Fast transforms over finite fields of characteristic two

Nicholas Coxon

FAST TRANSFORMS OVER FINITE FIELDS OF CHARACTERISTIC TWO

NICHOLAS COXON

Abstract. An additive fast Fourier transform over a finite field of characteristic two efficiently evaluates polynomials at every element of an $\mathbb{F}_2$-linear subspace of the field. We view these transforms as performing a change of basis from the monomial basis to the associated Lagrange basis, and consider the problem of performing the various conversions between these two bases, the associated Newton basis, and the "novel" basis of Lin, Chung and Han (FOCS 2014). Existing algorithms are divided between two families, those designed for arbitrary subspaces and more efficient algorithms designed for specially constructed subspaces of fields with degree equal to a power of two. We generalise techniques from both families to provide new conversion algorithms that may be applied to arbitrary subspaces, but which benefit equally from the specially constructed subspaces. We then construct subspaces of fields with smooth degree for which our algorithms provide better performance than existing algorithms.

1. Introduction

Let $\mathbb{F}$ be a finite field of characteristic two, and $W = \{\omega_0, \ldots, \omega_{2^n-1}\}$ be an n-dimensional $\mathbb{F}_2$-linear subspace of $\mathbb{F}$. Define polynomials

$$L_i = \prod_{\substack{j=0 \\ j \neq i}}^{2^n-1} \frac{x - \omega_j}{\omega_i - \omega_j}, \quad N_i = \prod_{j=0}^{i-1} \frac{x - \omega_j}{\omega_i - \omega_j} \quad \text{and} \quad X_i = \prod_{k=0}^{n-1} \prod_{j=0}^{2^k[i]_k - 1} \frac{x - \omega_j}{\omega_{2^k[i]_k} - \omega_j}$$

for $i \in \{0, \ldots, 2^n - 1\}$, where $[\,\cdot\,]_k : \mathbb{N} \to \{0, 1\}$ for $k \in \mathbb{N}$ such that

$$i = \sum_{k \in \mathbb{N}} 2^k [i]_k \quad \text{for } i \in \mathbb{N}.$$

Let $\mathbb{F}[x]_\ell$ denote the space of polynomials with coefficients in $\mathbb{F}$ and degree less than ℓ. Then $\{L_0, \ldots, L_{2^n-1}\}$ is the Lagrange basis of $\mathbb{F}[x]_{2^n}$ associated with the enumeration of W. Similarly, $\{N_0, \ldots, N_{2^n-1}\}$ is the associated Newton basis, normalised so that $N_i(\omega_i) = 1$. The definition of the functions $[\,\cdot\,]_k$ implies that each of the polynomials X_i has degree equal to i. Thus, $\{X_0, \ldots, X_{2^n-1}\}$ is also a basis of $\mathbb{F}[x]_{2^n}$. This unusual basis was introduced by Lin, Chung and Han in 2014 [21]. Consequently, we refer to it as the Lin–Chung–Han basis, or simply the LCH basis. In this paper, we describe new fast algorithms for converting between the Lagrange, (normalised) Newton, Lin–Chung–Han and monomial bases for specially constructed subspaces.

Date: July 20, 2018.

2010 *Mathematics Subject Classification.* Primary 68W30, 68W40, 12Y05.

This work was supported by Nokia in the framework of the common laboratory between Nokia Bell Labs and INRIA.

Converting to the Lagrange basis from one of the three remaining bases corresponds to evaluating a polynomial at each element in W. An algorithm that efficiently performs this evaluation for polynomials written on the monomial basis is referred to as an additive fast Fourier transform (FFT). The designation as "additive" reflects the fact that a fast Fourier transform traditionally evaluates polynomials at each element of a cyclic group. To avoid confusion, we refer to such algorithms as multiplicative FFTs hereafter. Additive FFTs have been investigated as an alternative to multiplicative FFTs for use in fast multiplication algorithms for binary polynomials [28, 6, 23, 8, 9, 18], and have also found applications in coding theory and cryptography [4, 3, 10, 1].

Additive FFTs first appeared in the 1980s with the of algorithm of Wang and Zhu [29], which was subsequently rediscovered by Cantor [7]. Applied to characteristic two finite fields, the Wang–Zhu–Cantor algorithm is restricted to extensions of degree equal to a power of two. This restriction is removed by the algorithm of von zur Gathen and Gerhard [28], but at the cost of a higher algebraic complexity. For these and subsequent algorithms [14, 4, 3], the subspace W is described by an ordered basis $\beta = (\beta_0, \ldots, \beta_{n-1})$, which also defines the enumeration of the space by

$$\omega_i = \sum_{k=0}^{n-1} [i]_k \beta_k \quad \text{for } i \in \{0, \ldots, 2^n - 1\}. \tag{1.1}$$

For an arbitrary choice of basis, the algorithm of von zur Gathen and Gerhard performs $\mathcal{O}(\ell \log^2 \ell)$ additions and multiplications, where $\ell = |W| = 2^n$. The subsequent algorithm of Gao and Mateer [14] achieves the same additive complexity while performing only $\mathcal{O}(\ell \log \ell)$ multiplications.

For extensions with degree equal to a power of two, faster algorithms are obtained through a special choice of subspace and its basis. The defining property of these spaces is the existence of a Cantor basis, i.e., a basis $\beta = (\beta_0, \ldots, \beta_{n-1})$ such that

$$\beta_0 = 1 \quad \text{and} \quad \beta_i = \beta_{i+1}^2 - \beta_{i+1} \quad \text{for } i \in \{0, \ldots, n-2\}. \tag{1.2}$$

For subspaces represented by a Cantor basis, the Wang–Zhu–Cantor algorithm performs $\mathcal{O}(\ell \log^{\log_2 3} \ell)$ additions and $\mathcal{O}(\ell \log \ell)$ multiplications. Gao and Mateer [14] also contribute to this special case by providing an algorithm that achieves the same multiplicative complexity while performing only $\mathcal{O}(\ell(\log \ell) \log\log \ell)$ additions.

For the same subspace enumeration used in the additive FFTs, Lin, Chung and Han [21] provide algorithms for converting between the Lagrange and LCH bases that perform $\mathcal{O}(\ell \log \ell)$ additions and multiplications. Lin, Chung and Han use their basis and conversion algorithms to provide fast encoding and decoding algorithms for Reed–Solomon codes. This application is further explored in the subsequent work of Lin, Al-Naffouri and Han [19] and Lin, Al-Naffouri, Han and Chung [20], while Ben-Sasson et al. [2] utilise the conversion algorithms within their zero-knowledge proof system. Lin, Al-Naffouri, Han and Chung [20] additionally consider the problem of converting between the LCH and monomial bases. For subspaces represented by an arbitrary choice of basis, they provide algorithms for converting between the two bases that perform $\mathcal{O}(\ell \log^2 \ell)$ additions and $\mathcal{O}(\ell \log \ell)$ multiplications. For subspaces represented by a Cantor basis they provide algorithms that require only $\mathcal{O}(\ell(\log \ell) \log\log \ell)$ additions and perform no multiplications. The techniques used in both cases, as well as for the algorithms of Lin,

Chung and Han, originate in the work of Gao and Mateer. This relationship becomes apparent when combining the algorithms to obtain additive FFTs, since one essentially obtains the algorithms of Gao and Mateer.

The techniques developed for additive FFTs have yet to be applied to conversions involving the Newton basis. In the realm of multiplicative FFTs, one has the algorithms of van der Hoeven and Schost [26], which convert between the monomial basis and the Newton basis associated with the radix-2 truncated Fourier transform points [25, 24]. Fast conversion between the two basis is a necessary requirement of multivariate evaluation and interpolation algorithms [26, 11] and their application to systematic encoding of Reed–Muller and multiplicity codes [11]. For applications in coding theory, characteristic two finite fields are particularly interesting due to their fast arithmetic. However, the algorithms of van der Hoeven and Schost are not suited to such fields as they require the existence of roots of unity with order equal to a power of two. It is likely that this problem may be partially overcome by generalising their algorithm in a manner analogous to the generalisation of the radix-2 truncated Fourier transform [25, 24] to mixed radices by Larrieu [17]. Developing an algorithm based on the ideas of additive FFTs provides a second and more widely applicable solution to the problem.

In this paper, we describe new fast conversion algorithms between the Lin–Chung–Han basis and each of the Newton, Lagrange and monomial bases. These algorithms may in-turn be combined to obtain fast conversions algorithms between any two of the four bases. We once again represent subspaces by ordered bases, and use (1.1) for their enumeration.

In Section 2, we show that if the defining basis $\beta = (\beta_0, \ldots, \beta_{n-1})$ has dimension greater than one and satisfies

$$1, \frac{\beta_1}{\beta_0}, \ldots, \frac{\beta_{d-1}}{\beta_0} \in \mathbb{F}_{2^d} \tag{1.3}$$

for some $d \in \{1, \ldots, n-1\}$, then each of the three conversions problems over the subspace generated by β may be efficiently reduced to instances of the problem over the subspaces generated by $\alpha = (\beta_0, \ldots, \beta_{d-1})$ and some vector $\delta \in \mathbb{F}^{n-d}$. One may always take d equal to one, allowing the reductions to be applied regardless of the choice of β. Consequently, fast conversions algorithms are obtained by recursively solving the smaller problems admitted by the reduction, and directly solving the problems for the base case of $n = 1$.

Our basis conversion algorithms are described in Section 3. Over the subspace generated by an n-dimensional basis β, the algorithms take as input the first ℓ coefficients on the input basis of a polynomial in $\mathbb{F}[x]_\ell \subseteq \mathbb{F}[x]_{2^n}$. The algorithms then return the first ℓ coefficients of the polynomial on the desired output basis. For conversion between the Lagrange and LCH bases, the first ℓ Lagrange basis polynomials do not form a basis of $\mathbb{F}[x]_\ell$ for $\ell < 2^n$. Consequently, we embed these cases in the larger case of $\ell = 2^n$, after-which we disregard unnecessary parts of the resulting computations so as not to incur a large penalty in complexity. This approach results in what is known as "pruned" or "truncated" algorithms in the literature on fast Fourier transforms. While truncated additive FFTs have been previously investigated [28, 23, 6, 4, 3, 8, 9], our algorithms for converting between the Lagrange and LCH bases are obtained as analogues of Harvey's "cache-friendly" truncated multiplicative FFTs [15]. As a consequence of this approach, the algorithms in fact solve slightly more general problems than those just described,

allowing us to in-turn provide the slightly generalised additive FFTs required by the fast Hermite interpolation and evaluation algorithms of Coxon [12].

Table 1 provides bounds on the number of additions and multiplications performed by our algorithms for conversion in either direction between the LCH basis and each of the three remaining bases. These bounds omit the cost of a small precomputation requiring $\mathcal{O}(n^3)$ field operations. The table also provides bounds on the number of field elements that are required to be stored in auxiliary space by the algorithms. The bound for conversion with the Newton basis is new. For conversion with the Lagrange basis, we only have the algorithm of Lin, Chung and Han [21] to compare with, and only for the case $\ell = 2^n$. Their algorithm performs fewer additions in this case, but only after a much larger precomputation, of unanalysed complexity, that stores $2^n - 1$ field elements in auxiliary space. For conversion with the monomial basis, we have the algorithms of Lin et al. [20] to compare with, but once again only for the case $\ell = 2^n$. Our algorithms perform the same number of additions as their algorithms in this case, while performing fewer multiplications.

Basis	Additions	Multiplications	Auxiliary space
Newton	$\ell(\lceil \log_2 \ell \rceil - 1) + 1$	$\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil$	$\mathcal{O}(n^2)$
Lagrange	$\lfloor \ell/2 \rfloor (3\lceil \log_2 \ell \rceil + 1)$	$\lfloor \ell/2 \rfloor (\lceil \log_2 \ell \rceil + 1)$	$2^n - \ell + \mathcal{O}(n^2)$
Monomial	$\lfloor \ell/2 \rfloor \binom{\lceil \log_2 \ell \rceil}{2}$	$3\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil + 1$	$\mathcal{O}(n)$

TABLE 1. Algebraic and space complexities.

The cost of the reductions used in our algorithms reduce with the size of the value d for which they are applied. For an arbitrary choice of the basis β, the condition (1.3) may only ever be satisfied by d equal to one. This will also be the case if $\mathbb{F}_2$ is the only subfield of $\mathbb{F}$ with degree less than n. The bounds in Table 1 describe the complexity of our algorithms for this case, and thus represent their worst-case complexities. When the field has degree equal to a power of two and β is a Cantor basis, the condition (1.3) is satisfied by d equal to any power of two less than n. Moreover, $\delta = (\beta_0, \ldots, \beta_{n-d-1})$ for all such values of d, so that the recursive cases are themselves represented by Cantor bases. Consequently, it is possible to always take d to be the largest power of two less than n. With this strategy, the algorithms for converting between the LCH and Newton bases perform only $(3\ell - 2)\lceil \log_2 \ell \rceil / 4$ additions. The algorithms for converting between the LCH and Lagrange bases enjoy a similar reduction in the number of additions they perform, while the algorithms for converting between the LCH and monomials bases perform only $\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil \lceil \log_2 \log_2 \max(\ell, 2) \rceil$ additions. Moreover, for conversions between the LCH and monomial bases, all multiplications in the algorithms become multiplications by one, allowing them to be eliminated altogether. In this case, the algorithms reduce to those of Lin et al. [20].

While the benefits of using a Cantor basis are clear, $\mathbb{F}$ will only admit a Cantor basis of a given dimension if its degree is divisible by a sufficiently large power of two. In Section 4, we propose new basis constructions that provide benefits similar to those afforded by Cantor bases when the degree of the field contains a sufficiently large smooth factor, i.e., one that factors into a product of small primes. Such a factor ensures the presence of a tower of subfields, which we use in Section 4.1

to construct bases that reduce the number of field operations performed by the algorithms of Section 3 by allowing their reductions to be applied more frequently with values of d greater than one. For towers containing quadratic extensions, we additionally show in Section 4.2 how to leverage freedom in the construction in order to eliminate some multiplications in the algorithms for conversion between the monomial and LCH bases, echoing the reduction in multiplications obtained for Cantor bases. Finally, in Section 4.3, we show how to take advantage of quadratic extensions in a different manner by generalising the construction of Cantor bases.

2. Preliminaries

For $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{F}^n$, define

$$\omega_{\beta,i} = \sum_{k=0}^{n-1} [i]_k \beta_k \quad \text{for } i \in \{0, \ldots, 2^n - 1\}.$$

If the entries of β are linearly independent over $\mathbb{F}_2$, then let $L_{\beta,i}$, $N_{\beta,i}$ and $X_{\beta,i}$ respectively denote the ith Lagrange, Newton and LCH basis polynomials associated with the enumeration $\{\omega_{\beta,0}, \ldots, \omega_{\beta,2^n-1}\}$ of the subspace it generates. That is, define

$$L_{\beta,i} = \prod_{\substack{j=0 \\ j \neq i}}^{2^n-1} \frac{x - \omega_{\beta,j}}{\omega_{\beta,i} - \omega_{\beta,j}}, \qquad N_{\beta,i} = \prod_{j=0}^{i-1} \frac{x - \omega_{\beta,j}}{\omega_{\beta,i} - \omega_{\beta,j}}$$

and

$$X_{\beta,i} = \prod_{k=0}^{n-1} \prod_{j=0}^{2^k [i]_k - 1} \frac{x - \omega_{\beta,j}}{\omega_{\beta,2^k [i]_k} - \omega_{\beta,j}}$$

for $i \in \{0, \ldots, 2^n - 1\}$.

2.1. Factorisations of basis polynomials.

The following lemma provides factorisations of the basis polynomials associated with a vector $\beta \in \mathbb{F}^n$. These factorisations in-turn provide the reductions employed in our basis conversion algorithms.

Lemma 2.1. *Let $n \geq 2$ and $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{F}^n$ have entries that are linearly independent over $\mathbb{F}_2$. For some $d \in \{1, \ldots, n-1\}$ such that $\beta_i/\beta_0 \in \mathbb{F}_{2^d}$ for $i \in \{0, \ldots, d-1\}$, set $\alpha = (\beta_0, \ldots, \beta_{d-1})$, $\gamma = (\beta_d, \ldots, \beta_{n-1})$ and*

$$\delta = \left(\left(\frac{\beta_d}{\beta_0} \right)^{2^d} - \frac{\beta_d}{\beta_0}, \ldots, \left(\frac{\beta_{n-1}}{\beta_0} \right)^{2^d} - \frac{\beta_{n-1}}{\beta_0} \right).$$

Then δ has entries that are linearly independent over $\mathbb{F}_2$, and

$$L_{\beta,2^d i+j} = L_{\delta,i}\left(\left(\frac{x}{\beta_0} \right)^{2^d} - \frac{x}{\beta_0} \right) L_{\alpha,j}(x - \omega_{\gamma,i}),$$

$$N_{\beta,2^d i+j} = N_{\delta,i}\left(\left(\frac{x}{\beta_0} \right)^{2^d} - \frac{x}{\beta_0} \right) N_{\alpha,j}(x - \omega_{\gamma,i}),$$

$$X_{\beta,2^d i+j} = X_{\delta,i}\left(\left(\frac{x}{\beta_0} \right)^{2^d} - \frac{x}{\beta_0} \right) X_{\alpha,j}(x)$$

for $i \in \{0, \ldots, 2^{n-d} - 1\}$ and $j \in \{0, \ldots, 2^d - 1\}$.

Proof. Let $n \geq 2$ and $\beta = (\beta_0, \dots, \beta_{n-1}) \in \mathbb{F}^n$ have entries that are linearly independent over $\mathbb{F}_2$. Define α, γ and δ as per the lemma for some $d \in \{1, \dots, n-1\}$ such that $\beta_i/\beta_0 \in \mathbb{F}_{2^d}$ for $i \in \{0, \dots, d-1\}$. Then

$$\omega_{\beta, 2^d i + j} = \sum_{k=0}^{d-1} [j]_k \beta_k + \sum_{k=0}^{n-d-1} [i]_k \beta_{d+k} = \omega_{\alpha, j} + \omega_{\gamma, i} \tag{2.1}$$

for $i \in \{0, \dots, 2^{n-d} - 1\}$ and $j \in \{0, \dots, 2^d - 1\}$. The choice of d implies that

$$\left(\frac{\beta_k}{\beta_0}\right)^{2^d} - \frac{\beta_k}{\beta_0} = \prod_{\omega \in \mathbb{F}_{2^d}} \frac{\beta_k}{\beta_0} - \omega = 0 \quad \text{for } k \in \{0, \dots, d-1\}.$$

Thus,

$$\left(\frac{\omega_{\beta, 2^d i + j}}{\beta_0}\right)^{2^d} - \frac{\omega_{\beta, 2^d i + j}}{\beta_0} = \sum_{k=0}^{n-d-1} [i]_k \left(\left(\frac{\beta_{d+k}}{\beta_0}\right)^{2^d} - \frac{\beta_{d+k}}{\beta_0}\right) = \omega_{\delta, i} \tag{2.2}$$

for $i \in \{0, \dots, 2^{n-d} - 1\}$ and $j \in \{0, \dots, 2^d - 1\}$. It follows that

$$\prod_{j=0}^{2^d - 1} x - \omega_{\beta, 2^d i + j} = \beta_0^{2^d} \left(\left(\frac{x}{\beta_0}\right)^{2^d} - \left(\frac{x}{\beta_0}\right) - \omega_{\delta, i}\right) \tag{2.3}$$

for $i \in \{0, \dots, 2^{n-d} - 1\}$, since the polynomials on either side of the equation are monic and split over $\mathbb{F}$ with identical roots. As the entries of β are linearly independent over $\mathbb{F}_2$, comparing roots shows that the polynomials on the left-hand side of (2.3) are distinct for different values of $i \in \{0, \dots, 2^{n-d} - 1\}$. Consequently, comparing the polynomials on the right-hand side of the equation shows that $\omega_{\delta, i} \neq \omega_{\delta, j}$ for distinct $i, j \in \{0, \dots, 2^{n-d} - 1\}$. Thus, the entries of δ are linearly independent over $\mathbb{F}_2$.

Collecting terms and substituting in (2.1), (2.2) and (2.3) shows that

$$\begin{aligned}
L_{\beta, 2^d i + j} &= \left(\prod_{\substack{s=0 \\ s \neq i}}^{2^{n-d} - 1} \prod_{t=0}^{2^d - 1} \frac{x - \omega_{\beta, 2^d s + t}}{\omega_{\beta, 2^d i + j} - \omega_{\beta, 2^d s + t}}\right) \left(\prod_{\substack{t=0 \\ t \neq j}}^{2^d - 1} \frac{x - \omega_{\beta, 2^d i + t}}{\omega_{\beta, 2^d i + j} - \omega_{\beta, 2^d i + t}}\right) \\
&= \left(\prod_{\substack{s=0 \\ s \neq i}}^{2^{n-d} - 1} \frac{(x/\beta_0)^{2^d} - (x/\beta_0) - \omega_{\delta, s}}{\omega_{\delta, i} - \omega_{\delta, s}}\right) \left(\prod_{\substack{t=0 \\ t \neq j}}^{2^d - 1} \frac{x - \omega_{\gamma, i} - \omega_{\alpha, t}}{\omega_{\gamma, i} + \omega_{\alpha, j} - \omega_{\gamma, i} - \omega_{\alpha, t}}\right) \\
&= \left(\prod_{\substack{s=0 \\ s \neq i}}^{2^{n-d} - 1} \frac{(x/\beta_0)^{2^d} - (x/\beta_0) - \omega_{\delta, s}}{\omega_{\delta, i} - \omega_{\delta, s}}\right) \left(\prod_{\substack{t=0 \\ t \neq j}}^{2^d - 1} \frac{x - \omega_{\gamma, i} - \omega_{\alpha, t}}{\omega_{\alpha, j} - \omega_{\alpha, t}}\right) \\
&= L_{\delta, i}\left(\left(\frac{x}{\beta_0}\right)^{2^d} - \frac{x}{\beta_0}\right) L_{\alpha, j}(x - \omega_{\gamma, i})
\end{aligned}$$

for $i \in \{0, \dots, 2^{n-d} - 1\}$ and $j \in \{0, \dots, 2^d - 1\}$.

Similarly,

$$\begin{aligned} N_{\beta,2^d i+j} &= \left(\prod_{s=0}^{i-1}\prod_{t=0}^{2^d-1}\frac{x-\omega_{\beta,2^d s+t}}{\omega_{\beta,2^d i+j}-\omega_{\beta,2^d s+t}}\right)\left(\prod_{t=0}^{j-1}\frac{x-\omega_{\beta,2^d i+t}}{\omega_{\beta,2^d i+j}-\omega_{\beta,2^d i+t}}\right) \\ &= \left(\prod_{s=0}^{i-1}\frac{(x/\beta_0)^{2^d}-(x/\beta_0)-\omega_{\delta,s}}{\omega_{\delta,i}-\omega_{\delta,s}}\right)\left(\prod_{t=0}^{j-1}\frac{x-\omega_{\gamma,i}-\omega_{\alpha,t}}{\omega_{\alpha,j}-\omega_{\alpha,t}}\right) \\ &= N_{\delta,i}\left(\left(\frac{x}{\beta_0}\right)^{2^d}-\frac{x}{\beta_0}\right)N_{\alpha,j}(x-\omega_{\gamma,i}) \end{aligned}$$

for $i \in \{0,\ldots,2^{n-d}-1\}$ and $j \in \{0,\ldots,2^d-1\}$. It follows immediately from the definition of the Newton and LCH bases that

$$X_{\beta,i} = \prod_{k=0}^{n-1} N_{\beta,2^k[i]_k} \quad \text{for } i \in \{0,\ldots,2^n-1\}.$$

Hence,

$$\begin{aligned} X_{\beta,2^d i+j} &= \left(\prod_{k=0}^{n-d-1} N_{\beta,2^{k+d}[i]_k}\right)\left(\prod_{k=0}^{d-1} N_{\beta,2^k[j]_k}\right) \\ &= \left(\prod_{k=0}^{n-d-1} N_{\delta,2^k[i]_k}\left(\left(\frac{x}{\beta_0}\right)^{2^d}-\frac{x}{\beta_0}\right)\right)\left(\prod_{k=0}^{d-1} N_{\alpha,2^k[j]_k}(x-\omega_{\gamma,0})\right) \\ &= X_{\delta,i}\left(\left(\frac{x}{\beta_0}\right)^{2^d}-\frac{x}{\beta_0}\right)X_{\alpha,j}(x) \end{aligned}$$

for $i \in \{0,\ldots,2^{n-d}-1\}$ and $j \in \{0,\ldots,2^d-1\}$. □

To illustrate how we can utilise Lemma 2.1, let us consider the problem of converting polynomials from the Lagrange basis to the LCH basis. The factorisation of the Lagrange basis polynomials provided by the lemma includes a shift of variables for one factor. Consequently, a recursive approach is facilitated by considering the more general problem of converting from a basis of shifted Lagrange polynomials to the LCH basis. An instance of this new problem is defined by a vector $\beta = (\beta_0,\ldots,\beta_{n-1}) \in \mathbb{F}^n$ with linearly independent entries over $\mathbb{F}_2$, a shift parameter $\lambda \in \mathbb{F}$, and coefficients $f_0,\ldots,f_{2^n-1} \in \mathbb{F}$. The goal is then to compute $h_0,\ldots,h_{2^n-1} \in \mathbb{F}$ such that

$$\sum_{i=0}^{2^n-1} h_i X_{\beta,i} = \sum_{i=0}^{2^n-1} f_i L_{\beta,i}(x-\lambda). \tag{2.4}$$

If $n = 1$, then

$$L_{\beta,0} = \frac{x}{\beta_0}+1, \quad L_{\beta,1} = \frac{x}{\beta_0}, \quad X_{\beta,0} = 1 \quad \text{and} \quad X_{\beta,1} = \frac{x}{\beta_0}.$$

Thus, one can simply compute $h_0 = f_0 - (\lambda/\beta_0)(f_0+f_1)$ and $h_1 = f_0+f_1$. If $n \geq 2$, then the following consequence of Lemma 2.1 decomposes the length 2^n problem into 2^{n-d} problems of length 2^d, and 2^d problems of length 2^{n-d}, for $d \in \{1,\ldots,n-1\}$ such that $\beta_i/\beta_0 \in \mathbb{F}_{2^d}$ for $i \in \{0,\ldots,d-1\}$. After choosing such

a value of d, for which one always has the possibility of taking $d = 1$, the smaller instances of the problem admitted by the decomposition can be solved recursively.

