} : R \to \mathbb{F}$, there are $l \leq \frac{2}{\delta^3}$ linear functions $L_1, \ldots, L_l : \mathbb{F}^m \to \mathbb{F}$, such that for every probabilistic oracle $\mathcal{A}$:

The probability, over the randomness of $\mathcal{A}$ and over the randomness of the tester, that $LinTest^{\widetilde{f},\mathcal{A}}$ accepts, although $f(\vec{x}_0) \notin \{L_1(\vec{x}_0), \ldots, L_l(\vec{x}_0)\}$ (where $f : \mathbb{F}^m \to \mathbb{F}$ is the function defined by $\widetilde{f}$ and $\vec{x}_0 \in \mathbb{F}^m$ is picked by the tester; see Figure 9), is at most $O(\delta)$.

Proof. Fix $\delta \geq 2\sqrt[6]{\frac{1}{|\mathbb{F}|}}$ and a function $\widetilde{f} : R \to \mathbb{F}$. Let $f : \mathbb{F}^m \to \mathbb{F}$ be the function defined by $\widetilde{f}$. Set $\delta' \doteq \delta^3$ and note that $\delta' \geq 2\sqrt{\frac{1}{|\mathbb{F}|}}$. Let $L_1, \ldots, L_l : \mathbb{F}^m \to \mathbb{F}$ be the linear functions corresponding to the δ'-list decoding of (the word corresponding to) f with respect to the Hadamard code. Recall that by Proposition 5.4, $l \leq \frac{2}{\delta^3}$. Let $\mathcal{A}$ be a probabilistic oracle.

Assume on way of contradiction that the probability, over the randomness of $\mathcal{A}$ and over the randomness of the tester, that $LinTest^{\widetilde{f},\mathcal{A}}$ accepts, although $f(\vec{z}) \notin \{L_1(\vec{z}), \ldots, L_l(\vec{z})\}$, is more than 2δ. Let $\widetilde{g} : R \to \mathbb{F}^m$ be a function that agrees with $\widetilde{f}$ on all points $\vec{z}$ for which $f(\vec{z}) \notin \{L_1(\vec{z}), \ldots, L_l(\vec{z})\}$. To other points, $\widetilde{g}$ assigns a function $\widetilde{e} : R \to \mathbb{F}$ that rarely agrees with *any* linear function (see Proposition C.1 below):

$$\widetilde{g}(\vec{z}) \doteq \begin{cases} \widetilde{f}(\vec{z}) & f(\vec{z}) \notin \{L_1(\vec{z}), \ldots, L_l(\vec{z})\} \\ \\ \widetilde{e}(\vec{z}) & otherwise \end{cases}$$

Let $g : \mathbb{F}^m \to \mathbb{F}$ be the function defined by $\widetilde{g}$. Note that by our assumption, $LinTest^{\widetilde{g},\mathcal{A}}$ accepts with probability more than 2δ. Hence, by Corollary 23, there is a linear function $L : \mathbb{F}^m \to \mathbb{F}$, such that

$$\Pr_{\vec{z}\in\mathbb{F}^m}[g(\vec{z}) = L(\vec{z})] > (2\delta)^3 - O\left(\frac{1}{|\mathbb{F}|}\right)$$

138

However, by Proposition C.1,

$$\Pr_{\vec{z} \in \mathbb{F}^m} \left[g(\vec{z}) = L(\vec{z}) \wedge \widetilde{g}([\vec{z}]) = \widetilde{e}([\vec{z}]) \right] \leq \frac{1}{\sqrt{|\mathbb{F}|}} \cdot (1 + o(1)) + \frac{1}{|R|}$$

Thus, for sufficiently large m and sufficiently large field $\mathbb{F}$,

$$\Pr_{\vec{z} \in \mathbb{F}^m} \left[g(\vec{z}) = L(\vec{z}) \wedge \widetilde{g}([\vec{z}]) \neq \widetilde{e}([\vec{z}]) \right] > 8\delta^3 - \frac{1}{\sqrt{|\mathbb{F}|}} \cdot (1 + o(1)) \geq \delta' \tag{9}$$

It follows that

$$\Pr_{\vec{z} \in \mathbb{F}^m} \left[f(\vec{z}) = L(\vec{z}) \right] \geq \delta'$$

Therefore, there must be $1 \leq i \leq l$, such that $L \equiv L_i$. On the other hand, returning to inequality 9, we now have

$$\Pr_{\vec{z} \in \mathbb{F}^m} \left[f(\vec{z}) = L_i(\vec{z}) \wedge f(\vec{z}) \notin \{L_1(\vec{z}), \ldots, L_l(\vec{z})\} \right] > 0$$

which is a contradiction. $\qquad\square$

The proposition we needed for the proof is proven below:

Proposition C.1 (Random function rarely agrees with linear). *There is some natural m_0, such that for every $m \geq m_0$, the following holds. There exists a function $\widetilde{e} : R \to \mathbb{F}$ such that for any linear function $L : \mathbb{F}^m \to \mathbb{F}$, when picking uniformly at random $\vec{z} \in R$, the probability that $\widetilde{e}(\vec{z}) = L(\vec{z})$ is at most $\sqrt{\frac{1}{|\mathbb{F}|}} \cdot (1 + o(1))$.*

Proof. Pick a function $\widetilde{e} : R \to \mathbb{F}$ uniformly at random. Let $L : \mathbb{F}^m \to \mathbb{F}$ be a linear function. For every $\vec{z} \in R$, let $X_{L,\vec{z}}$ be an indicator random variable for the event that $\widetilde{e}(\vec{z}) = L(\vec{z})$. For every $\vec{z} \in R$ it holds that $\Pr[X_{L,\vec{z}} = 1] = \frac{1}{|\mathbb{F}|}$. Let $X_L = \sum_{\vec{z} \in R} X_{L,\vec{z}}$. By the linearity of the expectation, it holds that $\mathbf{E}[X_L] = \frac{|R|}{|\mathbb{F}|}$. By the Chernoff bound, for every $\lambda > 0$,

$$\Pr \left[X_L > \mathbf{E}[X_L] + \lambda \right] < \exp \left(-\frac{2\lambda^2}{|R|} \right)$$

In particular, this holds for $\lambda \doteq \sqrt{m |R| \ln |\mathbb{F}|}$. For sufficiently large m, it holds that $\frac{\lambda}{|R|} \leq \sqrt{\frac{1}{|\mathbb{F}|}}$, implying that

$$\Pr \left[\frac{X_L}{|R|} > \sqrt{\frac{1}{|\mathbb{F}|}} + \frac{1}{|\mathbb{F}|} \right] < |\mathbb{F}|^{-2m}$$

Applying a union bound, the probability that for some linear $L : \mathbb{F}^m \to \mathbb{F}$ it holds that $\frac{X_L}{|R|} > \sqrt{\frac{1}{|\mathbb{F}|}} + \frac{1}{|\mathbb{F}|}$ is at most $|\mathbb{F}|^{-m}$. In particular, there exists $\widetilde{e} : R \to \mathbb{F}$ such that for all linear functions $L : \mathbb{F}^m \to \mathbb{F}$ it holds that $\frac{X_L}{|R|} \leq \sqrt{\frac{1}{|\mathbb{F}|}} \cdot (1 + o(1))$. $\qquad\square$