Lemma 2.2. *Suppose that $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{F}^n$ has $n \geq 2$ linearly independent entries over $\mathbb{F}_2$, and let α, γ and δ be defined as per Lemma 2.1 for some $d \in \{1, \ldots, n-1\}$ such that $\beta_i/\beta_0 \in \mathbb{F}_{2^d}$ for $i \in \{0, \ldots, d-1\}$. Suppose that $f_0, \ldots, f_{2^n-1}, \lambda, g_0, \ldots, g_{2^n-1}, h_0, \ldots, h_{2^n-1} \in \mathbb{F}$ satisfy*

$$\sum_{j=0}^{2^d-1} g_{2^d i+j} X_{\alpha,j} = \sum_{j=0}^{2^d-1} f_{2^d i+j} L_{\alpha,j}(x - \lambda - \omega_{\gamma,i}) \quad \text{for } i \in \{0, \ldots, 2^{n-d}-1\}, \tag{2.5}$$

and

$$\sum_{i=0}^{2^{n-d}-1} h_{2^d i+j} X_{\delta,i} = \sum_{i=0}^{2^{n-d}-1} g_{2^d i+j} L_{\delta,i}(x - \eta) \quad \text{for } j \in \{0, \ldots, 2^d-1\}, \tag{2.6}$$

where $\eta = (\lambda/\beta_0)^{2^d} - (\lambda/\beta_0)$. Then (2.4) holds.

Proof. Suppose that $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{F}^n$ has $n \geq 2$ linearly independent entries over $\mathbb{F}_2$, and let α, γ and δ be defined as per Lemma 2.1 for some $d \in \{1, \ldots, n-1\}$ such that $\beta_i/\beta_0 \in \mathbb{F}_{2^d}$ for $i \in \{0, \ldots, d-1\}$. Suppose that $f_0, \ldots, f_{2^n-1}, \lambda$, $g_0, \ldots, g_{2^n-1}, h_0, \ldots, h_{2^n-1} \in \mathbb{F}$ satisfy (2.5) and (2.6). Then Lemma 2.1 implies that

$$\sum_{i=0}^{2^n-1} f_i L_{\beta,i}(x - \lambda) = \sum_{i=0}^{2^{n-d}-1} \left(\sum_{j=0}^{2^d-1} f_{2^d i+j} L_{\alpha,j}(x - \lambda - \omega_{\gamma,i}) \right) \times L_{\delta,i}\left(\left(\frac{x}{\beta_0} \right)^{2^d} - \frac{x}{\beta_0} - \eta \right),$$

where $\eta = (\lambda/\beta_0)^{2^d} - (\lambda/\beta_0)$. By substituting in (2.5) and (2.6), it follows that

$$\begin{aligned}
\sum_{i=0}^{2^n-1} f_i L_{\beta,i}(x - \lambda) &= \sum_{i=0}^{2^{n-d}-1} \left(\sum_{j=0}^{2^d-1} g_{2^d i+j} X_{\alpha,j}(x) \right) L_{\delta,i}\left(\left(\frac{x}{\beta_0} \right)^{2^d} - \frac{x}{\beta_0} - \eta \right) \\
&= \sum_{j=0}^{2^d-1} \left(\sum_{i=0}^{2^{n-d}-1} g_{2^d i+j} L_{\delta,i}\left(\left(\frac{x}{\beta_0} \right)^{2^d} - \frac{x}{\beta_0} - \eta \right) \right) X_{\alpha,j}(x) \\
&= \sum_{j=0}^{2^d-1} \left(\sum_{i=0}^{2^{n-d}-1} h_{2^d i+j} X_{\delta,i}\left(\left(\frac{x}{\beta_0} \right)^{2^d} - \frac{x}{\beta_0} \right) \right) X_{\alpha,j}(x).
\end{aligned}$$

Hence, Lemma 2.1 implies that (2.4) holds. □

2.2. **Reduction trees.** We use full binary trees to encode the values of d for which the reductions provided by Lemma 2.1 are applied when converting between two of the bases.

Definition 2.3. A full binary tree is a tree that contains a unique vertex of degree zero or two, while all other vertices have degree one or three. Given a full binary tree (V, E) with unique vertex $r \in V$ of degree zero or two, we use the following nomenclature:

- the vertex r is called the root of the tree,
- a vertex of degree at most one is called a leaf,
- a vertex of degree greater than one is called an internal vertex,
- if $v \in V \setminus \{r\}$ and $v = v_0, v_1, \ldots, v_n = r$ is a path, then v is called a child of v_1, and a descendant of v_i for $i \in \{1, \ldots, n\}$,
- the set of leaves that are descended from or equal to $v \in V$ is denoted L_v,
- the subtree of (V, E) rooted on $v \in V$ is the full binary tree (V', E') such that V' consists of v and all its descendants, and E' consists of all edges $\{u, v\} \in E$ such that $u, v \in V'$.

Example 2.4. The graph shown in Figure 1 is a full binary tree with root vertex v_0, internal vertices v_0, v_1, v_2 and v_6, and leaves v_3, v_4, v_5, v_7 and v_8. The vertex v_1 has children v_2 and v_5, while its descendants are v_2, v_3, v_4 and v_5. Consequently, the subtree rooted on v_1 consists of those vertices and edges contained in the dotted box. Finally, $L_{v_0} = \{v_3, v_4, v_5, v_7, v_8\}$, $L_{v_1} = \{v_3, v_4, v_5\}$, $L_{v_2} = \{v_3, v_4\}$, $L_{v_6} = \{v_7, v_8\}$ and $L_{v_i} = \{v_i\}$ for $i \in \{3, 4, 5, 7, 8\}$.

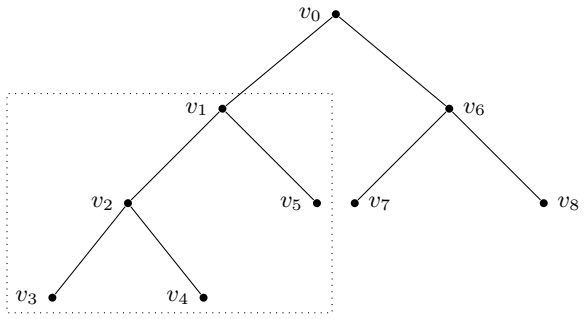

FIGURE 1. A full binary tree.

Each internal vertex of a full binary tree has exactly two children. As it is necessary for us to distinguish between the children of internal vertices in our algorithms, we assume that each full binary tree (V, E) comes equipped with a partition $\{E_\alpha, E_\delta\}$ of E such that if $v \in V$ has children v_0 and v_1, then $\{v, v_i\} \in E_\alpha$ if and only if $\{v, v_{1-i}\} \in E_\delta$. Then for each internal vertex $v \in V$, we denote by v_α the child of v such that $\{v, v_\alpha\} \in E_\alpha$, and by v_δ the child of v such that $\{v, v_\delta\} \in E_\delta$. The partition additionally induces a vertex labelling $d : V \to \mathbb{N}$ given by

$$d(v) = \begin{cases} 0 & \text{if } v \text{ is a leaf vertex,} \\ |L_{v_\alpha}| & \text{if } v \text{ is an internal vertex.} \end{cases}$$

Example 2.5. We obtain one such partition $\{E_\alpha, E_\delta\}$ for the tree shown in Figure 1 by letting E_α contain the leftmost (as shown in the figure), and E_δ the rightmost, of the two edges that connect each internal vertex with its children. Then $d(v_0) = |L_{v_1}| = 3$, $d(v_1) = |L_{v_2}| = 2$, $d(v_2) = |L_{v_3}| = 1$, $d(v_6) = |L_{v_7}| = 1$ and $d(v_i) = 0$ for $i \in \{3, 4, 5, 7, 8\}$.

We use the vertex labelling to encode the values of d for which the reductions provided by Lemma 2.1 are applied. A reduction tree is a full binary tree that fulfils this role for some subspace basis $\beta \in \mathbb{F}^n$. To allow us to formally define this

notion, we now introduce maps that send β to each of the vectors α and δ defined in Lemma 2.1. For $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{F}^n$ with $n \geq 2$ linearly independent entries over $\mathbb{F}_2$ and $d \in \{1, \ldots, n-1\}$, define $\alpha(\beta, d) = (\beta_0, \ldots, \beta_{d-1})$ and

$$\delta(\beta, d) = \left(\left(\frac{\beta_d}{\beta_0}\right)^{2^d} - \frac{\beta_d}{\beta_0}, \ldots, \left(\frac{\beta_{n-1}}{\beta_0}\right)^{2^d} - \frac{\beta_{n-1}}{\beta_0} \right).$$

When d has the additional property that $\beta_i/\beta_0 \in \mathbb{F}_{2^d}$ for $i \in \{0, \ldots, d-1\}$, Lemma 2.1 implies that the vector $\delta(\beta, d)$ has linearly independent entries over $\mathbb{F}_2$.

Definition 2.6. Let $\beta \in \mathbb{F}^n$ have entries that are linearly independent over $\mathbb{F}_2$, and (V, E) be a full binary tree with root vertex $r \in V$. Then (V, E) is a reduction tree for β if it has n leaves, and the following conditions hold if $n > 1$:

(1) $\beta_i/\beta_0 \in \mathbb{F}_{2^{d(r)}}$ for $i \in \{0, \ldots, d(r)-1\}$,
(2) the subtree of (V, E) rooted on r_α is a reduction tree for $\alpha(\beta, d(r))$, and
(3) the subtree of (V, E) rooted on r_δ is a reduction tree for $\delta(\beta, d(r))$.

If v is an internal vertex of a full binary tree, then

$$|L_{v_\alpha}| = d(v) \quad \text{and} \quad |L_{v_\delta}| = |L_v| - |L_{v_\alpha}| = |L_v| - d(v),$$

since $\{L_{v_\alpha}, L_{v_\delta}\}$ is a partition of L_v. Therefore, given a basis vector β and one of its reduction trees (V, E), there exists vectors $\beta_v = (\beta_{v,0}, \ldots, \beta_{v,|L_v|-1}) \in \mathbb{F}^{|L_v|}$ for $v \in V$, each with linearly independent entries over $\mathbb{F}_2$, such that

$$\frac{\beta_{v,0}}{\beta_{v,0}}, \ldots, \frac{\beta_{v,d(v)-1}}{\beta_{v,0}} \in \mathbb{F}_{2^{d(v)}}, \quad \alpha(\beta_v, d(v)) = \beta_{v_\alpha} \quad \text{and} \quad \delta(\beta_v, d(v)) = \beta_{v_\delta}$$

for all internal $v \in V$, and $\beta_v = \beta$ if v is the root of the tree. These vectors define instances of each of the basis conversions problem over the subspaces they generate. If v is an internal vertex, then Lemma 2.1 allows instances of the conversions problems over the subspace generated by β_v to be reduced to instances over the subspaces generated by β_{v_α} and β_{v_δ}. If v is in a leaf, then β_v is 1-dimensional and the corresponding conversion problems may be solved directly. Consequently, the existence of a reduction tree allows the basis conversion problems to be recursively solved with recursion depth equal to that of the tree, i.e., equal to the length of the longest path between its root vertex and one of its leaves.

As the first condition of Definition 2.6 is trivially satisfied if $d(r) = 1$, the existence of a reduction tree for an arbitrarily chosen subspace basis is guaranteed.

Proposition 2.7. *Let $\beta \in \mathbb{F}^n$ have entries that are linearly independent over $\mathbb{F}_2$. Then every full binary tree with n leaves and $\mathrm{Im}(d) \subseteq \{0, 1\}$ is a reduction tree for β.*

It is straightforward to prove Proposition 2.7 by induction on n, or to obtain the proposition as a consequence of Theorem 4.3 in Section 4.1. Consequently, we omit its proof. The choice of reduction trees provided by the proposition captures the strategy used in recent algorithms [14, 21, 20] for an arbitrary choice of subspace basis. Accordingly, we use such trees as our baseline for comparison. The reduction strategy use by recent algorithms specific to Cantor bases [14, 20] is captured by reduction trees such that $d(v) = 2^{\lceil \log_2 |L_v| \rceil - 1}$ for all internal vertices. We characterise the reduction trees of Cantor bases in the following proposition.

Proposition 2.8. *Suppose that $\beta \in \mathbb{F}^n$ is a Cantor basis. Then a full binary tree is a reduction tree for β if and only if it has n leaves and $\mathrm{Im}(d) \subseteq \{0, 2^0, 2^1, 2^2, \ldots\}$.*

We use the following properties of Cantor bases to prove Proposition 2.8.

Lemma 2.9 (Properties of Cantor bases)**.** *Suppose that $\beta = (\beta_0, \dots, \beta_{n-1}) \in \mathbb{F}^n$ is a Cantor basis. Then*

(1) *$\beta_0, \dots, \beta_{n-1}$ are linearly independent over $\mathbb{F}_2$,*
(2) *$\beta_0, \dots, \beta_{i-1} \in \mathbb{F}_{2^{2^k}}$ if $i \leq 2^k$ for some $k \in \mathbb{N}$, and*
(3) *$\beta_i^{2^{2^k}} - \beta_i = \beta_{i-2^k}$ if $i \geq 2^k$ for some $k \in \mathbb{N}$.*

Lemma 2.9 is proved by Gao and Mateer [14, Apendix A], and also obtained as a special case of Lemma 4.13 in Section 4.3.

Proof of Proposition 2.8. A full binary tree with one leaf has $\mathrm{Im}(d) = \{0\}$. Thus, the proposition holds for $n = 1$, since a full binary tree is a reduction tree for a 1-dimensional Cantor basis if and only if it has one leaf. Therefore, let $n \geq 2$, and suppose that the proposition is true for all smaller values of n. Suppose that $\beta = (\beta_0, \dots, \beta_{n-1}) \in \mathbb{F}^n$ is a Cantor basis, and let (V, E) be a full binary tree with root vertex $r \in V$. Then (V, E) is a reduction tree for β if and only if it has n leaves, $\beta_0, \dots, \beta_{d(r)-1} \in \mathbb{F}_{2^{d(r)}}$ (recall that $\beta_0 = 1$ for a Cantor basis), the subtree rooted on r_α is a reduction tree for $\alpha(\beta, d(r))$, and the subtree rooted on r_δ is a reduction tree for $\delta(\beta, d(r))$. If the tree has n leaves, then $1 \leq d(r) < n$ and properties (1) and (2) of Lemma 2.9 imply that $\beta_0, \dots, \beta_{d(r)-1} \in \mathbb{F}_{2^{d(r)}}$ if and only if $d(r)$ is a power of two. In this case, property (3) of the lemma implies that $\alpha(\beta, d(r)) = (\beta_0, \dots, \beta_{d(r)-1})$ and $\delta(\beta, d(r)) = (\beta_0, \dots, \beta_{n-d(r)-1})$. Finally, if the tree has n leaves, then the subtree rooted on r_α has $d(r)$ leaves, and the subtree rooted on r_δ has $|L_r| - d(r) = n - d(r)$ leaves. Therefore, the induction hypothesis implies that (V, E) is a reduction tree for β if and only if it has n leaves, $d(r)$ is a power of two, and the subtrees rooted on r_α and r_δ both satisfy $\mathrm{Im}(d) \subseteq \{0, 2^0, 2^1, 2^2, \dots\}$. That is, if and only if the tree has n leaves and $\mathrm{Im}(d) \subseteq \{0, 2^0, 2^1, 2^2, \dots\}$. Hence, the proposition follows by induction. □

3. Conversion algorithms

In this section, we describe how to convert between the polynomial bases once β and a suitable reduction tree have been chosen. Accordingly, we now fix a vector $\beta = (\beta_0, \dots, \beta_{n-1}) \in \mathbb{F}^n$ that has linearly independent entries over $\mathbb{F}_2$, and a reduction tree (V, E) for β. The algorithms of this section are then presented for this arbitrary selection. Recall that we only provide algorithms for converting between the LCH basis and each of the Newton, Lagrange and monomial bases. The algorithms that we propose for each of these problems require the precomputation of a small number of constants associated with the basis and the reduction tree. Consequently, we begin the section by discussing these precomputations and their cost.

3.1. Precomputations.

For the remainder of the section, we use the shorthand notations $d_v = d(v)$ and $n_v = |L_v|$ for $v \in V$. Define $\beta_v = (\beta_{v,0}, \dots, \beta_{v,n_v-1}) \in \mathbb{F}^{n_v}$ for $v \in V$ recursively as follows: if v is the root of the tree, then $\beta_v = \beta$; if v is an internal vertex, then $\beta_{v_\alpha} = \alpha(\beta_v, d_v)$ and $\beta_{v_\delta} = \delta(\beta_v, d_v)$. Then Definition 2.6 and Lemma 2.1 imply that the vectors β_v for $v \in V$ each have entries that are linearly independent over $\mathbb{F}_2$. Finally, let $\gamma_v = (\beta_{v,d_v}, \dots, \beta_{v,n_v-1})$ for all internal $v \in V$.

Given β_v for some internal $v \in V$, repeated squaring allows $\delta(\beta_v, d_v)$ to be computed with $(n_v - d_v)(d_v + 1)$ multiplications and $n_v - d_v$ additions. If $r \in V$ is the root vertex of the tree, then $V \setminus L_r$ is the set of internal vertices in V, and

$$\sum_{v \in V \setminus L_r} d_v(n_v - d_v) = \sum_{v \in V \setminus L_r} |L_{v_\alpha}||L_{v_\delta}| = \binom{|L_r|}{2} = \binom{n}{2}. \tag{3.1}$$

Thus, the vectors β_v for $v \in V$ can be computed with $\mathcal{O}(n^2)$ field operations. The algorithms we propose for converting between the monomial and LCH bases only require the precomputation and storage of $\beta_{v_\delta,0}$ or $1/\beta_{v_\delta,0}$ for all internal $v \in V$. As a full binary tree with n leaves has $n-1$ internal vertices, the algorithms require the storage of $\mathcal{O}(n)$ field elements, which can be computed with $\mathcal{O}(n^2)$ field operations.

For conversions involving the Lagrange or Newton bases, we generalise the problems to include a shift parameter, as we did at the end of Section 2.1 for Lagrange to LCH conversion. As a result, we require additional machinery to allow computations related to the shift parameter to be handled in a time and space-friendly manner. Define maps $\varphi_v : L_v \times \mathbb{F} \to \mathbb{F}$ for $v \in V$ recursively as follows: for $u \in L_v$ and $\lambda \in \mathbb{F}$,

$$\varphi_v(u, \lambda) = \begin{cases} \lambda/\beta_{v,0} & \text{if } u = v, \\ \varphi_{v_\alpha}(u, \lambda) & \text{if } u \neq v \text{ and } u \in L_{v_\alpha}, \\ \varphi_{v_\delta}\left(u, (\lambda/\beta_{v,0})^{2^{d_v}} - \lambda/\beta_{v,0}\right) & \text{if } u \neq v \text{ and } u \in L_{v_\delta}. \end{cases}$$

For internal $v \in V$, define

$$\sigma_{v,i} = \sum_{j=0}^{i} \beta_{v,d_v+j} \quad \text{for } i \in \{0, \ldots, n_v - d_v - 1\}.$$

Then the algorithms we propose for conversions involving the Newton or Lagrange bases require the precomputation and storage of $\varphi_v(u, \sigma_{v,0}), \ldots, \varphi_v(u, \sigma_{v,n_v-d_v-1})$ for all internal $v \in V$ and $u \in L_{v_\alpha}$. The identity (3.1) implies that $n(n-1)/2$ field elements are stored as result of this requirement. Moreover, it is straightforward to show that the precomputation can be performed with $\mathcal{O}(n^3)$ field operations.

Remark 3.1. If β is a Cantor basis, then Proposition 2.8 and property (3) of Lemma 2.9 imply that $\beta_v = (\beta_0, \ldots, \beta_{n_v-1})$ for $v \in V$. It follows that $\beta_{v_\delta,0} = \beta_0 = 1$ for all internal $v \in V$. Thus, no precomputations are required for converting between the LCH and monomial bases. For $v \in V$ and $u \in L_v$, the map $\varphi_v(u, \cdot) : \mathbb{F} \to \mathbb{F}$ is $\mathbb{F}_2$-linear, while Proposition 2.8 and properties (2) and (3) of Lemma 2.9 imply that

$$\varphi_v(u, \beta_i) = \begin{cases} \beta_i & \text{if } u = v, \\ \varphi_{v_\alpha}(u, \beta_i) & \text{if } u \neq v \text{ and } u \in L_{v_\alpha}, \\ \varphi_{v_\delta}(u, \beta_{i-d_v}) & \text{if } u \neq v,\ u \in L_{v_\delta} \text{ and } i \geq d_v, \\ 0 & \text{if } u \neq v,\ u \in L_{v_\delta} \text{ and } i < d_v, \end{cases}$$

for $i \in \{0, \ldots, n-1\}$. Consequently, the precomputations for conversions involving the Newton or Lagrange bases require fewer operations.

3.2. **Conversion between the Newton and Lin–Chung–Han bases.** The factorisations of the Newton basis polynomials provided by Lemma 2.1 include a shift of variables for one factor. Consequently, as we did for Lagrange basis to LCH basis conversion, we consider the more general problem of converting between a basis of shifted Newton polynomials and the LCH basis. Our conversions algorithms are then based on the following analogue of Lemma 2.2.

Lemma 3.2. *Let $v \in V$ be an internal vertex and $\ell \in \{1, \dots, 2^{n_v}\}$. Suppose that $f_0, \dots, f_{\ell-1}, \lambda, g_0, \dots, g_{\ell-1}, h_0, \dots, h_{\ell-1} \in \mathbb{F}$ satisfy*

(3.2)
$$\sum_{j=0}^{\min\left(\ell-2^{d_v}i,2^{d_v}\right)-1} g_{2^{d_v}i+j} X_{\beta_{v_\alpha},j} = \sum_{j=0}^{\min\left(\ell-2^{d_v}i,2^{d_v}\right)-1} f_{2^{d_v}i+j} N_{\beta_{v_\alpha},j}(x-\lambda-\omega_{\gamma_v,i})$$

for $i \in \{0, \dots, \lceil \ell/2^{d_v} \rceil - 1\}$, and

$$\sum_{i=0}^{\lceil (\ell-j)/2^{d_v} \rceil-1} h_{2^{d_v}i+j} X_{\beta_{v_\delta},i} = \sum_{i=0}^{\lceil (\ell-j)/2^{d_v} \rceil-1} g_{2^{d_v}i+j} N_{\beta_{v_\delta},i}(x-\eta) \tag{3.3}$$

for $j \in \{0, \dots, \min(2^{d_v}, \ell) - 1\}$, where $\eta = (\lambda/\beta_{v,0})^{2^{d_v}} - \lambda/\beta_{v,0}$. Then

$$\sum_{i=0}^{\ell-1} h_i X_{\beta_v,i} = \sum_{i=0}^{\ell-1} f_i N_{\beta_v,i}(x-\lambda). \tag{3.4}$$

The proof of Lemma 3.2 is omitted since it follows along similar lines to that of Lemma 2.2. Using the lemma, we obtain Algorithms 1 and 2 for conversion between the Newton and LCH bases. The algorithms each operate on a vector $(a_0, \dots, a_{\ell-1})$ of field elements that initially contains the coefficients of a polynomial on the input basis, and overwrite its entries with the coefficients of the polynomial on the output basis. The subvectors of this vector that appear in the algorithms would be represented in practice by auxiliary variables, e.g., by a pointer to their first element and a stride parameter. The map $\Delta : \mathbb{N} \to \mathbb{N}$ that appears in the algorithms is defined by $i \mapsto \min\{k \in \mathbb{N} \mid [i]_k = 0\}$. By noting that $\Delta(0), \Delta(1), \dots, \Delta(2^k - 2)$ is the transition sequence of the k-bit binary reflected Gray code, it is possible to successively compute the terms of the sequence at the cost of a small constant number of operations per element by the algorithm of Bitner, Ehrlich and Reingold [5] (see also [16]).

Theorem 3.3. *Algorithms 1 and 2 are correct.*

Proof. We only prove correctness for Algorithm 1, since the proof for Algorithm 2 is almost identical. Suppose that the input vertex v is a leaf. Then $\ell \in \{1, 2\}$ since $n_v = 1$. Moreover, $L_v = \{v\}$, $\varphi_v(v, \lambda) = \lambda/\beta_{v,0}$, $X_{\beta_v,0} = N_{\beta_v,0} = 1$ and $X_{\beta_v,1} = N_{\beta_v,1} = x/\beta_{v,0}$. It follows that Algorithm 1 produces the correct output whenever the input vertex is a leaf. Therefore, as (V, E) is a full binary tree, it is sufficient to show that for internal $v \in V$, if the algorithm produces the correct output whenever v_α or v_δ is given as an input, then the algorithm produces the correct output whenever v is given as an input.

Let $v \in V$ be an internal vertex and suppose that the algorithm produces the correct output whenever v_α or v_δ is given as an input. Let $\lambda \in \mathbb{F}$, $\ell \in \{1, \dots, 2^{n_v}\}$ and $f_0, \dots, f_{\ell-1} \in \mathbb{F}$. Suppose that Algorithm 1 is called on v, $(\varphi_v(u, \lambda))_{u \in L_v}$ and ℓ,

Algorithm 1 $\mathsf{N2X}(v, (\varphi_v(u,\lambda))_{u\in L_v}, \ell, (a_0,\dots,a_{\ell-1}))$

Input: a vertex $v \in V$, the vector $(\varphi_v(u,\lambda))_{u\in L_v} \in \mathbb{F}^{n_v}$ for some $\lambda \in \mathbb{F}$, $\ell \in \{1,\dots,2^{n_v}\}$, and $a_i = f_i \in \mathbb{F}$ for $i \in \{0,\dots,\ell-1\}$.
Output: $a_i = h_i \in \mathbb{F}$ for $i \in \{0,\dots,\ell-1\}$ such that (3.4) holds.

1: **if** v is a leaf **then**
2: **if** $\ell = 2$ **then** $a_0 \leftarrow a_0 + \varphi_v(v,\lambda)a_1$
3: **return**
4: $\ell_1 \leftarrow \lceil \ell/2^{d_v} \rceil - 1$, $\ell_2 \leftarrow \ell - 2^{d_v}\ell_1$, $\ell_2' \leftarrow \min(2^{d_v}, \ell)$
5: $\mu \leftarrow (\varphi_v(u,\lambda))_{u\in L_{v_\alpha}}$, $\nu \leftarrow (\varphi_v(u,\lambda))_{u\in L_{v_\delta}}$
6: **for** $i = 0,\dots,\ell_1 - 1$ **do**
7: $\mathsf{N2X}(v_\alpha, \mu, 2^{d_v}, (a_{2^{d_v}i}, a_{2^{d_v}i+1}, \dots, a_{2^{d_v}(i+1)-1}))$
8: $\mu \leftarrow \mu + (\varphi_v(u, \sigma_{v,\Delta(i)}))_{u\in L_{v_\alpha}}$
9: $\mathsf{N2X}(v_\alpha, \mu, \ell_2, (a_{2^{d_v}\ell_1}, a_{2^{d_v}\ell_1+1}, \dots, a_{\ell-1}))$
10: **for** $j = 0,\dots,\ell_2 - 1$ **do**
11: $\mathsf{N2X}(v_\delta, \nu, \ell_1 + 1, (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}\ell_1+j}))$
12: **for** $j = \ell_2,\dots,\ell_2' - 1$ **do**
13: $\mathsf{N2X}(v_\delta, \nu, \ell_1, (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}(\ell_1-1)+j}))$

Algorithm 2 $\mathsf{X2N}(v, (\varphi_v(u,\lambda))_{u\in L_v}, \ell, (a_0,\dots,a_{\ell-1}))$

Input: a vertex $v \in V$, the vector $(\varphi_v(u,\lambda))_{u\in L_v} \in \mathbb{F}^{n_v}$ for some $\lambda \in \mathbb{F}$, $\ell \in \{1,\dots,2^{n_v}\}$, and $a_i = h_i \in \mathbb{F}$ for $i \in \{0,\dots,\ell-1\}$.
Output: $a_i = f_i \in \mathbb{F}$ for $i \in \{0,\dots,\ell-1\}$ such that (3.4) holds.

1: **if** v is a leaf **then**
2: **if** $\ell = 2$ **then** $a_0 \leftarrow a_0 + \varphi_v(v,\lambda)a_1$
3: **return**
4: $\ell_1 \leftarrow \lceil \ell/2^{d_v} \rceil - 1$, $\ell_2 \leftarrow \ell - 2^{d_v}\ell_1$, $\ell_2' \leftarrow \min(2^{d_v}, \ell)$
5: $\mu \leftarrow (\varphi_v(u,\lambda))_{u\in L_{v_\alpha}}$, $\nu \leftarrow (\varphi_v(u,\lambda))_{u\in L_{v_\delta}}$
6: **for** $j = 0,\dots,\ell_2 - 1$ **do**
7: $\mathsf{X2N}(v_\delta, \nu, \ell_1 + 1, (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}\ell_1+j}))$
8: **for** $j = \ell_2,\dots,\ell_2' - 1$ **do**
9: $\mathsf{X2N}(v_\delta, \nu, \ell_1, (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}(\ell_1-1)+j}))$
10: **for** $i = 0,\dots,\ell_1 - 1$ **do**
11: $\mathsf{X2N}(v_\alpha, \mu, 2^{d_v}, (a_{2^{d_v}i}, a_{2^{d_v}i+1}, \dots, a_{2^{d_v}(i+1)-1}))$
12: $\mu \leftarrow \mu + (\varphi_v(u, \sigma_{v,\Delta(i)}))_{u\in L_{v_\alpha}}$
13: $\mathsf{X2N}(v_\alpha, \mu, \ell_2, (a_{2^{d_v}\ell_1}, a_{2^{d_v}\ell_1+1}, \dots, a_{\ell-1}))$

with $a_i = f_i$ for $i \in \{0,\dots,\ell-1\}$. Then Lines 5 and 8 of the algorithm and the $\mathbb{F}_2$-linearity of the maps $\varphi_v(u,\,\cdot\,) : \mathbb{F} \to \mathbb{F}$, for $u \in L_v$, ensure that

$$\mu = \left(\varphi_v(u,\lambda) + \sum_{j=0}^{i-1} \varphi_v\big(u, \sigma_{v,\Delta(j)}\big)\right)_{u\in L_{v_\alpha}} = \left(\varphi_v\left(u, \lambda + \sum_{j=0}^{i-1} \sigma_{v,\Delta(j)}\right)\right)_{u\in L_{v_\alpha}}.$$

each time the recursive call of Line 7 is performed. As $\gamma_v = (\beta_{v,d_v}, \ldots, \beta_{v,n_v-d_v-1})$, we have $\omega_{\gamma_v,0} = 0$ and

$$\omega_{\gamma_v,i} = \omega_{\gamma_v,i-1} + \sum_{j=0}^{\Delta(i-1)} \beta_{v,d_v+j} = \omega_{\gamma_v,i-1} + \sigma_{v,\Delta(i-1)} = \sum_{j=0}^{i-1} \sigma_{v,\Delta(j)}$$

for $i \in \{1, \ldots, 2^{n_v-d_v} - 1\}$. Thus,

$$\mu = (\varphi_v(u, \lambda + \omega_{\gamma_v,i}))_{u \in L_{v_\alpha}} = (\varphi_{v_\alpha}(u, \lambda + \omega_{\gamma_v,i}))_{u \in L_{v_\alpha}}$$

each time the recursive call of Line 7 is performed. Similarly,

$$\mu = \left(\varphi_{v_\alpha}\left(u, \lambda + \omega_{\gamma_v,\lceil \ell/2^{d_v} \rceil - 1}\right)\right)_{u \in L_{v_\alpha}}$$

when the recursive call of Line 9 is performed. Therefore, the assumption that Algorithm 1 produces the correct output whenever v_α is given as an input implies that Lines 4–9 set $a_i = g_i$ for $i \in \{0, \ldots, \ell - 1\}$, where $g_0, \ldots, g_{\ell-1}$ are the unique elements in $\mathbb{F}$ such that (3.2) holds.

The recursive calls of Lines 11 and 13 are made with

$$\nu = (\varphi_v(u, \lambda))_{u \in L_{v_\delta}} = (\varphi_{v_\delta}(u, \eta))_{u \in L_{v_\delta}},$$

where $\eta = (\lambda/\beta_{v,0})^{2^{d_v}} - \lambda/\beta_{v,0}$. Therefore, the assumption that Algorithm 1 produces the correct output whenever v_δ is given as an input implies that Lines 10–13 set $a_i = h_i$ for $i \in \{0, \ldots, \ell - 1\}$, where $h_0, \ldots, h_{\ell-1}$ are the unique elements in $\mathbb{F}$ such that (3.3) holds. Hence, Lemma 3.2 implies that (3.4) holds at the end of the algorithm. □

For conversions between the Newton and LCH bases the shift parameter λ is equal to zero for the initial calls to Algorithms 1 and 2. In this case, the vector $(\varphi_v(u, \lambda))_{u \in L_v}$ contains all zeros. If an application arises where it is necessary for the initial call to be made with $\lambda \neq 0$, then the input vector can be computed with $\mathcal{O}(n_v^2)$ field operations. Storing the input vector and the vectors μ and ν that appear in the algorithms requires storing $\mathcal{O}(n_v^2)$ field elements. Including the computation and storage of the precomputed elements $\varphi_v(u, \sigma_{v,i})$, it follows that Algorithms 1 and 2 require auxiliary storage for $\mathcal{O}(n^2)$ field elements, while all precomputations can be performed with $\mathcal{O}(n^3)$ field operations.

The number of multiplications performed by Algorithms 1 and 2 is independent of the choice of reduction tree. This is not true of the number of additions performed by the algorithms due to the updates made to the vector μ in the recursive case. Line 8 of Algorithm 1 and Line 12 of Algorithm 2 each perform $(\lceil \ell/2^{d_v} \rceil - 1)d_v$ additions over all iterations of their containing loops. Consequently, we should aim to avoid small values of d_v when choosing a reduction tree. Moreover, we expect the number of additions performed by the algorithms to be maximised when the subtree rooted on the initial input vertex satisfies $\mathrm{Im}(d) \subseteq \{0, 1\}$.

Example 3.4. If β is a Cantor basis, then Proposition 2.8 implies that $d_v \leq 2^{\lceil \log_2 n_v \rceil - 1}$ for all internal $v \in V$, and guarantees the existence of a reduction tree for β such that equality always holds. For n up to fifteen, we confirmed experimentally that this choice of reduction tree always minimises the number of additions performed by Algorithms 1 and 2 over all possible reduction trees, while those with $\mathrm{Im}(d) \subseteq \{0, 1\}$ were found to always maximise the number of additions performed by the algorithms. Figure 2 shows the maximum and minimum number

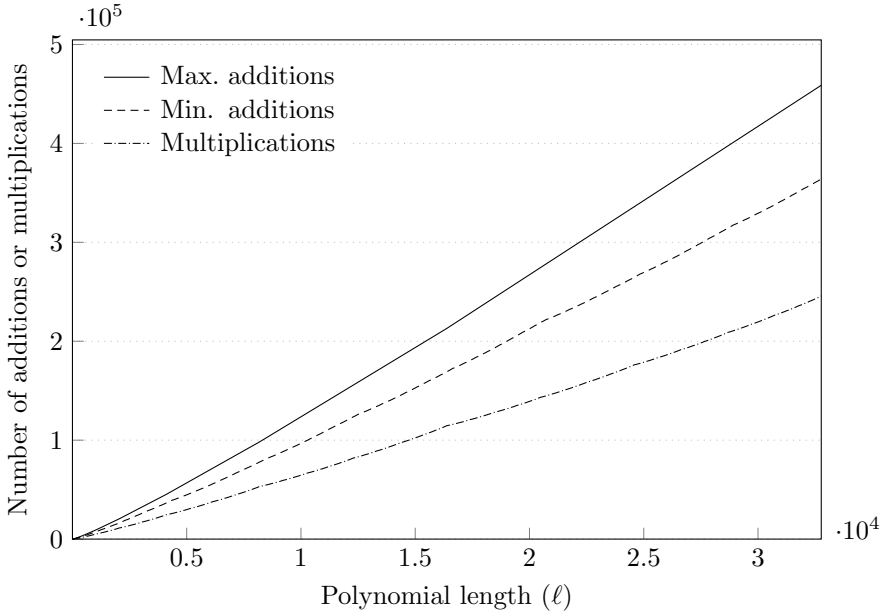

FIGURE 2. Maximum and minimum number of operations performed by Algorithms 1 and 2 for Cantor bases of dimension 15.

of additions performed over all reduction trees for $n = 15$, as well as the number of multiplications performed for all trees.

The difference between the maximum and minimum number of additions shown in Figure 2 is reflected in the bounds that we obtain on the complexities of the algorithms.

Theorem 3.5. *Algorithms 1 and 2 perform at most* $\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil$ *multiplications and* $\ell(\lceil \log_2 \ell \rceil - 1) + 1$ *additions in* $\mathbb{F}$. *If* $d_v = 2^{\lceil \log_2 n_v \rceil - 1}$ *for all internal* $v \in V$, *then the algorithms perform at most* $(3\ell - 2)\lceil \log_2 \ell \rceil / 4$ *additions in* $\mathbb{F}$.

We split the proof of Theorem 3.5 into three lemmas, one for each of the three bounds. It is clear that Algorithms 1 and 2 perform the same number of multiplications when given identical inputs. Consequently, we only prove the bounds for Algorithm 1. The three bounds are equal to zero if $\ell = 1$, and one if $\ell = 2$. Thus, the bounds hold if the input vertex is a leaf. Therefore, for each of the three bounds it is sufficient to show that if $v \in V$ is an internal vertex such that the bound holds whenever the input vertex is v_α or v_δ, then the bound holds whenever v is the input vertex.

Lemma 3.6. *Algorithms 1 and 2 perform at most* $\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil$ *multiplications in* $\mathbb{F}$.

Proof. Suppose that the input vertex $v \in V$ to Algorithm 1 is an internal vertex such that the algorithm performs at most $\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil$ multiplications in $\mathbb{F}$ whenever v_α or v_δ is given as the input vertex. If $\ell_1 = 0$, then $\ell_2 = \ell_2' = \ell$ and the algorithm performs at most

$$\left\lfloor \frac{\ell_2}{2} \right\rfloor \lceil \log_2 \ell_2 \rceil + \ell_2 \left\lfloor \frac{1}{2} \right\rfloor \lceil \log_2 1 \rceil = \left\lfloor \frac{\ell}{2} \right\rfloor \lceil \log_2 \ell \rceil$$

multiplications. Therefore, suppose that $\ell_1 \neq 0$. Then, as $\ell_2 \leq 2^{d_v}$, Lines 6–9 of the algorithm perform at most

$$2^{d_v-1}\ell_1 d_v + \left\lfloor \frac{\ell_2}{2} \right\rfloor \lceil \log_2 \ell_2 \rceil \leq \left(2^{d_v-1}\ell_1 + \left\lfloor \frac{\ell_2}{2} \right\rfloor \right) d_v = \left\lfloor \frac{\ell}{2} \right\rfloor d_v \tag{3.5}$$

multiplications. Let $k \in \mathbb{N}$ such that $2^{k-1} < \ell_1 + 1 \leq 2^k$. Then $\lceil \log_2 \ell_1 + 1 \rceil = k$ and $2^{d_v+k-1} < 2^{d_v}(\ell_1 + \ell_2/2^{d_v}) = \ell \leq 2^{d_v+k}$, since $k \geq 1$ and $0 < \ell_2/2^{d_v} \leq 1$. Thus, $\lceil \log_2 \ell_1 + 1 \rceil = \lceil \log_2 \ell \rceil - d_v$. It follows that Lines 10–13 of the algorithm perform at most

$$\begin{aligned} &\ell_2 \left\lfloor \frac{\ell_1 + 1}{2} \right\rfloor \lceil \log_2 \ell_1 + 1 \rceil + \left(2^{d_v} - \ell_2\right) \left\lfloor \frac{\ell_1}{2} \right\rfloor \lceil \log_2 \ell_1 \rceil \\ &\qquad \leq \left\lfloor \frac{\ell_2(\ell_1 + 1) + \left(2^{d_v} - \ell_2\right)\ell_1}{2} \right\rfloor \lceil \log_2 \ell_1 + 1 \rceil = \left\lfloor \frac{\ell}{2} \right\rfloor (\lceil \log_2 \ell \rceil - d_v) \end{aligned}$$

multiplications. By combining this bound with (3.5), it follows that Algorithm 1 performs at most $\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil$ multiplications if $\ell_1 \neq 0$. □

Lemma 3.7. *Algorithms 1 and 2 perform at most* $\ell(\lceil \log_2 \ell \rceil - 1) + 1$ *additions in* $\mathbb{F}$.

Proof. Suppose that the input vertex $v \in V$ to Algorithm 1 is an internal vertex such that the algorithm performs at most $\ell(\lceil \log_2 \ell \rceil - 1) + 1$ additions in $\mathbb{F}$ whenever v_α or v_δ is given as the input vertex. If $\ell_1 = 0$, then $\ell_2 = \ell_2' = \ell$ and the algorithm performs at most

$$\ell_2(\lceil \log_2 \ell_2 \rceil - 1) + 1 + \ell_2(1(\lceil \log_2 1 \rceil - 1) + 1) = \ell(\lceil \log_2 \ell \rceil - 1) + 1$$

additions. Therefore, suppose that $\ell_1 \neq 0$. Then Lines 6–9 of the algorithm perform at most

$$\begin{aligned} &\ell_1\left(2^{d_v}(d_v - 1) + 1 + d_v\right) + \ell_2(\lceil \log_2 \ell_2 \rceil - 1) + 1 \\ &\qquad = \ell(d_v - 1) + 1 + \ell_1(d_v + 1) + \ell_2(\lceil \log_2 \ell_2 \rceil - d_v) \end{aligned}$$

additions, while Lines 10–13 perform at most

$$\begin{aligned} &\ell_2(\ell_1 + 1)(\lceil \log_2 \ell_1 + 1 \rceil - 1) + \left(2^{d_v} - \ell_2\right)\ell_1(\lceil \log_2 \ell_1 \rceil - 1) + 2^{d_v} \\ &\qquad \leq \ell(\lceil \log_2 \ell_1 + 1 \rceil - 1) + 2^{d_v} = \ell(\lceil \log_2 \ell \rceil - d_v) - \ell + 2^{d_v} \end{aligned}$$

additions. As $1 \leq \ell_2 \leq 2^{d_v}$, it follows that Algorithm 1 performs at most

$$\begin{aligned} &\ell(\lceil \log_2 \ell \rceil - 1) + 1 + \ell_1(d_v + 1) + \ell_2(\lceil \log_2 \ell_2 \rceil - d_v) - \ell + 2^{d_v} \\ &\quad = \ell(\lceil \log_2 \ell \rceil - 1) + 1 - \left(2^{d_v} - d_v - 1\right)(\ell_1 - 1) + d_v + 1 + \ell_2 \left\lceil \log_2 \frac{\ell_2}{2^{d_v+1}} \right\rceil \\ &\quad \leq \ell(\lceil \log_2 \ell \rceil - 1) + 1 + d_v + 1 - \frac{\left\lfloor \log_2 2^{d_v+1}/\ell_2 \right\rfloor}{2^{d_v+1}/\ell_2} 2^{d_v+1} \\ &\quad \leq \ell(\lceil \log_2 \ell \rceil - 1) + 1 \end{aligned}$$

additions if $\ell_1 \neq 0$. □

Lemma 3.8. *Suppose that* $d_v = 2^{\lceil \log_2 n_v \rceil - 1}$ *for all internal* $v \in V$. *Then Algorithms 1 and 2 perform at most* $(3\ell - 2)\lceil \log_2 \ell \rceil / 4$ *additions in* $\mathbb{F}$.

Proof. Suppose that $d_v = 2^{\lceil \log_2 n_v \rceil - 1}$ for all internal $v \in V$. Furthermore, suppose that the input vertex $v \in V$ to Algorithm 1 is an internal vertex such that the algorithm performs at most $(3\ell - 2)\lceil \log_2 \ell \rceil / 4$ additions in $\mathbb{F}$ whenever v_α or v_δ is given as the input vertex. If $\ell_1 = 0$, then $\ell_2 = \ell_2' = \ell$ and the algorithm performs at most

$$\frac{3\ell_2 - 2}{4}\lceil \log_2 \ell_2 \rceil + \ell_2\left(\frac{3-2}{4}\lceil \log_2 1 \rceil\right) = \frac{3\ell - 2}{4}\lceil \log_2 \ell \rceil.$$

additions. Therefore, suppose that $\ell_1 \neq 0$. Then, as $\ell_2 \leq 2^{d_v}$, Lines 6–9 of the algorithm perform at most

$$\ell_1\left(\frac{3 \cdot 2^{d_v} - 2}{4} d_v + d_v\right) + \frac{3\ell_2 - 2}{4}\lceil \log_2 \ell_2 \rceil \leq \frac{3\ell - 2}{4} d_v + \frac{\ell_1}{2} d_v$$

additions, while Lines 10–13 perform at most

$$\begin{aligned} \ell_2 \frac{3(\ell_1 + 1) - 2}{4}\lceil \log_2 \ell_1 + 1 \rceil + \left(2^{d_v} - \ell_2\right)\frac{3\ell_1 - 2}{4}\lceil \log_2 \ell_1 \rceil & \\ \leq \frac{3\ell - 2^{d_v+1}}{4}\lceil \log_2 \ell_1 + 1 \rceil & \\ \leq \frac{3\ell - 2}{4}(\lceil \log_2 \ell \rceil - d_v) - \frac{2^{d_v} - 1}{2}\log_2(\ell_1 + 1) & \end{aligned}$$

additions. It follows that Algorithm 1 performs at most

$$\frac{3\ell - 2}{4}\lceil \log_2 \ell \rceil + \frac{d_v}{2}\log_2(\ell_1 + 1)\left(\frac{\ell_1}{\log_2(\ell_1 + 1)} - \frac{2^{d_v} - 1}{d_v}\right) \leq \frac{3\ell - 2}{4}\lceil \log_2 \ell \rceil$$

additions if $\ell_1 \neq 0$, since $\ell_1 + 1 \leq 2^{n_v - d_v} \leq 2^{d_v}$ and the function $x / \log_2(x + 1)$ is increasing for $x \geq 1$. □

3.3. **Conversion from the Lagrange basis to the Lin–Chung–Han basis.** Our algorithms for converting between the Lagrange and LCH bases are direct analogues of Harvey's cache-friendly truncated FFT and inverse truncated FFT algorithms [15]. We align our presentation of the algorithms and their proofs of correctness with their counterparts given by Harvey, and direct the reader to Harvey's paper for further motivation behind the algorithms. In this section, we focus on the problem of converting from the Lagrange basis to the LCH basis. The inverse problem of converting from the LCH basis to the Lagrange basis is considered separately in the next section.

We propose Algorithm 3 for converting from the Lagrange basis to the LCH basis. Like the algorithms of Section 3.2, Algorithm 3 operates on a vector of field elements whose initial entries are overwritten with the output. However, the length of the vector is determined by the input vertex rather than the parameter ℓ which bounds the polynomial length. Consequently, the vector may have coordinates that are never used to store input or output values, but which are still used for intermediate computations. If the parameter c is less than ℓ, then the algorithm differs substantially from the other basis conversion algorithms in this paper by requiring that the vector initially contain a combination of coefficients from a polynomial's representations on the input and output bases. If the parameter b is set equal to one, then the algorithm has the additional unique feature of being required to compute a coefficient from the input basis representation. These two parameters are part of the internal mechanics of the recursive approach used by the algorithm, and

in most practical applications, such as the multiplication of binary polynomials, the initial call to the algorithm is made with $c = \ell$ and $b = 0$. However, one may be required to initially call the algorithm with $c < \ell$ if it used within the Hermite interpolation algorithm of Coxon [12].

The reduction used by Algorithm 3 is provided by the following generalisation of Lemma 2.2, which is rephrased to reflect the fact that the algorithm takes in a mixture of coefficients from representations on the Lagrange and LCH bases.

Lemma 3.9. *Let $v \in V$ be an internal vertex and $\ell \in \{1, \dots, 2^{n_v}\}$. Suppose that $f_0, \dots, f_{2^{n_v}-1}, \lambda, h_0, \dots, h_{\ell-1} \in \mathbb{F}$ satisfy*

$$\sum_{i=0}^{\ell-1} h_i X_{\beta_v,i} = \sum_{i=0}^{2^{n_v}-1} f_i L_{\beta_v,i}(x - \lambda). \tag{3.6}$$

Then there exist unique elements $g_0, \dots, g_{2^{n_v}-1} \in \mathbb{F}$ such that

$$\sum_{j=0}^{\min\left(2^{d_v},\ell\right)-1} g_{2^{d_v}i+j} X_{\beta_{v_\alpha},j} = \sum_{j=0}^{2^{d_v}-1} f_{2^{d_v}i+j} L_{\beta_{v_\alpha},j}(x - \lambda - \omega_{\gamma_v,i}) \tag{3.7}$$

for $i \in \{0, \dots, 2^{n_v-d_v} - 1\}$, and

$$\sum_{i=0}^{\lfloor(\ell-1-j)/2^{d_v}\rfloor} h_{2^{d_v}i+j} X_{\beta_{v_\delta},i} = \sum_{i=0}^{2^{n_v-d_v}-1} g_{2^{d_v}i+j} L_{\beta_{v_\delta},i}(x - \eta) \tag{3.8}$$

for $j \in \{0, \dots, 2^{d_v} - 1\}$, where $\eta = (\lambda/\beta_{v,0})^{2^{d_v}} - \lambda/\beta_{v,0}$.

Proof. Let $v \in V$, $\ell \in \{1, \dots, 2^{n_v}\}$ and $f_0, \dots, f_{2^{n_v}-1}, \lambda, h_0, \dots, h_{\ell-1} \in \mathbb{F}$ satisfy the conditions of the lemma. Then there exist unique elements $g_0, \dots, g_{2^{n_v}-1} \in \mathbb{F}$ such that (3.8) holds for $j \in \{0, \dots, 2^{d_v} - 1\}$. We show that they also satisfy (3.7) for $i \in \{0, \dots, 2^{n_v-d_v} - 1\}$. If $j \in \{0, \dots, 2^{d_v} - 1\}$ satisfies $j \geq \ell$, then (3.8) implies that $g_{2^{d_v}i+j} = 0$ for $i \in \{0, \dots, 2^{n_v-d_v} - 1\}$. Therefore, if we let $h_\ell = \cdots = h_{2^{n_v}-1} = 0$, then the left-hand sides of (3.6), (3.7) and (3.8) remain unchanged if we replace ℓ by 2^{n_v} in the summation bounds. Consequently, (3.7) must hold for $i \in \{0, \dots, 2^{n_v-d_v} - 1\}$, since otherwise Lemma 2.2 allows us to contradict the uniqueness of the coefficients $f_0, \dots, f_{2^{n_v}-1}$ in (3.6) by writing the polynomial on the left-hand side of (3.7) on the basis

$$\left\{L_{\beta_{v_\alpha},0}(x - \lambda - \omega_{\gamma_v,i}), \dots, L_{\beta_{v_\alpha},2^{d_v}-1}(x - \lambda - \omega_{\gamma_v,i})\right\}$$

for $i \in \{0, \dots, 2^{n_v-d_v} - 1\}$. □

Theorem 3.10. *Algorithm 3 is correct.*

Proof. Suppose that $v \in V$ is a leaf. Then Table 2 displays the input and output requirements of Algorithm 3 on the vector $(a_0, \dots, a_{2^{n_v}-1}) = (a_0, a_1)$ for each possible input that includes v. The table also shows the output of the algorithm as computed by Lines 1–7. The elements f_i and h_i that appear in a row of the table are the coefficients of (3.6) for the specified value of ℓ. Elements denoted by asterisks are unspecified by the algorithm. As v is a leaf, we have

$$X_{\beta_v,0} = 1, \quad X_{\beta_v,1} = \frac{x}{\beta_{v,0}}, \quad L_{\beta_v,0} = \frac{x}{\beta_{v,0}} + 1 \quad \text{and} \quad L_{\beta_v,1} = \frac{x}{\beta_{v,0}}.$$

Algorithm 3 $\mathsf{L2X}(v, (\varphi_v(u,\lambda))_{u\in L_v}, c, \ell, b, (a_0,\dots,a_{2^{n_v}-1}))$

Input: a vertex $v \in V$, the vector $(\varphi_v(u,\lambda))_{u\in L_v} \in \mathbb{F}^{n_v}$ for some $\lambda \in \mathbb{F}$, $c, \ell \in \mathbb{N}$ such that $c \leq \ell$ and $1 \leq \ell \leq 2^{n_v}$, $b \in \{0,1\}$ such that $1 \leq b + c \leq 2^{n_v}$, $a_i = f_i \in \mathbb{F}$ for $i \in \{0,\dots,c-1\}$, and $a_i = h_i \in \mathbb{F}$ for $i \in \{c,\dots,\ell-1\}$.

Output: $a_i = h_i \in \mathbb{F}$ for $i \in \{0,\dots,c-1\}$ such that (3.6) holds for some $f_c,\dots,f_{2^{n_v}-1} \in \mathbb{F}$, and $a_c = f_c$ if $b = 1$.

if v is a leaf **then**
  if $c = 2$ **then** $a_1 \leftarrow a_0 + a_1$, $a_0 \leftarrow a_0 + \varphi_v(v,\lambda)a_1$
  if $c = 1$, $\ell = 2$ and $b = 1$ **then** $w \leftarrow \varphi_v(v,\lambda)a_1$, $a_1 \leftarrow a_0 + a_1$, $a_0 \leftarrow a_0 + w$
  if $c = 1$, $\ell = 2$ and $b = 0$ **then** $a_0 \leftarrow a_0 + \varphi_v(v,\lambda)a_1$
  if $c = 0$ and $\ell = 2$ **then** $a_0 \leftarrow a_0 + \varphi_v(v,\lambda)a_1$
  if $c = 1$, $\ell = 1$ and $b = 1$ **then** $a_1 \leftarrow a_0$
  return
$c_1 \leftarrow \lfloor c/2^{d_v} \rfloor$, $c_2 \leftarrow c - 2^{d_v}c_1$
$\ell_1 \leftarrow \lfloor \ell/2^{d_v} \rfloor$, $\ell_2 \leftarrow \ell - 2^{d_v}\ell_1$
$\ell_2' \leftarrow \min(2^{d_v}, \ell)$, $b' \leftarrow \min(b + c_2, 1)$
$s \leftarrow \min(c_2, \ell_2)$, $t \leftarrow \max(c_2, \ell_2)$
$\mu \leftarrow (\varphi_v(u,\lambda))_{u\in L_{v_\alpha}}$, $\nu \leftarrow (\varphi_v(u,\lambda))_{u\in L_{v_\delta}}$
for $i = 0,\dots,c_1 + b' - 2$ **do**
  $\mathsf{L2X}(v_\alpha, \mu, 2^{d_v}, 2^{d_v}, 0, (a_{2^{d_v}i}, a_{2^{d_v}i+1}, \dots, a_{2^{d_v}(i+1)-1}))$
  $\mu \leftarrow \mu + (\varphi_v(u, \sigma_{v,\Delta(i)}))_{u\in L_{v_\alpha}}$
if $b' = 0$ **then**
  $\mathsf{L2X}(v_\alpha, \mu, 2^{d_v}, 2^{d_v}, 0, (a_{2^{d_v}(c_1-1)}, a_{2^{d_v}(c_1-1)+1}, \dots, a_{2^{d_v}c_1-1}))$
for $j = c_2,\dots,t-1$ **do**
  $\mathsf{L2X}(v_\delta, \nu, c_1, \ell_1 + 1, b', (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}(2^{n_v-d_v}-1)+j}))$
for $j = t,\dots,\ell_2' - 1$ **do**
  $\mathsf{L2X}(v_\delta, \nu, c_1, \ell_1, b', (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}(2^{n_v-d_v}-1)+j}))$
if $b' = 1$ **then**
  $\mathsf{L2X}(v_\alpha, \mu, c_2, \ell_2', b, (a_{2^{d_v}c_1}, a_{2^{d_v}c_1+1}, \dots, a_{2^{d_v}(c_1+1)-1}))$
for $j = 0,\dots,s-1$ **do**
  $\mathsf{L2X}(v_\delta, \nu, c_1 + 1, \ell_1 + 1, 0, (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}(2^{n_v-d_v}-1)+j}))$
for $j = s,\dots,c_2-1$ **do**
  $\mathsf{L2X}(v_\delta, \nu, c_1 + 1, \ell_1, 0, (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}(2^{n_v-d_v}-1)+j}))$

Moreover, $\varphi_v(v,\lambda) = \lambda/\beta_{v,0}$ for $\lambda \in \mathbb{F}$. Thus, the coefficients of (3.6) satisfy $h_0 = f_0 + \varphi_v(v,\lambda)(f_0 + f_1)$ and $h_1 = f_0 + f_1$ if $\ell = 2$, and $h_0 = f_0 = f_1$ if $\ell = 1$. Using these equation, one can readily verify that the computed output agrees with the required output for all inputs. Consequently, Algorithm 3 produces the correct output whenever the input vertex is a leaf. Therefore, as (V, E) is a full binary tree, it is sufficient to show that for all internal $v \in V$, if the algorithm produces the correct output whenever v_α or v_δ is given as an input, then it produces the correct output whenever v is given as an input.

Let $v \in V$ be an internal vertex and suppose that Algorithm 3 produces the correct output whenever v_α or v_δ is given as an input. Let $\lambda \in \mathbb{F}$, $c, \ell \in \mathbb{N}$ such that $c \leq \ell$ and $1 \leq \ell \leq 2^{n_v}$, and $b \in \{0,1\}$ such that $1 \leq b + c \leq 2^{n_v}$. Suppose that Algorithm 3 is called on v, $(\varphi_v(u,\lambda))_{u\in L_v}$, c, ℓ, b and $(a_0,\dots,a_{2^{n_v}-1})$, with $a_i = f_i \in \mathbb{F}$ for $i \in \{0,\dots,c-1\}$, and $a_i = h_i \in \mathbb{F}$ for $i \in \{c,\dots,\ell-1\}$. Then

Input					Required output		Computed output	
c	ℓ	b	a_0	a_1	a_0	a_1	a_0	a_1
2	2	0	f_0	f_1	h_0	h_1	$f_0 + \varphi_v(v,\lambda)(f_0 + f_1)$	$f_0 + f_1$
1	2	1	f_0	h_1	h_0	f_1	$f_0 + \varphi_v(v,\lambda)h_1$	$f_0 + h_1$
1	2	0	f_0	h_1	h_0	$*$	$f_0 + \varphi_v(v,\lambda)h_1$	$*$
0	2	1	h_0	h_1	f_0	$*$	$h_0 + \varphi_v(v,\lambda)h_1$	$*$
1	1	1	f_0	$*$	h_0	f_1	f_0	f_0
1	1	0	f_0	$*$	h_0	$*$	f_0	$*$
0	1	1	h_0	$*$	f_0	$*$	h_0	$*$

TABLE 2. Required and computed outputs of Algorithm 3 when v is a leaf.

there exist unique elements $h_0, \ldots, h_{c-1}, f_c, \ldots, f_{2^{n_v}-1} \in \mathbb{F}$ such that (3.6) holds. In-turn, Lemma 3.9 implies that there exist unique elements $g_0, \ldots, g_{2^{n_v}-1} \in \mathbb{F}$ such that (3.7) and (3.8) hold.

Repeating arguments from the proof of Theorem 3.3 shows that vector μ is equal to $(\varphi_{v_\alpha}(u, \lambda + \omega_{\gamma_v,i}))_{u \in L_{v_\alpha}}$ each time the recursive call of Line 14 is made, equal to $(\varphi_{v_\alpha}(u, \lambda + \omega_{\gamma_v,c_1-1}))_{u \in L_{v_\alpha}}$ whenever the recursive call of Line 17 is performed, and equal to $(\varphi_{v_\alpha}(u, \lambda + \omega_{\gamma_v,c_1}))_{u \in L_{v_\alpha}}$ whenever the recursive call of Line 23 is performed. Similarly, the vector ν is equal to $(\varphi_{v_\delta}(u, \eta))_{u \in L_{v_\delta}}$ for the recursive calls of Lines 19, 21, 25 and 27. It follows that if the recursive call of Line 14 is made with $a_{2^{d_v}i+j} = f_{2^{d_v}i+j}$ for $j \in \{0, \ldots, 2^{d_v} - 1\}$, then (3.7) and the assumption that the algorithm produces the correct output whenever v_α is given as an input imply that $a_{2^{d_v}i+j} = g_{2^{d_v}i+j}$ for $j \in \{0, \ldots, 2^{d_v} - 1\}$ afterwards. Similarly, if the recursive call of Line 19 is made with $a_{2^{d_v}i+j} = g_{2^{d_v}i+j}$ for $i \in \{0, \ldots, c_1 - 1\}$ and $a_{2^{d_v}i+j} = h_{2^{d_v}i+j}$ for $i \in \{c_1, \ldots, \ell_1\}$, then (3.8) and the assumption that the algorithm produces the correct output whenever v_δ is given as an input imply that $a_{2^{d_v}i+j} = h_{2^{d_v}i+j}$ for $i \in \{0, \ldots, c_1 - 1\}$, and $a_{2^{d_v}c_1+j} = g_{2^{d_v}c_1+j}$ if $b' = 1$, afterwards. Similar statements hold for the remaining recursive calls made by the algorithm.

The remainder of the proof is split into four cases. For each case, we provide an example in either Figure 3 or 4 of how the vector $(a_0, \ldots, a_{2^{n_v}-1})$ evolves during the algorithm. In the figures, the vector is represented by the $2^{n_v-d_v} \times 2^{d_v}$ matrix $(a_{2^{d_v}i+j})_{0 \le i < 2^{n_v-d_v}, 0 \le j < 2^{d_v}}$. Under this representation, the subvectors that are subjected to recursive calls by the algorithm correspond to row and column vectors of the matrix. Asterisks in the figures represent unspecified entries, while entries surrounded by parenthesis are computed only if $b = 1$, and are unspecified otherwise.

Case a: Suppose that $\ell_1 = 0$. Then Lines 8–11 of the algorithm set $c_1 = 0$, $c_2 = s = c$, $\ell_2 = \ell_2' = t = \ell$ and $b' = 1$. Thus, Lines 13–17 have no effect. Equation (3.8) implies that Lines 18–19 set $a_j = g_j$ for $j \in \{c, \ldots, \ell - 1\}$. Lines 20–21 have no effect since $\ell_2' = t$. Equation (3.7) implies that Lines 22–23 set $a_i = g_i$ for $i \in \{0, \ldots, c-1\}$, and $a_c = f_c$ if $b = 1$. Equation (3.8) implies that Lines 24–25 set $a_i = h_i$ for $i \in \{0, \ldots, c-1\}$. Finally, Lines 26–27 have no effect since $s = c_2$.

Case b: Suppose that $\ell_1 \neq 0$ and $c_2 = 0$. Then Lines 8–11 set $c_1 = c/2^{d_v}$, $\ell_2' = 2^{d_v}$, $b' = b$, $s = 0$ and $t = \ell_2$. Equation (3.7) implies that Lines 13–17 set

Case a **Case b**

$$\begin{pmatrix} f_0 & f_1 & f_2 & h_3 & h_4 & h_5 & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix} \quad \begin{pmatrix} f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ f_8 & f_9 & f_{10} & f_{11} & f_{12} & f_{13} & f_{14} & f_{15} \\ h_{16} & h_{17} & h_{18} & h_{19} & h_{20} & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix}$$

(3a) Initial contents.

$$\begin{pmatrix} f_0 & f_1 & f_2 & h_3 & h_4 & h_5 & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix} \quad \begin{pmatrix} g_0 & g_1 & g_2 & g_3 & g_4 & g_5 & g_6 & g_7 \\ g_8 & g_9 & g_{10} & g_{11} & g_{12} & g_{13} & g_{14} & g_{15} \\ h_{16} & h_{17} & h_{18} & h_{19} & h_{20} & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix}$$

(3b) After Lines 13–17 make recursive calls on the highlighted rows.

$$\begin{pmatrix} f_0 & f_1 & f_2 & g_3 & g_4 & g_5 & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix} \quad \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 \\ h_8 & h_9 & h_{10} & h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ (g_{16}) & (g_{17}) & (g_{18}) & (g_{19}) & (g_{20}) & (g_{21}) & (g_{22}) & (g_{23}) \\ * & * & * & * & * & * & * & * \end{pmatrix}$$

(3c) After Lines 18–21 make recursive calls on the highlighted columns.

$$\begin{pmatrix} g_0 & g_1 & g_2 & (f_3) & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix} \quad \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 \\ h_8 & h_9 & h_{10} & h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ (f_{16}) & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix}$$

(3d) After Lines 23 makes a recursive call on the highlighted row.

$$\begin{pmatrix} h_0 & h_1 & h_2 & (f_3) & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix} \quad \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 \\ h_8 & h_9 & h_{10} & h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ (f_{16}) & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix}$$

(3e) After Lines 24–27 make recursive calls on the highlighted columns.

FIGURE 3. Evolution of the vector $(a_0, \ldots, a_{2^{n_v}-1})$ during Algorithm 3 for $n_v = 5$, $d_v = 3$, $c = 3$, $\ell = 6$ (Case a), and $n_v = 5$, $d_v = 3$, $c = 16$, $\ell = 21$ (Case b).

$a_i = g_i$ for $i \in \{0, \ldots, c-1\}$. Thus, (3.8) implies that Lines 18–21 set $a_i = h_i$ for $i \in \{0, \ldots, c-1\}$, and $a_{c+j} = g_{c+j}$ for $j \in \{0, \ldots, 2^{d_v} - 1\}$ if $b = 1$. Equation (3.7) implies that Lines 22–23 set $a_c = f_c$ if $b = 1$, and have no effect otherwise. Lines 24–27 have no effect since $s = c_2 = 0$.

Case c: Suppose that $\ell_1 \neq 0$ and $0 < c_2 \leq \ell_2$. Then Lines 8–11 set $\ell_2' = 2^{d_v}$, $b' = 1$, $s = c_2$ and $t = \ell_2$. Equation (3.7) implies that Lines 13–17 set $a_i = g_i$ for $i \in \{0, \ldots, 2^{d_v} c_1 - 1\}$. Thus, (3.8) implies that Lines 18–21 set $a_{2^{d_v} i + j} = h_{2^{d_v} i + j}$ and

$a_{2^{d_v}c_1+j} = g_{2^{d_v}c_1+j}$ for $i \in \{0, \ldots, c_1 - 1\}$ and $j \in \{c_2, \ldots, 2^{d_v} - 1\}$. Equation (3.7) implies that Lines 22–23 set $a_{2^{d_v}c_1+j} = g_{2^{d_v}c_1+j}$ for $j \in \{0, \ldots, c_2 - 1\}$, and $a_c = f_c$ if $b = 1$. Equation (3.8) implies that Lines 24–25 set $a_{2^{d_v}i+j} = h_{2^{d_v}i+j}$ for $i \in \{0, \ldots, c_1\}$ and $j \in \{0, \ldots, c_2 - 1\}$. Lines 26–27 have no effect since $s = c_2$.

Case d: Suppose that $\ell_1 \neq 0$ and $\ell_2 < c_2$. Then Lines 8–11 set $\ell_2' = 2^{d_v}$, $b' = 1$, $s = \ell_2$ and $t = c_2$. Equation (3.7) implies that Lines 13–17 set $a_i = g_i$ for $i \in \{0, \ldots, 2^{d_v}c_1 - 1\}$. Lines 18–19 have no effect since $t = c_2$. Equation (3.8) implies that Lines 20–21 set $a_{2^{d_v}i+j} = h_{2^{d_v}i+j}$ and $a_{2^{d_v}c_1+j} = g_{2^{d_v}c_1+j}$ for $i \in \{0, \ldots, c_1 - 1\}$ and $j \in \{c_2, \ldots, 2^{d_v} - 1\}$. Equation (3.7) implies Lines 22–23 set $a_{2^{d_v}c_1+j} = g_{2^{d_v}c_1+j}$ for $j \in \{0, \ldots, c_2 - 1\}$, and $a_c = f_c$ if $b = 1$. Equation (3.8) implies that Lines 24–27 set $a_{2^{d_v}i+j} = h_{2^{d_v}i+j}$ for $i \in \{0, \ldots, c_1\}$ and $j \in \{0, \ldots, c_2 - 1\}$.

In each of the four cases, Algorithm 3 terminates with $a_i = h_i$ for $i \in \{0, \ldots, c - 1\}$, and $a_c = f_c$ if $b = 1$, as required. Hence, for internal $v \in V$, if the algorithm produces the correct output whenever v_α or v_δ is given as an input, then it produces the correct output whenever v is given as an input. □

Algorithm 3 requires the same precomputations as the algorithms of Section 3.2. However, as the length of the vector on which the algorithm operates is no longer tied to the polynomial length ℓ, the auxiliary space requirements grow to $2^{n_v} - \ell + \mathcal{O}(n^2)$ field elements, where the last term accounts for the storage of the vectors μ and ν, and the precomputed elements $\varphi_v(u, \sigma_{v,i})$. The update to the vector μ in Line 15 of the algorithm performs $(c_1 + b' - 1)d_v = (\lfloor c/2^{d_v} \rfloor + b' - 1)d_v$ additions over all iterations of its containing loop. Therefore, as for the algorithms of Section 3.2, small values of d_v should be avoided when choosing a reduction tree.

Example 3.11. For β equal to a Cantor basis of dimension 15, and inputs $\ell \in \{1, \ldots, 2^{15}\}$, $c = \ell$ and $b = 0$, Figure 5 shows the maximum and minimum number of additions performed by Algorithm 3 over all possible reduction trees for the basis. The number of multiplications performed by the algorithm for each value of ℓ, which is independent of the choice of reduction tree, is also shown in the figure. As for Example 3.4, the number of additions performed for each value of ℓ is maximised by the tree with $\mathrm{Im}(d) \subseteq \{0, 1\}$, and minimised by the tree such that $d_v = 2^{\lceil \log_2 n_v \rceil - 1}$ for all internal $v \in V$.

Theorem 3.12. *Algorithm 3 performs at most*

$$\min\left(\frac{c + b - 1}{2}(\lceil \log_2 c + b \rceil - 1) + \ell - 1, 2^{n_v - 1}n_v\right)$$

multiplications in $\mathbb{F}$, and at most

$$\min\left(\frac{c + b - 1}{2}(3\lceil \log_2 c + b \rceil - 1) + \ell - 1, 2^{n_v - 1}(3n_v - 2) + 1\right)$$

additions in $\mathbb{F}$.

We split the proof of Theorem 3.12 into four lemmas, one for each bound. It is readily verified that the bounds hold if the input vertex is a leaf. Therefore, as (V, E) is a full binary tree, it is sufficient to show for each bound that if $v \in V$ is an internal vertex such that the bound holds whenever the input vertex is v_α or v_δ, then the bound holds whenever v is the input vertex.

Lemma 3.13. *Algorithm 3 performs at most $2^{n_v - 1}n_v$ multiplications in $\mathbb{F}$.*

Case c **Case d**

$$\begin{pmatrix} f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ f_8 & f_9 & f_{10} & f_{11} & f_{12} & f_{13} & f_{14} & f_{15} \\ f_{16} & f_{17} & h_{18} & h_{19} & h_{20} & h_{21} & h_{22} & h_{23} \\ h_{24} & h_{25} & h_{26} & h_{27} & * & * & * & * \end{pmatrix} \quad \begin{pmatrix} f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ f_8 & f_9 & f_{10} & f_{11} & f_{12} & f_{13} & f_{14} & f_{15} \\ f_{16} & f_{17} & f_{18} & f_{19} & f_{20} & f_{21} & f_{22} & f_{23} \\ h_{24} & h_{25} & h_{26} & h_{27} & * & * & * & * \end{pmatrix}$$

(4a) Initial contents.

$$\begin{pmatrix} g_0 & g_1 & g_2 & g_3 & g_4 & g_5 & g_6 & g_7 \\ g_8 & g_9 & g_{10} & g_{11} & g_{12} & g_{13} & g_{14} & g_{15} \\ f_{16} & f_{17} & h_{18} & h_{19} & h_{20} & h_{21} & h_{22} & h_{23} \\ h_{24} & h_{25} & h_{26} & h_{27} & * & * & * & * \end{pmatrix} \quad \begin{pmatrix} g_0 & g_1 & g_2 & g_3 & g_4 & g_5 & g_6 & g_7 \\ g_8 & g_9 & g_{10} & g_{11} & g_{12} & g_{13} & g_{14} & g_{15} \\ f_{16} & f_{17} & f_{18} & f_{19} & f_{20} & f_{21} & f_{22} & f_{23} \\ h_{24} & h_{25} & h_{26} & h_{27} & * & * & * & * \end{pmatrix}$$

(4b) After Lines 13–17 make recursive calls on the highlighted rows.

$$\begin{pmatrix} g_0 & g_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 \\ g_8 & g_9 & h_{10} & h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ f_{16} & f_{17} & g_{18} & g_{19} & g_{20} & g_{21} & g_{22} & g_{23} \\ h_{24} & h_{25} & * & * & * & * & * & * \end{pmatrix} \quad \begin{pmatrix} g_0 & g_1 & g_2 & g_3 & g_4 & g_5 & h_6 & h_7 \\ g_8 & g_9 & g_{10} & g_{11} & g_{12} & g_{13} & h_{14} & h_{15} \\ f_{16} & f_{17} & f_{18} & f_{19} & f_{20} & f_{21} & g_{22} & g_{23} \\ h_{24} & h_{25} & h_{26} & h_{27} & * & * & * & * \end{pmatrix}$$

(4c) After Lines 18–21 make recursive calls on the highlighted columns.

$$\begin{pmatrix} g_0 & g_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 \\ g_8 & g_9 & h_{10} & h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ g_{16} & g_{17} & (f_{18}) & * & * & * & * & * \\ h_{24} & h_{25} & * & * & * & * & * & * \end{pmatrix} \quad \begin{pmatrix} g_0 & g_1 & g_2 & g_3 & g_4 & g_5 & h_6 & h_7 \\ g_8 & g_9 & g_{10} & g_{11} & g_{12} & g_{13} & h_{14} & h_{15} \\ g_{16} & g_{17} & g_{18} & g_{19} & g_{20} & g_{21} & (f_{22}) & * \\ h_{24} & h_{25} & h_{26} & h_{27} & * & * & * & * \end{pmatrix}$$

(4d) After Lines 23 makes a recursive call on the highlighted row.

$$\begin{pmatrix} h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 \\ h_8 & h_9 & h_{10} & h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{16} & h_{17} & (f_{18}) & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{pmatrix} \quad \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 \\ h_8 & h_9 & h_{10} & h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{16} & h_{17} & h_{18} & h_{19} & h_{20} & h_{21} & (f_{22}) & * \\ * & * & * & * & * & * & * & * \end{pmatrix}$$

(4e) After Lines 24–27 make recursive calls on the highlighted columns.

FIGURE 4. Evolution of the vector $(a_0, \ldots, a_{2^{n_v}-1})$ during Algorithm 3 for $n_v = 5$, $d_v = 3$, $c = 18$, $\ell = 28$ (Case c), and $n_v = 5$, $d_v = 3$, $c = 22$, $\ell = 28$ (Case d).

Proof. Suppose that Algorithm 3 is called on an internal vertex $v \in V$ such that the bound of the lemma holds whenever the input vertex is v_α or v_δ. Then Lines 13–17 of the algorithm perform at most $c_1 2^{d_v-1} d_v \leq (2^{n_v-d_v} - b')2^{d_v-1} d_v$ multiplications, since $c_1 \leq 2^{n_v-d_v}$ with equality implying $c_2 = b = b' = 0$. If $\ell \geq 2^{d_v}$, then $\ell_2' = 2^{d_v} \geq t \geq c_2$. If $\ell < 2^{d_v}$, then $\ell = \ell_2 = \ell_2' \geq c = c_2$. Thus, the inequalities $c_2 \leq t \leq \ell_2' \leq 2^{d_v}$ hold in either case. It follows that Lines 18–21 perform at most $(2^{d_v} - c_2)2^{n_v-d_v-1}(n_v - d_v)$ multiplications. Lines 22–23 performs at most

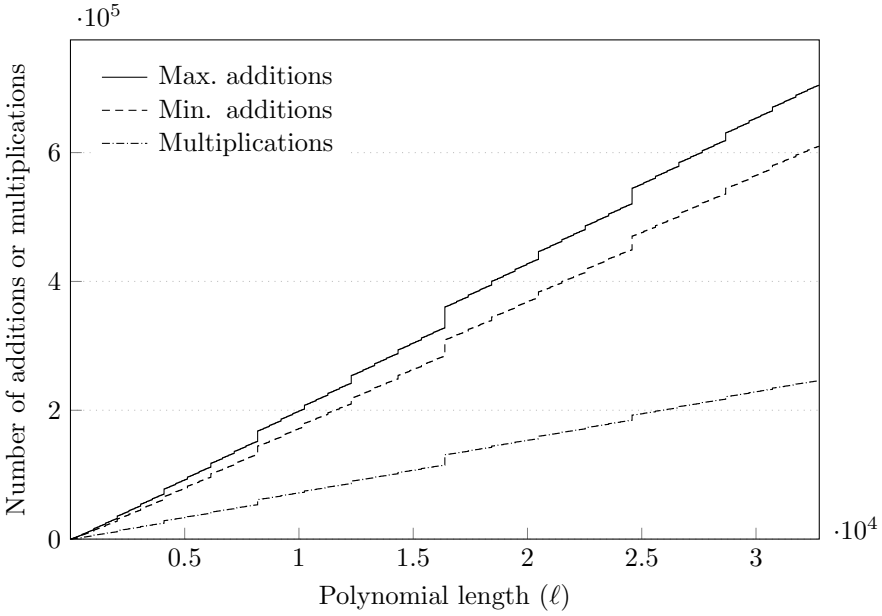

FIGURE 5. Maximum and minimum number of operations performed by Algorithm 3 (Algorithm 4) for Cantor bases of dimension 15, with parameters $c = \ell$ and $b = 0$ (respectively, $c = \ell$).

$b'2^{d_v-1}d_v$ multiplications, while Lines 24–27 perform at most $c_2 2^{n_v-d_v-1}(n_v - d_v)$. Summing these contributions, it follows that Algorithm 3 performs at most $2^{n_v-1}n_v$ multiplications. □

Lemma 3.14. *Algorithm 3 performs at most*

$$\frac{c+b-1}{2}(\lceil \log_2 c + b \rceil - 1) + \ell - 1 \tag{3.9}$$

multiplications in $\mathbb{F}$.

Proof. Suppose that Algorithm 3 is called on an internal vertex $v \in V$ such that the bound of the lemma holds whenever the input vertex is v_α or v_δ. Then Lemma 3.13 implies that Lines 13–17 perform at most

$$c_1 2^{d_v-1}d_v = \frac{c-c_2}{2}d_v \tag{3.10}$$

multiplications. As $c_2 \le t \le \ell_2' \le 2^{d_v}$, Lines 18–21 perform at most

$$\left(2^{d_v} - c_2\right)\frac{c_1+b'-1}{2}(\lceil \log_2 c_1 + b' \rceil - 1) + (\ell_2' - c_2)(\ell_1 - 1) + t - c_2 \tag{3.11}$$

multiplications. Lines 22–23 perform at most

$$b'\left(\frac{c_2+b-1}{2}(\lceil \log_2 \max(c_2+b,1) \rceil - 1) + \ell_2' - 1\right) \tag{3.12}$$

multiplications, while Lines 24–27 perform at most

$$c_2\left(\frac{c_1}{2}(\lceil \log_2 c_1 + 1 \rceil - 1) + \ell_1 - 1\right) + s \tag{3.13}$$

multiplications. We show that the sum of these contributions is bounded by (3.9).

Suppose that $b' = 1$. Then $1 \leq c_2 + b \leq c + b$. Thus, the sum of (3.10)–(3.13) in this case is at most

$$\frac{c - c_2}{2}(\lceil \log_2 c_1 + 1 \rceil + d_v - 1) + \frac{c_2 + b - 1}{2}(\lceil \log_2 c + b \rceil - 1) + \ell - 1, \tag{3.14}$$

since

$$\ell_2'(\ell_1 - 1) + s + t - c_2 = \ell_2'(\ell_1 - 1) + \ell_2 \leq 2^{d_v}\ell_1 + \ell_2 - \ell_2' = \ell - \ell_2'.$$

If $c_1 = 0$, then $c_2 = c$. If $c_1 \neq 0$ and $c_2 \neq 0$, then

$$\lceil \log_2 c_1 + 1 \rceil = \left\lceil \log_2 \lceil c/2^{d_v} \rceil \right\rceil = \lceil \log_2 c \rceil - d_v \leq \lceil \log_2 c + b \rceil - d_v.$$

If $c_1 \neq 0$ and $c_2 = 0$, then $b = 1$ since $c_2 + b \geq 1$, and

$$\lceil \log_2 c_1 + 1 \rceil = \left\lceil \log_2 c/2^{d_v} + 1 \right\rceil = \lceil \log_2 c + 1 \rceil - d_v = \lceil \log_2 c + b \rceil - d_v.$$

In all three of these cases, substituting into (3.14) yields (3.9).

Suppose now that $b' = 0$. Then $c_2 = 0$ and $b = 0$. As $c + b \geq 1$, it follows that $c_1 = c/2^{d_v} \neq 0$. Thus, $\ell_2' = 2^{d_v}$, since $\ell \geq c \geq 2^{d_v}$. Therefore, the sum of (3.10)–(3.13) in this case is at most

$$\begin{aligned}
&\frac{c}{2}d_v + \frac{c - 2^{d_v}}{2}(\lceil \log_2 c_1 \rceil - 1) + \ell - 2^{d_v} \\
&\qquad = \frac{c}{2}d_v + \frac{c-1}{2}(\lceil \log_2 c \rceil - d_v - 1) - \frac{2^{d_v} - 1}{2}(\lceil \log_2 c_1 \rceil - 1) + \ell - 2^{d_v} \\
&\qquad \leq \frac{c}{2}d_v + \frac{c-1}{2}(\lceil \log_2 c \rceil - d_v - 1) + \frac{2^{d_v} - 1}{2} + \ell - 2^{d_v} \\
&\qquad = \frac{c-1}{2}(\lceil \log_2 c \rceil - 1) + \ell - 1 - \frac{1}{2}\big(2^{d_v} - d_v - 1\big) \\
&\qquad \leq \frac{c + b - 1}{2}(\lceil \log_2 c + b \rceil - 1) + \ell - 1,
\end{aligned}$$

as required. □

Lemma 3.15. *Algorithm 3 performs at most* $2^{n_v - 1}(3n_v - 2) + 1$ *additions in* $\mathbb{F}$.

Proof. Suppose that Algorithm 3 is called on an internal vertex $v \in V$ such that the bound of the lemma holds whenever the input vertex is v_α or v_δ. Then Lines 13–17 of the algorithm perform at most

$$\begin{aligned}
&c_1\big(2^{d_v - 1}(3d_v - 2) + 1 + d_v\big) - (1 - b')d_v \\
&\qquad\qquad \leq \big(2^{n_v - d_v} - b'\big)\big(2^{d_v - 1}(3d_v - 2) + 1\big) + \big(2^{n_v - d_v} - 1\big)d_v
\end{aligned}$$

additions, since $c_1 \leq 2^{n_v - d_v}$ with equality implying that $c_2 = b = b' = 0$. As $c_2 \leq t \leq \ell_2' \leq 2^{d_v}$, Lines 18–21 perform at most $(2^{d_v} - c_2)(2^{n_v - d_v - 1}(3(n_v - d_v) - 2) + 1)$ additions. Lines 22–23 perform at most $b'(2^{d_v - 1}(3d_v - 2) + 1)$ additions, while Lines 24–27 perform at most $c_2(2^{n_v - d_v - 1}(3(n_v - d_v) - 2) + 1)$ additions. Summing these contributions, it follows that Algorithm 3 performs at most

$$2^{n_v - 1}(3n_v - 2) + 1 - \big(2^{d_v} - d_v - 1\big)\big(2^{n_v - d_v} - 1\big) \leq 2^{n_v - 1}(3n_v - 2) + 1$$

additions. □

Lemma 3.16. *Algorithm 3 performs at most*

$$\frac{c+b-1}{2}(3\lceil\log_2 c+b\rceil-1)+\ell-1 \tag{3.15}$$

additions in $\mathbb{F}$.

Proof. Suppose that Algorithm 3 is called on an internal vertex $v \in V$ such that the bound of the lemma holds whenever the input vertex is v_α or v_δ. Then Lemma 3.15 implies that Lines 13–17 perform at most

$$c_1\big(2^{d_v-1}(3d_v-2)+1+d_v\big)-(1-b')d_v \le \frac{c-c_2}{2}3d_v-(1-b')d_v \tag{3.16}$$

additions. As $c_2 \le t \le \ell_2' \le 2^{d_v}$, Lines 18–21 perform at most

$$\big(2^{d_v}-c_2\big)\frac{c_1+b'-1}{2}(3\lceil\log_2 c_1+b'\rceil-1)+(\ell_2'-c_2)(\ell_1-1)+t-c_2 \tag{3.17}$$

additions. Lines 22–23 perform at most

$$b'\left(\frac{c_2+b-1}{2}(3\lceil\log_2\max(c_2+b,1)\rceil-1)+\ell_2'-1\right) \tag{3.18}$$

additions, while Lines 24–27 perform at most

$$c_2\Big(\frac{c_1}{2}(3\lceil\log_2 c_1+1\rceil-1)+\ell_1-1\Big)+s \tag{3.19}$$

additions. We show that the sum of these contributions is bounded by (3.15).

Suppose that $b'=1$. Then the sum of (3.16)–(3.19) is at most

$$\frac{c-c_2}{2}(3\lceil\log_2 c_1+1\rceil+3d_v-1)+\frac{c_2+b-1}{2}(3\lceil\log_2 c+b\rceil-1)+\ell-1. \tag{3.20}$$

If $c_1=0$, then $c_2=c$. If not, then $\lceil\log_2 c_1+1\rceil \le \lceil\log_2 c+b\rceil-d_v$. In either case, substituting into (3.20) yields (3.15).

Suppose now that $b'=0$. Then $c_2=b=0$, $c_1\neq 0$ and $\ell_2'=2^{d_v}$. It follows that the sum of (3.16)–(3.19) in this case is at most

$$\begin{aligned}
&\frac{c}{2}3d_v-d_v+\frac{c-2^{d_v}}{2}(3\lceil\log_2 c_1\rceil-1)+\ell-2^{d_v}\\
&\qquad\le \frac{c-1}{2}3d_v+\frac{1}{2}d_v+\frac{c-1}{2}(3\lceil\log_2 c\rceil-3d_v-1)+\frac{2^{d_v}-1}{2}+\ell-2^{d_v}\\
&\qquad= \frac{c-1}{2}(3\lceil\log_2 c\rceil-1)+\ell-1-\frac{1}{2}\big(2^{d_v}-d_v-1\big)\\
&\qquad\le \frac{c+b-1}{2}(3\lceil\log_2 c+b\rceil-1)+\ell-1,
\end{aligned}$$

as required. □

3.4. **Conversion from the Lin–Chung–Han basis to the Lagrange basis.** We propose Algorithm 4 for converting from the LCH basis to the Lagrange basis. The parameter c plays a different role than in Algorithm 3, with its function being to specify the number Lagrange basis coefficients returned by the algorithm, rather than to specify a mixture of coefficients. For conversion from the LCH basis to the Lagrange basis, Algorithm 4 is initially called with $c = 2^{n_v}$. Smaller initial values of c are relevant, for example, when using the algorithm within the Hermite evaluation algorithm of Coxon [12].

Theorem 3.17. *Algorithm 4 is correct.*

Algorithm 4 $\mathsf{X2L}(v, (\varphi_v(u,\lambda))_{u\in L_v}, c, \ell, (a_0,\dots,a_{2^{n_v}-1}))$

Input: a vertex $v \in V$, the vector $(\varphi_v(u,\lambda))_{u\in L_v} \in \mathbb{F}^{n_v}$ for some $\lambda \in \mathbb{F}$, $c, \ell \in \{1,2,\dots,2^{n_v}\}$, and $a_i = h_i \in \mathbb{F}$ for $i \in \{0,\dots,\ell-1\}$.

Output: $a_i = f_i \in \mathbb{F}$ for $i \in \{0,\dots,c-1\}$ such that (3.6) holds for some $f_c,\dots,f_{2^{n_v}-1} \in \mathbb{F}$.

1: **if** v is a leaf **then**
2: **if** $c=2$ and $\ell=2$ **then** $a_0 \leftarrow a_0 + \varphi_v(v,\lambda)a_1$, $a_1 \leftarrow a_0 + a_1$
3: **if** $c=1$ and $\ell=2$ **then** $a_0 \leftarrow a_0 + \varphi_v(v,\lambda)a_1$
4: **if** $c=2$ and $\ell=1$ **then** $a_1 \leftarrow a_0$
5: **return**
6: $c_1 \leftarrow \lceil c/2^{d_v} \rceil - 1$, $c_2 \leftarrow c - 2^{d_v}c_1$
7: $\ell_1 \leftarrow \lfloor \ell/2^{d_v} \rfloor$, $\ell_2 \leftarrow \ell - 2^{d_v}\ell_1$ $\ell_2' \leftarrow \min(2^{d_v}, \ell)$
8: $\mu \leftarrow (\varphi_v(u,\lambda))_{u\in L_{v_\alpha}}$, $\nu \leftarrow (\varphi_v(u,\lambda))_{u\in L_{v_\delta}}$
9: **for** $j = 0,\dots,\ell_2 - 1$ **do**
10: $\mathsf{X2L}(v_\delta, \nu, c_1+1, \ell_1+1, (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}(2^{n_v-d_v}-1)+j}))$
11: **for** $j = \ell_2,\dots,\ell_2' - 1$ **do**
12: $\mathsf{X2L}(v_\delta, \nu, c_1+1, \ell_1, (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}(2^{n_v-d_v}-1)+j}))$
13: **for** $i = 0,\dots,c_1 - 1$ **do**
14: $\mathsf{X2L}(v_\alpha, \mu, 2^{d_v}, \ell_2', (a_{2^{d_v}i}, a_{2^{d_v}i+1}, \dots, a_{2^{d_v}(i+1)-1}))$
15: $\mu \leftarrow \mu + (\varphi_v(u, \sigma_{v,\Delta(i)}))_{u\in L_{v_\alpha}}$
16: $\mathsf{X2L}(v_\alpha, \mu, c_2, \ell_2', (a_{2^{d_v}c_1}, a_{2^{d_v}c_1+1}, \dots, a_{2^{d_v}(c_1+1)-1}))$

Proof. Table 3 displays the input and output requirements of Algorithm 4 when the input vertex v is a leaf, as well as showing the output of the algorithm as computed by Lines 1–5. The elements f_i and h_i that appear in a row of the table are the coefficients of (3.6) for the specified value of ℓ. Elements denoted by asterisks are unspecified by the algorithm. As v is a leaf, the coefficients of (3.6) satisfy $f_0 = h_0 + \varphi_v(v,\lambda)h_1$ and $f_1 = h_1 + (h_0 + \varphi_v(v,\lambda)h_1)$ if $\ell = 2$, and $f_0 = f_1 = h_0$ if $\ell = 1$. Using these equation, one can readily verify that the computed output agrees with the required output for all inputs. Consequently, Algorithm 4 produces the correct output whenever the input vertex is a leaf. Therefore, as (V,E) is a full binary tree, it is sufficient to show that for all internal $v \in V$, if the algorithm produces the correct output whenever v_α or v_δ is given as an input, then it produces the correct output whenever v is given as an input.

Input				Required output		Computed output	
c	ℓ	a_0	a_1	a_0	a_1	a_0	a_1
2	2	h_0	h_1	f_0	f_1	$h_0 + \varphi_v(v,\lambda)h_1$	$h_1 + (h_0 + \varphi_v(v,\lambda)h_1)$
1	2	h_0	h_1	f_0	$*$	$h_0 + \varphi_v(v,\lambda)h_1$	$*$
2	1	h_0	$*$	f_0	f_1	h_0	h_0
1	1	h_0	$*$	f_0	$*$	h_0	$*$

TABLE 3. Required and computed outputs of Algorithm 4 when v is a leaf.

Let $v \in V$ be an internal vertex and suppose that Algorithm 4 produces the correct output whenever v_α or v_δ is given as an input. Suppose that the algorithm

is called on v, the vector $(\varphi_v(u,\lambda))_{u\in L_v}$ for some $\lambda \in \mathbb{F}$, integers $c, \ell \in \{1, 2, \ldots, 2^{n_v}\}$ and $(a_0, \ldots, a_{2^{n_v}-1})$, with $a_i = h_i \in \mathbb{F}$ for $i \in \{0, \ldots, \ell-1\}$. Then there exist unique elements $f_0, \ldots, f_{2^{n_v}-1} \in \mathbb{F}$ such that (3.6) holds. In-turn, Lemma 3.9 implies that there exist unique elements $g_0, \ldots, g_{2^{n_v}-1} \in \mathbb{F}$ such that (3.7) and (3.8) hold.

Once again repeating arguments from the proof of Theorem 3.3 shows that $\nu = (\varphi_{v_\delta}(u,\eta))_{u\in L_{v_\delta}}$ for the recursive calls of Lines 9–12, $\mu = (\varphi_{v_\alpha}(u, \lambda + \omega_{\gamma_v,i}))_{u\in L_{v_\alpha}}$ each time the recursive call of Line 14 is performed, and finally $\mu = (\varphi_{v_\alpha}(u, \lambda + \omega_{\gamma_v,c_1}))_{u\in L_{v_\alpha}}$ for the recursive call of Line 16. Thus, (3.8) and the assumption that the algorithm produces the correct output whenever v_δ is given as an input imply that Lines 9–12 set $a_{2^{d_v}i+j} = g_{2^{d_v}i+j}$ for $i \in \{0, \ldots, c_1\}$ and $j \in \{0, \ldots, \min(2^{d_v}, \ell)-1\}$. Consequently, (3.7) and the assumption that the algorithm produces the correct output whenever v_α is given as an input imply that Lines 13–15 set $a_i = f_i$ for $i \in \{0, \ldots, 2^{d_v}c_1 - 1\}$, and that Line 16 sets $a_i = f_i$ for $i \in \{2^{d_v}c_1, \ldots, c-1\}$. Therefore, the algorithm terminates with $a_i = f_i$ for $i \in \{0, \ldots, c-1\}$, as required. Hence, for internal $v \in V$, if the algorithm produces the correct output whenever v_α or v_δ is given as an input, then it produces the correct output whenever v is given as an input. □

Algorithm 4 requires the same precomputations as the algorithms of Sections 3.2 and 3.3, while the algorithm requires auxiliary space for $2^{n_v} - \max(c, \ell) + \mathcal{O}(n^2)$ field elements. Small values of d_v should once again be avoided when choosing a reduction tree for the algorithm, in order to help reduce the number of additions performed by the updates to the vector μ in Line 15.

Example 3.18. For β equal to a Cantor basis of dimension 15, and inputs $\ell \in \{1, \ldots, 2^{15}\}$ and $c = \ell$, Figure 5 also shows the maximum and minimum number of additions performed by Algorithm 4 over all possible reduction trees for the basis, as well as the number of multiplications performed by the algorithm for all such trees. Thus, Algorithm 3 with $c = \ell$ and $b = 0$ performs the same number of operations for both extremes (see Example 3.11). As for Examples 3.4 and 3.11, the maximum and minimum number of additions performed for each ℓ are given respectively by the trees with $d_v = 1$ and $d_v = 2^{\lceil \log_2 n_v \rceil - 1}$ for all internal $v \in V$.

Theorem 3.19. *Algorithm 4 performs at most*

$$\min\left(\frac{c-1}{2}(\lceil \log_2 c \rceil - 1) + \ell - 1, 2^{n_v-1}n_v\right)$$

multiplications in $\mathbb{F}$, and at most

$$\min\left(\frac{c-1}{2}(3\lceil \log_2 c \rceil - 1) + \ell - 1, 2^{n_v-1}(3n_v - 2) + 1\right)$$

additions in $\mathbb{F}$.

We split the proof of Theorem 3.19 into four lemmas, one for each bound. It is readily verified that the bounds hold if the input vertex is a leaf. Therefore, as (V, E) is a full binary tree, it is sufficient to show for each bound that if $v \in V$ is an internal vertex such that the bound holds whenever the input vertex is v_α or v_δ, then the bound holds whenever v is the input vertex.

Lemma 3.20. *Algorithm 4 performs at most $2^{n_v-1}n_v$ multiplications in $\mathbb{F}$.*

Proof. Suppose that Algorithm 4 is called on an internal vertex $v \in V$ such that the bound of the lemma holds whenever the input vertex is v_α or v_δ. Then Lines 9–12 of the algorithm perform at most $\ell_2' 2^{n_v - d_v - 1}(n_v - d_v) \leq 2^{n_v-1}(n_v - d_v)$ multiplications, while Lines 13–16 perform at most $(c_1 + 1)2^{d_v - 1} d_v \leq 2^{n_v-1} d_v$ multiplications. Summing theses contributions, it follows that Algorithm 4 performs at most $2^{n_v-1} n_v$ multiplications. □

Lemma 3.21. *Algorithm 4 performs at most* $(c-1)(\lceil \log_2 c \rceil - 1)/2 + \ell - 1$ *multiplications in* $\mathbb{F}$.

Proof. Suppose that Algorithm 4 is called on an internal vertex $v \in V$ such that the bound of the lemma holds whenever the input vertex is v_α or v_δ. Then Lines 9–12 perform at most

$$2^{d_v} \frac{c_1}{2}(\lceil \log_2 c_1 + 1 \rceil - 1) + \ell_2'(\ell_1 - 1) + \ell_2 \leq \frac{c - c_2}{2}(\lceil \log_2 c_1 + 1 \rceil - 1) + \ell - \ell_2'$$

multiplications. Lemma 3.20 implies that Lines 13–16 perform at most

$$c_1 2^{d_v - 1} d_v + \frac{c_2 - 1}{2}(\lceil \log_2 c_2 \rceil - 1) + \ell_2' - 1 = \frac{c - c_2}{2} d_v + \frac{c_2 - 1}{2}(\lceil \log_2 c_2 \rceil - 1) + \ell_2' - 1$$

multiplications. As $c_2 \leq c$, it follows that Algorithm 4 performs at most

$$\frac{c - c_2}{2}(\lceil \log_2 c_1 + 1 \rceil + d_v - 1) + \frac{c_2 - 1}{2}(\lceil \log_2 c \rceil - 1) + \ell - 1$$

multiplications. Hence, Algorithm 4 performs at most $(c-1)(\lceil \log_2 c \rceil - 1)/2 + \ell - 1$ multiplications, since $c_2 = c$ if $c_1 = 0$, and $\lceil \log_2 c_1 + 1 \rceil = \lceil \log_2 c \rceil - d_v$ otherwise. □

Lemma 3.22. *Algorithm 4 performs at most* $2^{n_v-1}(3n_v - 2) + 1$ *additions in* $\mathbb{F}$.

Proof. Suppose that Algorithm 4 is called on an internal vertex $v \in V$ such that the bound of the lemma holds whenever the input vertex is v_α or v_δ. Then Lines 9–12 of the algorithm perform at most

$$\ell_2'\big(2^{n_v - d_v - 1}(3(n_v - d_v) - 2) + 1\big) \leq 2^{n_v-1}(3n_v - 3d_v) - 2^{d_v}\big(2^{n_v - d_v} - 1\big)$$

additions. Lines 13–16 perform at most

$$(c_1 + 1)\big(2^{d_v - 1}(3d_v - 2) + d_v + 1\big) - d_v \leq 2^{n_v-1}(3d_v - 2) + 1 + (d_v + 1)\big(2^{n_v - d_v} - 1\big)$$

additions. It follows that Algorithm 4 performs at most

$$2^{n_v-1}(3n_v - 2) + 1 - \big(2^{d_v} - d_v - 1\big)\big(2^{n_v - d_v} - 1\big) \leq 2^{n_v-1}(3n_v - 2) + 1$$

additions. □

Lemma 3.23. *Algorithm 4 performs at most* $(c-1)(3\lceil \log_2 c \rceil - 1)/2 + \ell - 1$ *additions in* $\mathbb{F}$.

Proof. Suppose that Algorithm 4 is called on an internal vertex $v \in V$ such that the bound of the lemma holds whenever the input vertex is v_α or v_δ. Then Lines 9–12 perform at most

$$2^{d_v} \frac{c_1}{2}(3\lceil \log_2 c_1 + 1 \rceil - 1) + \ell_2'(\ell_1 - 1) + \ell_2 \leq \frac{c - c_2}{2}(3\lceil \log_2 c_1 + 1 \rceil - 1) + \ell - \ell_2'$$

additions. Lemma 3.22 implies that Lines 13–15 perform at most

$$c_1\big(2^{d_v - 1}(3d_v - 2) + 1 + d_v\big) = \frac{c - c_2}{2} 3d_v - c_1\big(2^{d_v} - d_v - 1\big) \leq \frac{c - c_2}{2} 3d_v$$

additions. Line 16 performs at most

$$\frac{c_2 - 1}{2}(3\lceil \log_2 c_2 \rceil - 1) + \ell_2' - 1$$

additions. As $c_2 \leq c$, it follows that Algorithm 4 performs at most

$$\frac{c - c_2}{2}(3\lceil \log_2 c_1 + 1 \rceil + 3d_v - 1) + \frac{c_2 - 1}{2}(3\lceil \log_2 c \rceil - 1) + \ell - 1$$

additions. Hence, Algorithm 4 performs at most $(c-1)(3\lceil \log_2 c \rceil - 1)/2 + \ell - 1$ additions, since $c_2 = c$ if $c_1 = 0$, and $\lceil \log_2 c_1 + 1 \rceil = \lceil \log_2 c \rceil - d_v$ otherwise. □

3.5. **Interlude: generalised Taylor expansion.** The generalised Taylor expansion of a polynomial $F \in \mathbb{F}[x]$ at a degree $t \geq 1$ polynomial $T \in \mathbb{F}[x]$, also called its T-adic expansion, is the series expansion

$$F = F_0 + F_1 T + F_2 T^2 + \cdots$$

such that $F_i \in \mathbb{F}[x]_t$ for $i \in \mathbb{N}$. Gao and Mateer [14, Section II] provide a fast algorithm for computing the coefficients of the Taylor expansion when $T = x^t - x$ with $t \geq 2$. The algorithm is then utilised as part of their additive FFT algorithms. Our algorithm for converting from the monomial basis to the LCH basis similarly relies on their generalised Taylor expansion algorithm. Consequently, we make a brief aside to recall their algorithm.

The algorithm of Gao and Mateer can be viewed as a specialisation of the recursive algorithm of von zur Gathen [27] that takes advantage of easy division by $(x^t - x)^{2^k} = x^{2^k t} - x^{2^k}$ in characteristic two. We present a nonrecursive version of their algorithm modelled on the basis conversion algorithms of van der Hoeven and Schost [26, Section 2.2]. We also present the inverse algorithm, which recovers a polynomial from the coefficients of its Taylor expansion at $x^t - x$, as it is required by our algorithm for converting from the LCH basis to monomial basis. Finally, we derive a bound on the complexity of both algorithms that is tighter than the one provided by Gao and Mateer.

Let $F \in \mathbb{F}[x]_\ell$ and $t \geq 2$ be an integer. For $k \in \mathbb{N}$, define $F_{k,0}, F_{k,1}, \ldots \in \mathbb{F}[x]_{2^k t}$ by the equation

$$F = \sum_{i \in \mathbb{N}} F_{k,i}\big(x^t - x\big)^{2^k i}. \tag{3.21}$$

Then $F_{0,0}, F_{0,1}, \ldots$ are the coefficients of the Taylor expansion at $x^t - x$, while $F_{k,0} = F$ for $k \geq \lceil \log_2 \lceil \ell/t \rceil \rceil$. By grouping terms of indices $2i$ and $2i+1$ in (3.21), it follows that

$$F = \sum_{i \in \mathbb{N}} \Big(F_{k,2i} + F_{k,2i+1}\Big(x^{2^k t} - x^{2^k}\Big)\Big)\big(x^t - x\big)^{2^{k+1} i} \quad \text{for } k \in \mathbb{N}.$$

Thus, we obtain the recursive formula

$$F_{k+1,i} = F_{k,2i} + x^{2^k} F_{k,2i+1} + x^{2^k t} F_{k,2i+1} \quad \text{for } k, i \in \mathbb{N}.$$

Given $F_{k,2i}$ and $F_{k,2i+1}$ on the monomial basis, the recursive formula allows $F_{k+1,i}$ to be readily computed on the monomial basis. The formula also allows this computation to be easily inverted. Therefore, given the Taylor coefficients $F_{0,0}, F_{0,1}, \ldots$ on the monomial basis, we can efficiently compute $F = F_{\lceil \log_2 \lceil \ell/t \rceil \rceil, 0}$ on the monomial basis by means of the recursive formula, and vice versa. Using this observation, we obtain Algorithms 5 and 6.

Algorithm 5 TaylorExpansion$(t, \ell, (a_0, \dots, a_{\ell-1}))$

Input: Integers $t \geq 2$ and $\ell \geq 1$, and $a_i = f_i \in \mathbb{F}$ for $i \in \{0, \dots, \ell-1\}$.
Output: $a_i = c_i$ for $i \in \{0, \dots, \ell-1\}$ such that

$$\sum_{i=0}^{\lceil \ell/t \rceil - 1} \left(\sum_{j=0}^{\min(\ell - ti, t) - 1} c_{ti+j} x^j \right) \left(x^t - x\right)^i = \sum_{i=0}^{\ell-1} f_i x^i. \tag{3.22}$$

1: **for** $k = \lceil \log_2 \lceil \ell/t \rceil \rceil - 1, \dots, 0$ **do**
2: $\quad \ell_1 \leftarrow \lfloor \ell/(2^{k+1}t) \rfloor$, $\ell_2 \leftarrow \ell - 2^{k+1} t \ell_1$
3: $\quad$ **for** $i = 0, \dots, \ell_1 - 1$ **do**
4: $\quad\quad$ **for** $j = 2^k t - 1, \dots, 0$ **do**
5: $\quad\quad\quad a_{2^k t(2i) + 2^k + j} \leftarrow a_{2^k t(2i) + 2^k + j} + a_{2^k t(2i+1) + j}$
6: $\quad$ **for** $j = \ell_2 - 2^k t - 1, \dots, 0$ **do**
7: $\quad\quad a_{2^k t(2\ell_1) + 2^k + j} \leftarrow a_{2^k t(2\ell_1) + 2^k + j} + a_{2^k t(2\ell_1 + 1) + j}$

Algorithm 6 InverseTaylorExpansion$(t, \ell, (a_0, \dots, a_{\ell-1}))$

Input: Integers $t \geq 2$ and $\ell \geq 1$, and $a_i = c_i \in \mathbb{F}$ for $i \in \{0, \dots, \ell-1\}$.
Output: $a_i = f_i$ $i \in \{0, \dots, \ell-1\}$ such that (3.22) holds.

1: **for** $k = 0, \dots, \lceil \log_2 \lceil \ell/t \rceil \rceil - 1$ **do**
2: $\quad \ell_1 \leftarrow \lfloor \ell/(2^{k+1}t) \rfloor$, $\ell_2 \leftarrow \ell - 2^{k+1} t \ell_1$
3: $\quad$ **for** $i = 0, \dots, \ell_1 - 1$ **do**
4: $\quad\quad$ **for** $j = 0, \dots, 2^k t - 1$ **do**
5: $\quad\quad\quad a_{2^k t(2i) + 2^k + j} \leftarrow a_{2^k t(2i) + 2^k + j} + a_{2^k t(2i+1) + j}$
6: $\quad$ **for** $j = 0, \dots, \ell_2 - 2^k t - 1$ **do**
7: $\quad\quad a_{2^k t(2\ell_1) + 2^k + j} \leftarrow a_{2^k t(2\ell_1) + 2^k + j} + a_{2^k t(2\ell_1 + 1) + j}$

Lemma 3.24. *Algorithms 5 and 6 perform at most* $\lfloor \ell/2 \rfloor \lceil \log_2 \lceil \ell/t \rceil \rceil$ *additions in* $\mathbb{F}$.

Proof. For each $k \in \{0, \dots, \lceil \log_2 \lceil \ell/t \rceil \rceil - 1\}$, Lines 2–7 of either algorithm perform at most

$$2^k t \ell_1 + \max(\ell_2 - 2^k t, 0) \leq 2^k t \ell_1 + \max(\ell_2 - \lceil \ell_2/2 \rceil, \lfloor \ell_2/2 \rfloor) = 2^k t \ell_1 + \lfloor \ell_2/2 \rfloor = \lfloor \ell/2 \rfloor$$

additions in $\mathbb{F}$. □

3.6. **Conversion between the Lin–Chung–Han and monomial bases.** We use Lemma 2.1 to provide algorithms for converting between the monomial basis and the "twisted" LCH basis $\{X_{\beta_v,0}(\beta_{v,0}x), \dots, X_{\beta_v,\ell-1}(\beta_{v,0}x)\}$ of $\mathbb{F}[x]_\ell$, for $v \in V$ and $\ell \in \{1, \dots, 2^{n_v}\}$. Conversions between the LCH and monomial bases then require at most an additional $\max(2\ell - 3, 0)$ multiplications for performing the substitution $x \mapsto x/\beta_{v,0}$ or $x \mapsto \beta_{v,0}x$. In particular, no additional multiplications are required if β is a Cantor basis, since $\beta_{v,0} = 1$ for all $v \in V$ (see Remark 3.1). We base the conversion algorithms on the following analogue of Lemma 2.2.

Lemma 3.25. *Let* $v \in V$ *be an internal vertex and* $\ell \in \{1, \dots, 2^{n_v}\}$. *Suppose that* $h_0, \dots, h_{\ell-1}, \lambda, g_0, \dots, g_{\ell-1}, c_0, \dots, c_{\ell-1} \in \mathbb{F}$ *satisfy*

$$\sum_{j=0}^{\min\left(\ell - 2^{d_v} i, 2^{d_v}\right) - 1} g_{2^{d_v} i + j} x^j = \sum_{j=0}^{\min\left(\ell - 2^{d_v} i, 2^{d_v}\right) - 1} h_{2^{d_v} i + j} X_{\beta_{v_\alpha}, j}(\beta_{v_\alpha,0} x) \tag{3.23}$$

for $i \in \{0, \dots, \lceil \ell/2^{d_v} \rceil - 1\}$, *and*

$$\sum_{i=0}^{\lceil (\ell-j)/2^{d_v} \rceil - 1} c_{2^{d_v} i + j} x^i = \sum_{i=0}^{\lceil (\ell-j)/2^{d_v} \rceil - 1} g_{2^{d_v} i + j} X_{\beta_{v_\delta}, i}(\beta_{v_\delta, 0} x) \tag{3.24}$$

for $j \in \{0, \dots, \min(2^{d_v}, \ell) - 1\}$. *Then*

$$\sum_{i=0}^{\lceil \ell/2^{d_v} \rceil - 1} \left(\sum_{j=0}^{\min(\ell - 2^{d_v} i, 2^{d_v}) - 1} c_{2^{d_v} i + j} x^j \right) \left(\frac{x^{2^{d_v}} - x}{\beta_{v_\delta, 0}} \right)^i = \sum_{i=0}^{\ell-1} h_i X_{\beta_v, i}(\beta_{v,0} x). \tag{3.25}$$

Proof. Let $v \in V$ be an internal vertex and $\ell \in \{1, \dots, 2^{n_v}\}$. Suppose that $h_0, \dots, h_{\ell-1}, \lambda, g_0, \dots, g_{\ell-1}, c_0, \dots, c_{\ell-1} \in \mathbb{F}$ satisfy equations (3.23) and (3.24). Then $\beta_{v,i}/\beta_{v,0} \in \mathbb{F}_{2^{d_v}}$ for $i \in \{0, \dots, d_v - 1\}$, since (V, E) is a reduction tree for β. Thus, Lemma 2.1 implies that

$$\sum_{i=0}^{\ell-1} h_i X_{\beta_v, i}(\beta_{v,0} x) = \sum_{i=0}^{\lceil \ell/2^{d_v} \rceil - 1} \left(\sum_{j=0}^{\min(\ell - 2^{d_v} i, 2^{d_v}) - 1} h_{2^{d_v} i + j} X_{\beta_{v_\alpha}, j}(\beta_{v,0} x) \right) \times X_{\beta_{v_\delta}, i}\Big(x^{2^{d_v}} - x \Big).$$

Substituting in $\beta_{v,0} = \beta_{v_\alpha,0}$, (3.23) and (3.24), it follows that

$$\begin{aligned}
\sum_{i=0}^{\ell-1} h_i X_{\beta_v, i}(\beta_{v,0} x) &= \sum_{i=0}^{\lceil \ell/2^{d_v} \rceil - 1} \left(\sum_{j=0}^{\min(\ell - 2^{d_v} i, 2^{d_v}) - 1} g_{2^{d_v} i + j} x^j \right) X_{\beta_{v_\delta}, i}\Big(x^{2^{d_v}} - x \Big) \\
&= \sum_{j=0}^{\min(2^{d_v}, \ell) - 1} \left(\sum_{i=0}^{\lceil (\ell-j)/2^{d_v} \rceil - 1} g_{2^{d_v} i + j} X_{\beta_{v_\delta}, i}\left(\beta_{v_\delta, 0} \frac{x^{2^{d_v}} - x}{\beta_{v_\delta, 0}} \right) \right) x^j \\
&= \sum_{j=0}^{\min(2^{d_v}, \ell) - 1} \left(\sum_{i=0}^{\lceil (\ell-j)/2^{d_v} \rceil - 1} c_{2^{d_v} i + j} \left(\frac{x^{2^{d_v}} - x}{\beta_{v_\delta, 0}} \right)^i \right) x^j.
\end{aligned}$$

Hence, (3.25) holds. □

Using Lemma 3.25, we obtain Algorithms 7 and 8 for converting between the monomial basis and the twisted basis $\{X_{\beta_v,0}(\beta_{v,0}x), \dots, X_{\beta_v,\ell-1}(\beta_{v,0}x)\}$ of $\mathbb{F}[x]_\ell$. Each algorithm makes what is now a familiar pattern of recursive calls, but with the addition of the computation of either a generalised Taylor expansion or the inverse transformation, for which the algorithms of Section 3.5 are used.

Theorem 3.26. *Algorithms 7 and 8 are correct.*

Proof. We prove correctness for Algorithm 7 by induction on ℓ. The proof of correctness for Algorithm 8 is omitted since it is almost identical. For $v \in V$, we have $X_{\beta_v,0}(\beta_{v,0}x) = 1$ and $X_{\beta_v,1}(\beta_{v,0}x) = x$. Thus, Algorithm 7 produces the correct output for all inputs with $\ell \leq 2$. In particular, it follows that the algorithm produces the correct output whenever the input vertex is a leaf. Therefore, it is sufficient to show that for internal $v \in V$, if the algorithm produces the correct

Algorithm 7 X2M$(v, \ell, (a_0, \dots, a_{\ell-1}))$

Input: a vertex $v \in V$, $\ell \in \{1, \dots, 2^{n_v}\}$, and $a_i = h_i \in \mathbb{F}$ for $i \in \{0, \dots, \ell-1\}$.
Output: $a_i = f_i \in \mathbb{F}$ for $i \in \{0, \dots, \ell-1\}$ such that

$$\sum_{i=0}^{\ell-1} f_i x^i = \sum_{i=0}^{\ell-1} h_i X_{\beta_v, i}(\beta_{v,0} x). \tag{3.26}$$

1: **if** $\ell \leq 2$ **then return**
2: $\ell_1 \leftarrow \lceil \ell / 2^{d_v} \rceil - 1$, $\ell_2 \leftarrow \ell - 2^{d_v}\ell_1$, $\ell_2' \leftarrow \min(2^{d_v}, \ell)$
3: **for** $i = 0, \dots, \ell_1 - 1$ **do**
4: X2M$(v_\alpha, 2^{d_v}, (a_{2^{d_v} i}, a_{2^{d_v} i + 1}, \dots, a_{2^{d_v}(i+1)-1}))$
5: X2M$(v_\alpha, \ell_2, (a_{2^{d_v}\ell_1}, a_{2^{d_v}\ell_1 + 1}, \dots, a_{\ell-1}))$
6: **for** $j = 0, \dots, \ell_2 - 1$ **do**
7: X2M$(v_\delta, \ell_1 + 1, (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}\ell_1 + j}))$
8: **for** $j = \ell_2, \dots, \ell_2' - 1$ **do**
9: X2M$(v_\delta, \ell_1, (a_j, a_{2^{d_v}+j}, \dots, a_{2^{d_v}(\ell_1 - 1) + j}))$
10: **if** $\ell_1 \neq 0$ and $1/\beta_{v_\delta, 0} \neq 1$ **then**
11: $w \leftarrow 1/\beta_{v_\delta, 0}$
12: **for** $i = 1, \dots, \ell_1 - 1$ **do**
13: **for** $j = 0, \dots, 2^{d_v} - 1$ **do**
14: $a_{2^{d_v} i + j} \leftarrow w a_{2^{d_v} i + j}$
15: $w \leftarrow w / \beta_{v_\delta, 0}$
16: **for** $j = 0, \dots, \ell_2 - 1$ **do**
17: $a_{2^{d_v}\ell_1 + j} \leftarrow w a_{2^{d_v}\ell_1 + j}$
18: InverseTaylorExpansion$(2^{d_v}, \ell, (a_0, a_1, \dots, a_{\ell-1}))$

output whenever v_α or v_δ is given as an input, then it produces the correct output whenever v and $\ell \in \{3, \dots, 2^{n_v}\}$ are given as inputs.

Let $v \in V$ be an internal vertex and suppose that Algorithm 7 produces the correct output whenever v_α or v_δ is given as an input. Suppose that the algorithm is called on v and $\ell \in \{3, \dots, 2^{n_v}\}$, with $a_i = h_i \in \mathbb{F}$ for $i \in \{0, \dots, \ell-1\}$. Then the assumption that Algorithm 7 produces the correct output whenever v_α is given as an input implies that Lines 2–5 of the algorithm set $a_i = g_i$ for $i \in \{0, \dots, \ell-1\}$, where $g_0, \dots, g_{\ell-1}$ are the unique elements in $\mathbb{F}$ such that (3.23) holds. Similarly, the assumption implies that Lines 6–9 then set $a_i = c_i$ for $i \in \{0, \dots, \ell-1\}$, where $c_0, \dots, c_{\ell-1}$ are the unique elements in $\mathbb{F}$ such that (3.24) holds. As v is an internal vertex, Lemma 3.25 implies that $c_0, \dots, c_{\ell-1}$ also satisfy (3.25).

Let $f_0, \dots, f_{\ell-1}$ be the unique elements in $\mathbb{F}$ such that (3.26) holds. If $\ell \leq 2^{d_v}$, then (3.25) and (3.26) imply that $f_i = c_i$ for $i \in \{0, \dots, \ell-1\}$. Moreover, Lines 10–18 have no effect in this case. Therefore, the algorithm produces the correct output if $\ell \leq 2^{d_v}$. If $\ell > 2^{d_v}$, then Lines 10–17 set $a_{2^{d_v} i + j} = c_{2^{d_v} i + j} / \beta_{v_\delta, 0}^i$ for $i \in \{1, \dots, \lceil \ell / 2^{d_v} \rceil - 1\}$ and $j \in \{0, \dots, \min(\ell - 2^{d_v} i, 2^{d_v}) - 1\}$. Substituting into (3.25), it follows that

$$\sum_{i=0}^{\lceil \ell / 2^{d_v} \rceil - 1} \left(\sum_{j=0}^{\min(\ell - 2^{d_v} i, 2^{d_v}) - 1} a_{2^{d_v} i + j} x^j \right) \left(x^{2^{d_v}} - x \right)^i = \sum_{i=0}^{\ell-1} h_i X_{\beta_v, i}(\beta_{v,0} x)$$

Algorithm 8 M2X$(v, \ell, (a_0, \ldots, a_{\ell-1}))$

Input: a vertex $v \in V$, $\ell \in \{1, \ldots, 2^{n_v}\}$, and $a_i = f_i \in \mathbb{F}$ for $i \in \{0, \ldots, \ell-1\}$.
Output: $a_i = h_i \in \mathbb{F}$ for $i \in \{0, \ldots, \ell-1\}$ such that (3.26) holds.

1: **if** $\ell \leq 2$ **then return**
2: $\ell_1 \leftarrow \lceil \ell/2^{d_v} \rceil - 1$, $\ell_2 \leftarrow \ell - 2^{d_v}\ell_1$, $\ell_2' \leftarrow \min(2^{d_v}, \ell)$
3: TaylorExpansion$(2^{d_v}, \ell, (a_0, a_1, \ldots, a_{\ell-1}))$
4: **if** $\ell_1 \neq 0$ and $\beta_{v_\delta,0} \neq 1$ **then**
5: $\quad w \leftarrow \beta_{v_\delta,0}$
6: $\quad$ **for** $i = 1, \ldots, \ell_1 - 1$ **do**
7: $\quad\quad$ **for** $j = 0, \ldots, 2^{d_v} - 1$ **do**
8: $\quad\quad\quad a_{2^{d_v}i+j} \leftarrow w a_{2^{d_v}i+j}$
9: $\quad\quad w \leftarrow \beta_{v_\delta,0} w$
10: $\quad$ **for** $j = 0, \ldots, \ell_2 - 1$ **do**
11: $\quad\quad a_{2^{d_v}\ell_1+j} \leftarrow w a_{2^{d_v}\ell_1+j}$
12: **for** $j = 0, \ldots, \ell_2 - 1$ **do**
13: $\quad$ M2X$(v_\delta, \ell_1 + 1, (a_j, a_{2^{d_v}+j}, \ldots, a_{2^{d_v}\ell_1+j}))$
14: **for** $j = \ell_2, \ldots, \ell_2' - 1$ **do**
15: $\quad$ M2X$(v_\delta, \ell_1, (a_j, a_{2^{d_v}+j}, \ldots, a_{2^{d_v}(\ell_1-1)+j}))$
16: **for** $i = 0, \ldots, \ell_1 - 1$ **do**
17: $\quad$ M2X$(v_\alpha, 2^{d_v}, (a_{2^{d_v}i}, a_{2^{d_v}i+1}, \ldots, a_{2^{d_v}(i+1)-1}))$
18: M2X$(v_\alpha, \ell_2, (a_{2^{d_v}\ell_1}, a_{2^{d_v}\ell_1+1}, \ldots, a_{\ell-1}))$

when InverseTaylorExpansion is called in Line 18. Thus, the algorithm produces the correct output if $\ell > 2^{d_v}$. Hence, for internal $v \in V$, if the algorithm produces the correct output whenever v_α or v_δ is given as an input, then it produces the correct output whenever v and $\ell \in \{3, \ldots, 2^{n_v}\}$ are given as inputs. □

Algorithm 8 requires the precomputation and storage of the elements $\beta_{v_\delta,0}$, while their inverses are required for Algorithm 7. Consequently, the algorithms require auxiliary storage for $\mathcal{O}(n)$ field elements, while all precomputations can be performed with $\mathcal{O}(n^2)$ field operations. If $\ell_1 = \lceil \ell/2^{d_v} \rceil - 1$ is nonzero, then Lines 10–17 of Algorithm 7 and Lines 4–11 of Algorithm 8 perform

$$(\ell_1 - 1)\big(2^{d_v} + 1\big) + \ell_2 = \ell + \lceil \ell/2^{d_v} \rceil - 2^{d_v} - 2$$

multiplications, while the calls made by the algorithms to either TaylorExpansion or InverseTaylorExpansion perform at most $\lfloor \ell/2 \rfloor \lceil \log_2 \lceil \ell/2^{d_v} \rceil \rceil$ additions. It follows that we should once again aim to avoid small values of d_v when choosing a reduction tree for the algorithms. However, compared to the algorithms for conversion between the LCH and the Newton and Lagrange bases, a much greater cost in terms of multiplications and additions is incurred if one fails to do so. If β is a Cantor basis, then $\beta_{v_\delta,0} = 1$ for all internal $v \in V$, regardless of the choice of reduction tree (see Remark 3.1). Thus, Algorithms 7 and 8 perform no multiplications in this case, and require no precomputations.

Lin et al. [20] provide two algorithms for converting from the monomial basis to the LCH basis when ℓ is a power of two, one for arbitrary bases and one for Cantor bases. The reduction strategy they apply for the arbitrary bases corresponds to reduction trees with $\mathrm{Im}(d) \subseteq \{0, 1\}$. For such reduction trees, Algorithm 8 performs the same number of additions as their algorithm, but fewer multiplications in the

recursive case (after equalising precomputations). For Cantor bases, Algorithm 8 reduces to their algorithm by choosing the reduction tree so that $d_v = 2^{\lceil \log_2 n_v \rceil - 1}$ for all internal $v \in V$.

Theorem 3.27. *Algorithms 7 and 8 perform at most* $\lfloor \ell/2 \rfloor (3\lceil \log_2 \ell \rceil - 4) + 1$ *multiplications and* $\lfloor \ell/2 \rfloor \binom{\lceil \log_2 \ell \rceil}{2}$ *additions in* $\mathbb{F}$*. If* $d_v = 2^{\lceil \log_2 n_v \rceil - 1}$ *for all internal* $v \in V$*, then the algorithms perform at most* $\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil \lceil \log_2 \log_2 \max(\ell, 2) \rceil$ *additions in* $\mathbb{F}$*. If* β *is a Cantor basis, then the algorithms perform no multiplications.*

We have already shown that Algorithms 5 and 6 perform no multiplications when β is a Cantor basis. We split the remainder of the proof of Theorem 3.27 into three lemmas, one for each of three remaining bounds. It is clear that Algorithms 7 and 8 perform the same number of multiplications when given identical inputs. Consequently, we only prove the bounds for Algorithm 7. All three bounds are equal to zero or one for $\ell \leq 2$, while Algorithm 7 performs no additions or multiplications for such input values of ℓ. In particular, it follows that all three bound holds if the input vertex is a leaf. Consequently, for each of the three bounds it is sufficient to show that if $v \in V$ is an internal vertex such that the bound holds whenever the input vertex is v_α or v_δ, then the bound holds whenever the input vertex is v and $\ell \in \{3, \ldots, 2^{n_v}\}$.

Lemma 3.28. *Algorithms 7 and 8 perform at most* $\lfloor \ell/2 \rfloor (3\lceil \log_2 \ell \rceil - 4) + 1$ *multiplications in* $\mathbb{F}$*.*

Proof. Suppose that for some internal vertex $v \in V$, Algorithm 7 performs at most $\lfloor \ell/2 \rfloor (3\lceil \log_2 \ell \rceil - 4) + 1$ multiplications in $\mathbb{F}$ whenever v_α or v_δ is given as the input vertex. Furthermore, suppose that v and $\ell \in \{3, \ldots, 2^{n_v}\}$ are given as inputs to the algorithm. If $\ell_1 = 0$, then $\ell_2 = \ell_2' = \ell$ and the algorithm performs at most

$$\left\lfloor \frac{\ell_2}{2} \right\rfloor (3\lceil \log_2 \ell_2 \rceil - 4) + 1 + \ell_2 \times 0 = \left\lfloor \frac{\ell}{2} \right\rfloor (3\lceil \log_2 \ell \rceil - 4) + 1$$

multiplications. Therefore, suppose that $\ell_1 \neq 0$. Then, as $\ell_2 \leq 2^{d_v}$, Lines 3–5 of the algorithm perform at most

$$\ell_1 \big(2^{d_v - 1}(3d_v - 4) + 1\big) + \left\lfloor \frac{\ell_2}{2} \right\rfloor (3\lceil \log_2 \ell_2 \rceil - 4) + 1 \leq \left\lfloor \frac{\ell}{2} \right\rfloor (3d_v - 4) + \ell_1 + 1$$

multiplications. Lines 6–9 perform at most

$$\begin{aligned} \ell_2 \left\lfloor \frac{\ell + 1}{2} \right\rfloor (3\lceil \log_2 \ell_1 + 1 \rceil - 4) + \big(2^{d_v} - \ell_2\big) \left\lfloor \frac{\ell_1}{2} \right\rfloor (3\lceil \log_2 \ell_1 \rceil - 4) + 2^{d_v} \\ \leq \left\lfloor \frac{\ell_2(\ell + 1) + (2^{d_v} - \ell_2)\ell_1}{2} \right\rfloor (3\lceil \log_2 \ell_1 + 1 \rceil - 4) + 2^{d_v} \\ = \left\lfloor \frac{\ell}{2} \right\rfloor (3\lceil \log_2 \ell \rceil - 3d_v - 4) + 2^{d_v} \end{aligned}$$

multiplications, Lines 10–17 perform $(\ell_1 - 1)\big(2^{d_v} + 1\big) + \ell_2$ multiplications, and Line 18 performs no multiplications. Summing these bounds, it follows that Algorithm 7 performs at most

$$\begin{aligned}\left\lfloor \frac{\ell}{2} \right\rfloor (3\lceil \log_2 \ell \rceil - 4) + 1 - 4\left\lfloor \frac{\ell}{2} \right\rfloor + \big(2^{d_v} + 2\big)\ell_1 + \ell_2 - 1 \\ \leq \left\lfloor \frac{\ell}{2} \right\rfloor (3\lceil \log_2 \ell \rceil - 4) + 1 - 2\ell + \big(2^{d_v} + 2\big)\ell_1 + \ell_2 + 1 \\ = \left\lfloor \frac{\ell}{2} \right\rfloor (3\lceil \log_2 \ell \rceil - 4) + 1 - \big(2^{d_v} - 2\big)\ell_1 - (\ell_2 - 1) \\ \leq \left\lfloor \frac{\ell}{2} \right\rfloor (3\lceil \log_2 \ell \rceil - 4) + 1\end{aligned}$$

multiplications. □

Lemma 3.29. *Algorithms 7 and 8 perform at most* $\lfloor \ell/2 \rfloor \binom{\lceil \log_2 \ell \rceil}{2}$ *additions in* $\mathbb{F}$.

Proof. Suppose that for some internal vertex $v \in V$, Algorithm 7 performs at most $\lfloor \ell/2 \rfloor \binom{\lceil \log_2 \ell \rceil}{2}$ additions in $\mathbb{F}$ whenever v_α or v_δ is given as the input vertex. Furthermore, suppose that v and $\ell \in \{3, \ldots, 2^{n_v}\}$ are given as inputs to the algorithm. If $\ell_1 = 0$, then $\ell_2 = \ell_2' = \ell$ and the algorithm performs at most

$$\left\lfloor \frac{\ell_2}{2} \right\rfloor \binom{\lceil \log_2 \ell_2 \rceil}{2} + \ell_2 \times 0 + \left\lfloor \frac{\ell}{2} \right\rfloor \lceil \log_2 1 \rceil = \left\lfloor \frac{\ell}{2} \right\rfloor \binom{\lceil \log_2 \ell \rceil}{2}$$

additions. Therefore, suppose that $\ell_1 > 0$. Then, as $\ell_2 \leq 2^{d_v}$, Lines 3–5 of the algorithm perform at most

$$\ell_1 2^{d_v - 1} \binom{d_v}{2} + \left\lfloor \frac{\ell_2}{2} \right\rfloor \binom{\lceil \log_2 \ell_2 \rceil}{2} \leq \ell_1 2^{d_v - 1} \binom{d_v}{2} + \left\lfloor \frac{\ell_2}{2} \right\rfloor \binom{d_v}{2} = \left\lfloor \frac{\ell}{2} \right\rfloor \binom{d_v}{2}$$

additions. Lines 6–9 of the algorithm perform at most

$$\ell_2 \left\lfloor \frac{\ell + 1}{2} \right\rfloor \binom{\lceil \log_2 \ell_1 + 1 \rceil}{2} + \big(2^{d_v} - \ell_2\big) \left\lfloor \frac{\ell_1}{2} \right\rfloor \binom{\lceil \log_2 \ell_1 \rceil}{2} \leq \left\lfloor \frac{\ell}{2} \right\rfloor \binom{\lceil \log_2 \ell_1 + 1 \rceil}{2}$$

additions, since $\ell_2(\ell + 1) + (2^{d_v} - \ell_2)\ell_1 = \ell$. Lines 10–17 perform no additions, while Lemma 3.24 implies that Line 18 performs at most $\lfloor \ell/2 \rfloor \lceil \log_2 \lceil \ell/2^{d_v} \rceil \rceil = \lfloor \ell/2 \rfloor \lceil \log_2 \ell_1 + 1 \rceil$ additions. As $\lceil \log_2 \ell_1 + 1 \rceil = \lceil \log_2 \ell \rceil - d_v$, it follows by summing these bounds that Algorithm 7 performs at most

$$\left\lfloor \frac{\ell}{2} \right\rfloor \left(\binom{\lceil \log_2 \ell \rceil}{2} - \lceil \log_2 \ell_1 + 1 \rceil (d_v - 1) \right) \leq \left\lfloor \frac{\ell}{2} \right\rfloor \binom{\lceil \log_2 \ell \rceil}{2}$$

additions. □

Lemma 3.30. *Suppose that* $d_v = 2^{\lceil \log_2 n_v \rceil - 1}$ *for all internal* $v \in V$. *Then Algorithms 7 and 8 perform at most* $\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil \lceil \log_2 \log_2 \max(\ell, 2) \rceil$ *additions in* $\mathbb{F}$.

Proof. Suppose that $d_v = 2^{\lceil \log_2 n_v \rceil - 1}$ for all internal vertices $v \in V$. Furthermore, suppose that for some internal vertex $v \in V$, Algorithm 7 performs at most $\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil \lceil \log_2 \log_2 \max(\ell, 2) \rceil$ additions in $\mathbb{F}$ whenever v_α or v_δ is given as the input vertex. Finally, suppose that v and $\ell \in \{3, \ldots, 2^{n_v}\}$ are given as inputs to the algorithm. If $\ell_1 = 0$, then $\ell_2 = \ell_2' = \ell$ and the algorithm performs at most

$$\left\lfloor \frac{\ell_2}{2} \right\rfloor \lceil \log_2 \ell_2 \rceil \lceil \log_2 \log_2 \ell_2 \rceil + \ell_2 \times 0 + \left\lfloor \frac{\ell}{2} \right\rfloor \lceil \log_2 1 \rceil = \left\lfloor \frac{\ell}{2} \right\rfloor \lceil \log_2 \ell \rceil \lceil \log_2 \log_2 \ell \rceil$$

additions. Therefore, suppose that $\ell_1 > 0$. Then, as $\ell_2 \leq 2^{d_v} < \ell$, Lines 3–5 of the algorithm perform at most

$$\ell_1 2^{d_v - 1} d_v \lceil \log_2 \log_2 \ell \rceil + \left\lfloor \frac{\ell_2}{2} \right\rfloor d_v \lceil \log_2 \log_2 \ell \rceil = \left\lfloor \frac{\ell}{2} \right\rfloor d_v \lceil \log_2 \log_2 \ell \rceil \tag{3.27}$$

additions. Lines 6–7 of the algorithm perform at most

$$\ell_2 \left\lfloor \frac{\ell_1 + 1}{2} \right\rfloor \lceil \log_2 \ell_1 + 1 \rceil \lceil \log_2 \log_2 \ell_1 + 1 \rceil$$

additions, while Lines 8–9 perform at most

$$\left(2^{d_v} - \ell_2\right) \left\lfloor \frac{\ell_1}{2} \right\rfloor \lceil \log_2 \ell_1 \rceil \lceil \log_2 \log_2 \max(\ell_1, 2) \rceil$$

additions. As $\ell_2(\ell_1 + 1) + (2^{d_v} - \ell_2)\ell_1 = \ell$, it follows that Lines 6–9 of the algorithm perform at most $\lfloor \ell/2 \rfloor \lceil \log_2 \ell_1 + 1 \rceil \lceil \log_2 \log_2 \ell_1 + 1 \rceil$ additions. If $\ell_1 \geq 2$, then there exists an integer $k \geq 1$ such that $2^{2^{k-1}} < \ell_1 + 1 \leq 2^{2^k}$. Then $\lceil \log_2 \log_2 \ell_1 + 1 \rceil = k$, $2^{2^{k-1}} < 2^{n_v - d_v} \leq 2^{d_v}$ and $\ell = 2^{d_v}(\ell_1 + \ell_2/2^{d_v}) > 2^{d_v + 2^{k-1}} > 2^{2^k}$. Thus, $\lceil \log_2 \log_2 \ell_1 + 1 \rceil \leq \lceil \log_2 \log_2 \ell \rceil - 1$ if $\ell_1 \geq 2$. As $\ell \geq 3$, the inequality also holds if $\ell_1 = 1$. Therefore, Lines 6–9 of the algorithm perform at most

$$\left\lfloor \frac{\ell}{2} \right\rfloor (\lceil \log_2 \ell \rceil - d_v) \lceil \log_2 \log_2 \ell \rceil - \left\lfloor \frac{\ell}{2} \right\rfloor \lceil \log_2 \ell_1 + 1 \rceil \tag{3.28}$$

additions. Lines 10–17 of the algorithm perform no additions, while Lemma 3.24 implies that Line 18 performs at most $\lfloor \ell/2 \rfloor \lceil \log_2 \lceil \ell / 2^{d_v} \rceil \rceil = \lfloor \ell/2 \rfloor \lceil \log_2 \ell_1 + 1 \rceil$ additions. By combining this last bound with the bounds (3.27) and (3.28) on the number of additions performed by Lines 3–5 and Lines 6–9, it follows that Algorithm 7 performs at most $\lfloor \ell/2 \rfloor \lceil \log_2 \ell \rceil \lceil \log_2 \log_2 \ell \rceil$ additions, which is the required bound since $\ell \geq 3$. □

4. Constructing a basis and reduction tree

Let $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{F}^n$ have entries that are linearly independent over $\mathbb{F}_2$. If $n = 1$, then there exists a unique reduction tree for β, the tree consisting of a single vertex. If $n > 1$, then a full binary tree is a reduction tree for β if it has n leaves, the subtrees rooted on the children r_α and r_δ of the tree's root vertex r are themselves reduction trees for $\alpha(\beta, d(r))$ and $\delta(\beta, d(r))$, respectively, and the quotients $\beta_0/\beta_0, \ldots, \beta_{d(r)-1}/\beta_0$ belong to $\mathbb{F}_{2^{d(r)}}$. The requirement on the quotients is trivially satisfied if $d(r) = 1$. Consequently, the full binary tree with n leaves and $\mathrm{Im}(d) \subseteq \{0, 1\}$ is a reduction tree for β (Proposition 2.7). We view this tree as the trivial choice of reduction tree for the basis, and as capturing the approach used by existing algorithms. We expect such trees to yield the worst algebraic complexity for the algorithms of Section 3. Accordingly, when we have freedom to choose the basis vector, our choice should enable us to avoid their use. Cantor bases provide such a choice, and we witnessed the benefits they provide in Section 3. However, Cantor bases are restricted to extensions with degree divisible by sufficiently large powers of two. In this section, we propose new basis constructions that allow us to benefit similarly in other extensions.

Regardless of the chosen basis vector, the choice of reduction trees is limited by the subfield structure of $\mathbb{F}$.

Proposition 4.1. *If (V, E) is a reduction tree for some vector in $\mathbb{F}^n$, then*

$$d(v_\alpha) < d(v) < n \leq [\mathbb{F} : \mathbb{F}_2] \quad \textit{and} \quad \max(d(v_\alpha), 1) \mid d(v) \mid [\mathbb{F} : \mathbb{F}_2]$$

for all internal $v \in V$.

Proof. Suppose that (V, E) is a reduction tree for some vector $\beta \in \mathbb{F}^n$. Then $n \leq [\mathbb{F} : \mathbb{F}_2]$ since Definition 2.6 requires β to have linearly independent entries over $\mathbb{F}_2$. The definition also requires the tree to have n leaves. Thus,

$$d(v_\alpha) < |L_{v_\alpha}| = d(v) < |L_v| \leq n \leq [\mathbb{F} : \mathbb{F}_2]$$

for all internal $v \in V$.

Let $v \in V$ be an internal vertex. Then it follows from Definition 2.6 that the subtree rooted on v is a reduction tree for some vector $\beta_v = (\beta_{v,0}, \ldots, \beta_{v,|L_v|-1}) \in \mathbb{F}^{|L_v|}$ that has linearly independent entries over $\mathbb{F}_2$. Moreover, as $|L_v| > 1$, the definition implies that the quotients $\beta_{v,0}/\beta_{v,0}, \ldots, \beta_{v,d(v)-1}/\beta_{v,0}$ belong to $\mathbb{F}_{2^{d(v)}}$. These quotients inherit linear independence over $\mathbb{F}_2$. Thus, they form a basis of the extension $\mathbb{F}_{2^{d(v)}}/\mathbb{F}_2$. As the quotients also belong to $\mathbb{F}$, it follows that $\mathbb{F}_{2^{d(v)}}$ is a subfield of $\mathbb{F}$. Therefore, $d(v)$ divides $[\mathbb{F} : \mathbb{F}_2]$. Similarly, if the vertex is v_α is an internal vertex, then $\mathbb{F}_{2^{d(v_\alpha)}}$ is a subfield of $\mathbb{F}_{2^{d(v)}}$, since the subtree rooted on v_α is a reduction tree for $\alpha(\beta_v, d(v)) = (\beta_{v,0}, \ldots, \beta_{v,d(v)-1})$. As v_α is an internal vertex if and only if $d(v_\alpha) \geq 1$, it follows that $\max(d(v_\alpha), 1)$ divides $d(v)$. □

Corollary 4.2. *Suppose that the entries of $\beta \in \mathbb{F}^n$ are linearly independent over $\mathbb{F}_2$, and $[\mathbb{F} : \mathbb{F}_2]$ has no proper factor less than n. Then a full binary tree is a reduction tree for β if and only if it has n leaves and $\mathrm{Im}(d) \subseteq \{0, 1\}$.*

Proof. Proposition 2.7 implies that it is sufficient to have n leaves and $\mathrm{Im}(d) \subseteq \{0, 1\}$, while Proposition 4.1 implies that it is also necessary. □

4.1. **A construction for arbitrary fields.** It follows from Proposition 4.1 that a path in a reduction tree that consists of two or more edges of the form $\{v, v_\alpha\}$ admits a nontrivial tower of subfields of $\mathbb{F}$. However, the existence of a basis vector of a prescribed dimension that has a nontrivial reduction tree is not guaranteed by the existence of nontrivial tower of subfields. Indeed, Corollary 4.2 shows that it is necessary for the tower to contain a subfield other than $\mathbb{F}_2$ of degree bounded by the dimension. In this section, we show that this requirement is also sufficient.

Theorem 4.3. *Suppose there exists a tower of subfields*

$$\mathbb{F}_2 = \mathbb{F}_{2^{n_0}} \subset \mathbb{F}_{2^{n_1}} \subset \cdots \subset \mathbb{F}_{2^{n_m}} = \mathbb{F}. \tag{4.1}$$

Let $\{\vartheta_{k,0}, \ldots, \vartheta_{k,n_{k+1}/n_k-1}\}$ be a basis of $\mathbb{F}_{2^{n_{k+1}}}/\mathbb{F}_{2^{n_k}}$ for $k \in \{0, \ldots, m-1\}$, and

$$\beta_i = \prod_{k=0}^{m-1} \vartheta_{k,i_k} \quad \textit{such that} \quad \sum_{k=0}^{m-1} i_k n_k = i$$

for $i \in \{0, \ldots, n_m - 1\}$. Then $\beta_0, \ldots, \beta_{n_m-1} \in \mathbb{F}$ are linearly independent over $\mathbb{F}_2$. Moreover, a full binary tree (V, E) with $n \leq n_m$ leaves is a reduction tree for $(\beta_0, \ldots, \beta_{n-1})$ if $\mathrm{Im}(d) \subseteq \{0, n_0, \ldots, n_{m-1}\}$ and $d(v_\delta) \leq d(v)$ for all internal $v \in V$.

The requirements of Theorem 4.3 are satisfied by the full binary tree with n leaves and $\mathrm{Im}(d) \subseteq \{0, 1\}$. Consequently, Proposition 2.7 follows from the case $m = 1$. We delay the proof of the theorem until the end of the section. Instead, we

now show that the basis vectors given by the construction of the theorem allow a nontrivial and, more importantly, beneficial choice of reduction trees.

Proposition 4.4. *If $I \subseteq \mathbb{N}$ and (V, E) is a full binary tree such that*

$$d(v) = \max\{i \in I \mid i < |L_v|\}$$

for all internal $v \in V$, then $d(v_\delta) \leq d(v)$ for all internal $v \in V$.

Proof. Suppose that $I \subseteq \mathbb{N}$ and a full binary tree (V, E) satisfy the conditions of the proposition. Then $d(v_\delta) < |L_{v_\delta}| < |L_v|$ and $d(v_\delta) \in I \cup \{0\}$ for all internal $v \in V$. Hence, $d(v_\delta) \leq \max\{i \in I \mid i < |L_v|\} = d(v)$ for all internal $v \in V$. □

For tuples of positive integers $(n_0, \ldots, n_m)$ such that (4.1) holds, let $T_n^{(n_0,\ldots,n_m)}$ denote the full binary tree with n leaves and $d(v) = \max\{n_k \mid n_k < |L_v|\}$ for all internal vertices v. If $n_1 < n \leq n_m$, then the root vertex r of $T_n^{(n_0,\ldots,n_m)}$ satisfies $d(r) \geq n_1 > 1$, establishing the existence of a tree with $\mathrm{Im}(d) \not\subseteq \{0, 1\}$ that satisfies the conditions of Theorem 4.3. Moreover, we expect this tree to approximately minimise the algebraic complexity of the conversion algorithms of Section 3 over all trees that satisfy the conditions of the theorem.

Example 4.5. Suppose that $\mathbb{F} = \mathbb{F}_{2^{12}}$. Then there are eight tuples of positive integers $(n_0, \ldots, n_m)$ such that (4.1) holds. For each such tuple, Figure 6 displays the relative number of additions performed by the basis conversion algorithms of Section 3 for $\beta = (\beta_0, \ldots, \beta_{11})$ given by the constructed of Theorem 4.3 (the choice of the bases for the extensions $\mathbb{F}_{2^{n_{k+1}}}/\mathbb{F}_{2^{n_k}}$ doesn't matter here), the reduction tree $T_{12}^{(n_0,\ldots,n_m)}$, and polynomial length (ℓ) ranging over $\{1, \ldots, 2^{12}\}$. The number of additions performed in each case is given as a fraction of the number performed for the tuple $(1, 12)$, which is taken to be one if zero additions are performed for both tuples. The reduction tree that corresponds to the tuple $(1, 12)$ has $\mathrm{Im}(d) = \{0, 1\}$. Thus, it represents the complexity obtained with the reduction strategy of existing algorithms. The additional parameters $c = \ell$ and $b = 0$ are used for Algorithm 3, and $c = \ell$ is used for Algorithm 4. Figure 7 similarly displays the relative number of multiplications performed by Algorithms 7 and 8, under the assumption that $\beta_{v_\delta,0}$ is never equal to one in Line 10 of Algorithm 7 and Line 4 of Algorithm 8. The daggered tuples that appear in the figure are discussed in the next section.

Example 4.5 demonstrates that the construction of Theorem 4.3 and the choice of reduction trees it provides allows us to achieve a lower algebraic complexity if $[\mathbb{F} : \mathbb{F}_2]$ has even a single sufficiently small factor. The potential benefits are greater still when the degree of the field contains many small prime factors, echoing the benefits obtained by using roots of unity with smooth order in multiplicative FFTs. While a reduction in algebraic complexity is certainly desirable, it is not the only consideration in practice. For example, Harvey's "cache-friendly" variant [15] of the radix-2 truncated Fourier transform [25] obtains better practical performance by optimising cache effects. This variant employs a reduction strategy that more rapidly reduces to problems of size that fit into cache, helping to reduce data exchanges with RAM. An analogous approach for the algorithms of Section 3 is to use reduction trees that balance the size of $|L_{v_\alpha}|$ and $|L_{v_\delta}|$ for internal vertices. If $[\mathbb{F} : \mathbb{F}_2]$ is smooth and the construction of Theorem 4.3 is applied with the number of subfields in the tower taken as large as possible, then the following proposition shows that it is possible to construct such trees while meeting requirements of

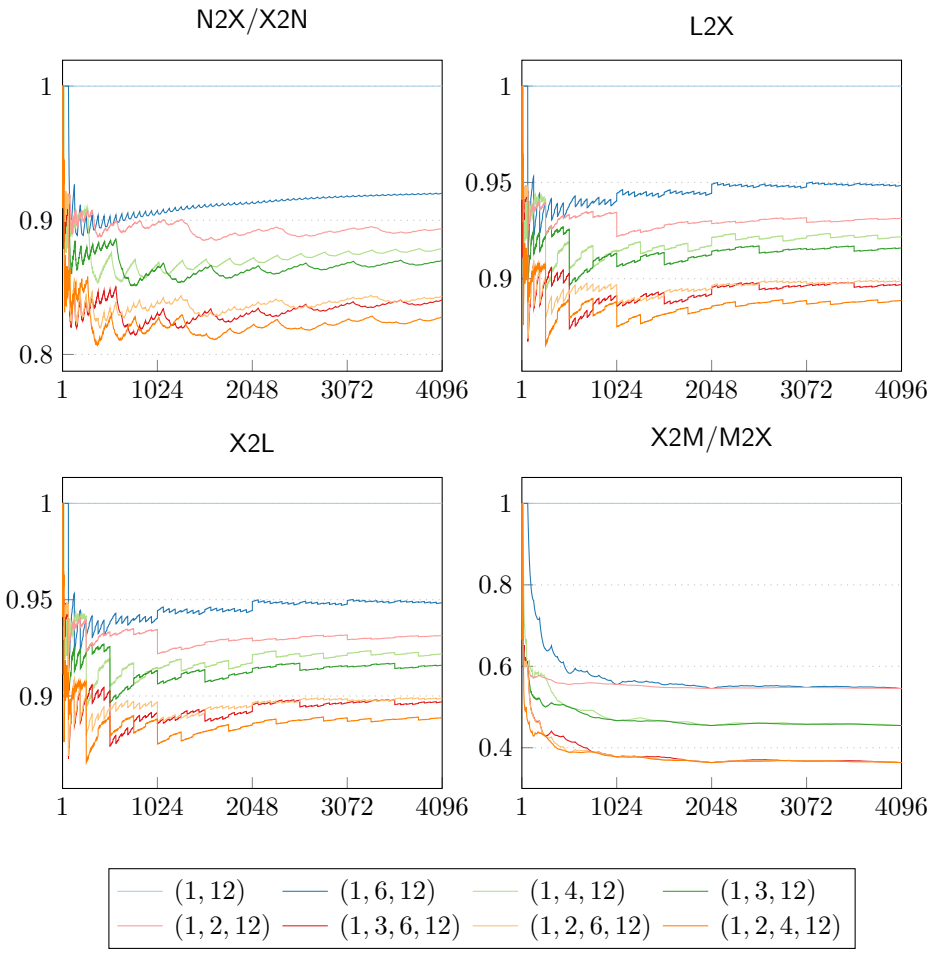

FIGURE 6. Relative number of additions performed by the algorithms of Section 3 (see Example 4.5).

the theorem by choosing $d(v) \in \{n_0, \ldots, n_m\}$ to minimise $\max(n_k, |L_v| - n_k)$ for all internal vertices. While there are trade-offs between optimising cache effects and reducing the operation count to be considered in practice, it appears that Theorem 4.3 offers us some freedom, especially when the degree of the field is smooth, to tune the algorithms of Section 3.

Proposition 4.6. *If $I \subseteq \mathbb{N}$ and (V, E) is a full binary tree such that*

$$d(v) \in \arg\min_{i \in I} \max(i, |L_v| - i)$$

for all internal $v \in V$, then $d(v_\delta) \leq d(v)$ for all internal $v \in V$.

Proof. We prove the proposition by contradiction. Suppose that $I \subseteq \mathbb{N}$ and a full binary tree (V, E) satisfy the conditions of the proposition. Furthermore, suppose there exists an internal vertex $v \in V$ such that $d(v_\delta) > d(v)$. Then v_δ is an internal

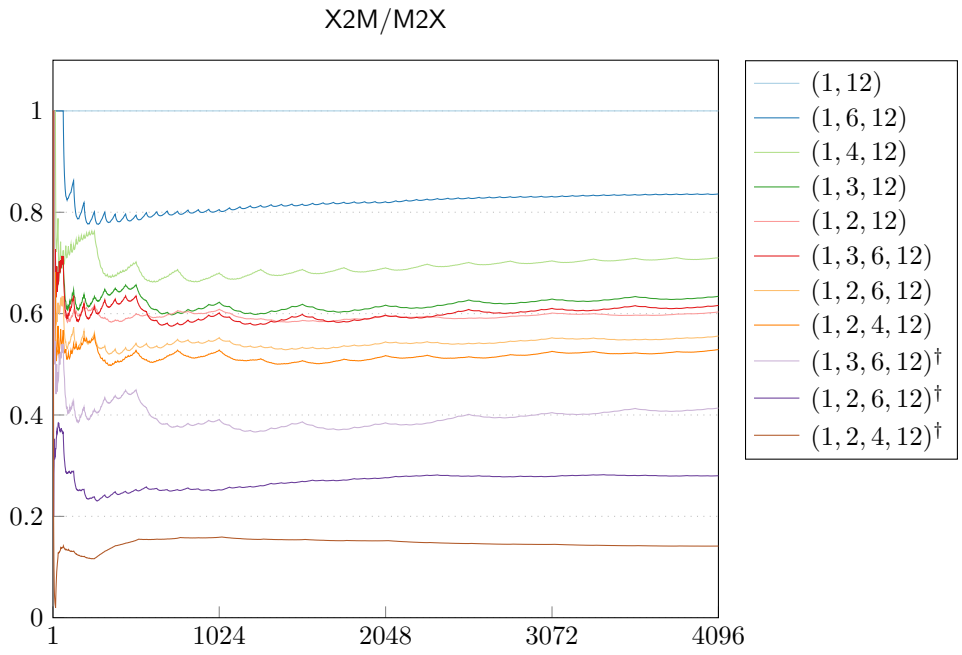

FIGURE 7. Relative number of multiplications performed by Algorithms 7 and 8 (see Example 4.5).

vertex since $d(v_\delta) > 1$. Thus, $d(v_\delta) \in I$ and

$$\begin{aligned}\max(d(v_\delta), |L_v| - d(v_\delta)) &< \max(|L_{v_\delta}|, |L_v| - d(v)) \\ &= \max(|L_v| - |L_{v_\alpha}|, |L_v| - d(v)) \\ &= \max(|L_v| - d(v), |L_v| - d(v)) \\ &\leq \max(d(v), |L_v| - d(v)),\end{aligned}$$

which contradicts the minimality of $\max(d(v), |L_v| - d(v))$. □

We now turn our attention to the proof of Theorem 4.3, which we obtain as a consequence of the following more general result.

Lemma 4.7. *Suppose there exists a tower of subfields $\mathbb{F}_2 = \mathbb{F}_{2^{n_0}} \subset \cdots \subset \mathbb{F}_{2^{n_m}} = \mathbb{F}$. Let $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{F}^n$ have entries that are linearly independent over $\mathbb{F}_2$, and (V, E) be a full binary tree with n leaves and root vertex $r \in V$. Then (V, E) is a reduction tree for β if the following conditions are satisfied:*

1. $\mathrm{Im}(d) \subseteq \{0, n_0, \ldots, n_{m-1}\}$,
2. $d(v_\delta) \leq d(v)$ *for all internal* $v \in V$, *and*
3. $\beta_i / \beta_{n_k \lfloor i/n_k \rfloor} \in \mathbb{F}_{2^{n_k}}$ *for* $i \in \{0, \ldots, n-1\}$ *and* $k \in \{0, \ldots, m-1\}$ *such that* $n_k \leq d(r)$.

Proof. We prove the lemma by induction on n. The lemma holds trivially if $n = 1$, since it is sufficient for (V, E) to have n leaves in this case. Therefore, let $n \geq 2$ and suppose that the lemma is true for all smaller values of n. Suppose that $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{F}^n$ and a full binary tree (V, E) satisfy the conditions of the

lemma. Then the root vertex $r \in V$ of the tree is not a leaf since $|L_r| = n \geq 2$. Thus, (1) implies that $d(r) = n_\ell$ for some $\ell \in \{0, \ldots, m-1\}$ such that $n_\ell < n$.

Let $\alpha(\beta, d(r)) = (\alpha_0, \ldots, \alpha_{n_\ell - 1})$ and $\delta(\beta, d(r)) = (\delta_0, \ldots, \delta_{n-n_\ell-1})$, which have linearly independent entries over $\mathbb{F}_2$ by Lemma 2.1. Then, as $d(r_\alpha) < d(r)$, (3) implies that $\alpha_i/\alpha_{n_k\lfloor i/n_k \rfloor} = \beta_i/\beta_{n_k\lfloor i/n_k \rfloor} \in \mathbb{F}_{2^{n_k}}$ for $i \in \{0, \ldots, n_\ell - 1\}$ and $k \in \{0, \ldots, m-1\}$ such that $n_k \leq d(r_\alpha)$. Moreover, as $n_0, \ldots, n_\ell$ divide n_ℓ, we have

$$\frac{\beta_{n_\ell+i}}{\beta_0} = \frac{\beta_{n_\ell+i}}{\beta_{n_\ell+n_k\lfloor i/n_k \rfloor}} \frac{\beta_{n_\ell+n_k\lfloor i/n_k \rfloor}}{\beta_0} = \frac{\beta_{n_\ell+i}}{\beta_{n_k\lfloor (n_\ell+i)/n_k \rfloor}} \frac{\beta_{n_\ell+n_k\lfloor i/n_k \rfloor}}{\beta_0},$$

where $\beta_{n_\ell+i}/\beta_{n_k\lfloor (n_\ell+i)/n_k \rfloor} \in \mathbb{F}_{2^{n_k}} \subseteq \mathbb{F}_{2^{n_\ell}}$, for $i \in \{0, \ldots, n-n_\ell-1\}$ and $k \in \{0, \ldots, \ell\}$. It follows that

$$\begin{aligned}\delta_i &= \frac{\beta_{n_\ell+i}}{\beta_{n_k\lfloor (n_\ell+i)/n_k \rfloor}} \left(\left(\frac{\beta_{n_\ell+n_k\lfloor i/n_k \rfloor}}{\beta_0} \right)^{2^{n_\ell}} - \frac{\beta_{n_\ell+n_k\lfloor i/n_k \rfloor}}{\beta_0} \right) \\ &= \frac{\beta_{n_\ell+i}}{\beta_{n_k\lfloor (n_\ell+i)/n_k \rfloor}} \delta_{n_k\lfloor i/n_k \rfloor}\end{aligned}$$

for $i \in \{0, \ldots, n-n_\ell-1\}$ and $k \in \{0, \ldots, \ell\}$. As $d(r_\delta) \leq d(r) = n_\ell$ by (2), it follows that $\delta_i/\delta_{n_k\lfloor i/n_k \rfloor} \in \mathbb{F}_{2^{n_k}}$ for $i \in \{0, \ldots, n-n_\ell-1\}$ and $k \in \{0, \ldots, m-1\}$ such that $n_k \leq d(r_\delta)$. Conditions (1) and (2) are satisfied by the subtrees of (V, E) rooted on r_α and r_δ through inheritance. The subtree rooted on r_α has $|L_{r_\alpha}| = d(r) = n_\ell < n$ leaves, while the subtree rooted on r_δ has $|L_{r_\delta}| = |L_r| - |L_{r_\alpha}| = n - d(r) = n - n_\ell < n$ leaves. Therefore, the induction hypothesis implies that the subtree rooted on r_α is a reduction tree for $\alpha(\beta, d(r))$, and the subtree rooted on r_δ is a reduction tree for $\delta(\beta, d(r))$. Finally, (3) implies that $\beta_i/\beta_0 = \beta_i/\beta_{n_\ell\lfloor i/n_\ell \rfloor} \in \mathbb{F}_{2^{d(r)}}$ for $i \in \{0, \ldots, d(r)-1\}$. Hence, (V, E) is a reduction tree for β. □

If $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{F}^n$ is a Cantor basis with $n \geq 4$, then properties (1) and (2) of Lemma 2.9 imply that $\beta_3 \in \mathbb{F}_{16} \setminus \mathbb{F}_4$. Thus, β_3/β_2 does not belong to $\mathbb{F}_4$, since $\beta_3 = \beta_2/\beta_3 + 1$. Consequently, Proposition 2.8 implies that the converse of Lemma 4.7 does not hold. We now complete the proof of Theorem 4.3 by establishing linear independence and showing that the vectors $(\beta_0, \ldots, \beta_{n-1})$ for $n \in \{1, \ldots, n_m\}$ always satisfy the third condition of Lemma 4.7.

Lemma 4.8. *Suppose that $\beta_0, \ldots, \beta_{n_m-1}$ are given by the construction of Theorem 4.3. Then $\beta_0, \ldots, \beta_{n_m-1}$ are linearly independent over $\mathbb{F}_2$, and $\beta_i/\beta_{n_k\lfloor i/n_k \rfloor} \in \mathbb{F}_{2^{n_k}}$ for $i \in \{0, \ldots, n_m - 1\}$ and $k \in \{0, \ldots, m-1\}$.*

Proof. Suppose that $\beta_0, \ldots, \beta_{n_m-1}$ are given by the construction of Theorem 4.3. Let $i \in \{0, \ldots, n_m - 1\}$ and $i = \sum_{k=0}^{m-1} i_k n_k$ with $i_k \in \{0, \ldots, n_{k+1}/n_k - 1\}$ for $k \in \{0, \ldots, m-1\}$. Then

$$\frac{\beta_i}{\beta_{n_k\lfloor i/n_k \rfloor}} = \frac{\vartheta_{0,i_0} \cdots \vartheta_{k-1,i_{k-1}} \vartheta_{k,i_k} \cdots \vartheta_{m-1,i_{m-1}}}{\vartheta_{0,0} \cdots \vartheta_{k-1,0} \vartheta_{k,i_k} \cdots \vartheta_{m-1,i_{m-1}}} = \frac{\vartheta_{0,i_0} \cdots \vartheta_{k-1,i_{k-1}}}{\vartheta_{0,0} \cdots \vartheta_{k-1,0}} \in \mathbb{F}_{2^{n_k}}$$

for $k \in \{0, \ldots, m\}$, as required. Similarly,

$$\frac{\beta_{in_k+j}}{\beta_0} = \frac{\vartheta_{k,i}}{\vartheta_{k,0}} \frac{\beta_j}{\beta_0} \quad \text{and} \quad \frac{\beta_j}{\beta_0} = \frac{\beta_j}{\beta_{n_k\lfloor j/n_k \rfloor}} \in \mathbb{F}_{2^{n_k}} \tag{4.2}$$

for $i \in \{0, \ldots, n_{k+1}/n_k - 1\}$, $j \in \{0, \ldots, n_k - 1\}$ and $k \in \{0, \ldots, m-1\}$. Now $\{\vartheta_{k,0}/\vartheta_{k,0}, \ldots, \vartheta_{k,n_{k+1}/n_k-1}/\vartheta_{k,0}\}$ is a basis of $\mathbb{F}_{2^{n_{k+1}}}/\mathbb{F}_{2^{n_k}}$ for $k \in \{0, \ldots, m-1\}$. Therefore, if $\beta_0/\beta_0, \ldots, \beta_{n_k-1}/\beta_0$ are linearly independent over $\mathbb{F}_2$ for some $k \in$

$\{0, \ldots, m-1\}$, then (4.2) implies that $\beta_0/\beta_0, \ldots, \beta_{n_{k+1}-1}/\beta_0$ are linearly independent over $\mathbb{F}_2$. As $n_0 = 1$, it follows that $\beta_0, \ldots, \beta_{n_m-1}$ are linearly independent over $\mathbb{F}_2$. □

4.2. **Fewer multiplications for quadratic extensions.** Theorem 4.3 does not place any restrictions on the choice of bases for the extensions $\mathbb{F}_{2^{n_{k+1}}}/\mathbb{F}_{2^{n_k}}$. In this section, we show that if some of these extension are quadratic, then it possible to choose their bases so that Algorithms 7 and 8 perform fewer multiplications.

Theorem 4.9. *Let $\beta_0, \ldots, \beta_{n_m-1} \in \mathbb{F}$ be constructed as per Theorem 4.3, $n \in \{1, \ldots, n_m\}$, and V be the vertex set of the tree $T_n^{(n_0,\ldots,n_m)}$. Recursively define vectors $\beta_v = (\beta_{v,0}, \ldots, \beta_{v,|L_v|-1})$ for $v \in V$ as follows: if v is the root of the tree, then $\beta_v = (\beta_0, \ldots, \beta_{n-1})$; and if v is an internal vertex, then $\beta_{v_\alpha} = \alpha(\beta_v, d(v))$ and $\beta_{v_\delta} = \delta(\beta_v, d(v))$. Suppose there exists $t \in \{0, \ldots, m-1\}$ such that*

$$\frac{n_{t+1}}{n_t} = 2 \quad \text{and} \quad \operatorname{Tr}_{\mathbb{F}_{2^{n_{t+1}}}/\mathbb{F}_{2^{n_t}}}\left(\frac{\vartheta_{t,1}}{\vartheta_{t,0}}\right) = 1. \tag{4.3}$$

Then $\beta_{v_\delta,0} = 1$ for all $v \in V$ such that $d(v) = n_t$.

The definition of the vectors β_v in Theorem 4.9 matches that of Section 3. Consequently, for $\beta = (\beta_0, \ldots, \beta_{n-1})$ given by the construction of Theorem 4.3, and the reduction tree $T_n^{(n_0,\ldots,n_m)}$, Lines 10–17 of Algorithm 7 (similarly, Lines 4–11 of Algorithm 8) perform no multiplications if the input vertex v satisfies $d(v) = n_t$ for some t such that (4.3) holds. If $n_{t+1}/n_t = 2$, then we may ensure that the condition on the trace in (4.3) is satisfied by taking $\vartheta_{t,0} = 1$ and $\vartheta_{t,1} \in \mathbb{F}_{2^{n_{t+1}}}$ with $\operatorname{Tr}_{\mathbb{F}_{2^{n_{t+1}}}/\mathbb{F}_{2^{n_t}}}(\vartheta_{t,1}) = 1$, which yields a basis since the trace of one is equal to zero. Therefore, it is possible to achieve a significant reduction in the number of multiplications performed by Algorithms 7 and 8 when the tower used in the construction contains several quadratic extensions. Returning to Example 4.5, we see such an improvement for $\mathbb{F} = \mathbb{F}_{2^{12}}$ by looking at the relative number of multiplications performed for the daggered tuples in Figure 7. For these tuples, it is assumed that $\beta_{v_\delta,0} = 1$ if and only if $d(v) = n_t$ for some t such that $n_{t+1}/n_t = 2$.

We now turn our attention to the proof of Theorem 4.9.

Lemma 4.10. *Assume the hypothesis and notation of Theorem 4.9. If $v \in V$ is an internal vertex and $d(v) = n_\ell$, then there exist elements $\vartheta'_{\ell,0}, \vartheta'_{\ell,1}, \ldots \in \mathbb{F}_{2^{n_{\ell+1}}}$ that are linearly independent over $\mathbb{F}_{2^{n_\ell}}$, for which*

$$\beta_{v_\delta,i} = \frac{\vartheta_{0,i_0}}{\vartheta_{0,0}} \cdots \frac{\vartheta_{\ell-1,i_{\ell-1}}}{\vartheta_{\ell-1,0}} \vartheta'_{\ell,i_\ell} \quad \text{such that} \quad \sum_{k=0}^{\ell} i_k n_k = i, \tag{4.4}$$

for $i \in \{0, \ldots, |L_{v_\delta}| - 1\}$. Furthermore,

$$\vartheta'_{\ell,0} = \left(\frac{\vartheta_{\ell,1}}{\vartheta_{\ell,0}}\right)^{2^{n_\ell}} - \frac{\vartheta_{\ell,1}}{\vartheta_{\ell,0}} \tag{4.5}$$

if there is no vertex $u \in V$ such that $u_\delta = v$ and $d(u) = n_\ell$.

Proof. Throughout the proof, if $i \in \{0, \ldots, n_m - 1\}$, then $i_0, \ldots, i_{m-1}$ denote the coefficients of the expansion $i = \sum_{k=0}^{m-1} i_k n_k$ such that $i_k \in \{0, \ldots, n_{k+1}/n_k - 1\}$ for $k \in \{0, \ldots, m-1\}$. We note in particular that $\{0, \ldots, |L_v|-1\} \subseteq \{0, \ldots, n_m - 1\}$ for $v \in V$, since $|L_v| \leq n \leq n_m$. We begin by showing that the assertions of the lemma

hold for a special subset of the internal vertices in the tree, before completing the proof by induction.

Let $v \in V$ be an internal vertex such that the (possibly trivial) path $r = v_0, \ldots, v_h = v$ that connects v to the root vertex $r \in V$ satisfies $v_i = (v_{i-1})_\alpha$ for $i \in \{1, \ldots, h\}$. Then

$$\beta_{v,i} = \beta_{v_{h-1},i} = \cdots = \beta_{v_0,i} = \beta_{r,i} = \prod_{k=0}^{m-1} \vartheta_{k,i_k} \quad \text{for } i \in \{0, \ldots, |L_v| - 1\}.$$

As v is an internal vertex, $d(v) = \max\{n_k \mid n_k < |L_v|\}$ by the definition of the tree $T_n^{(n_0,\ldots,n_m)}$. Thus, $d(v) = n_\ell$ for some $\ell \in \{0, \ldots, m-1\}$. Moreover, $|L_{v_\delta}| = |L_v| - n_\ell < n_{\ell+1}$, since otherwise the maximality of n_ℓ is contradicted. Consequently,

$$\frac{\beta_{v,n_\ell+i}}{\beta_{v,0}} = \frac{\vartheta_{0,i_0}}{\vartheta_{0,0}} \cdots \frac{\vartheta_{\ell-1,i_{\ell-1}}}{\vartheta_{\ell-1,0}} \frac{\vartheta_{\ell,1+i_\ell}}{\vartheta_{\ell,0}} \quad \text{for } i \in \{0, \ldots, |L_{v_\delta}| - 1\}. \tag{4.6}$$

As the quotients $\vartheta_{k,i}/\vartheta_{k,0} \in \mathbb{F}_{2^{n_{k+1}}} \subseteq \mathbb{F}_{2^{n_\ell}}$ for $k \in \{0, \ldots, \ell-1\}$, it follows that (4.4) holds with

$$\vartheta'_{\ell,i} = \left(\frac{\vartheta_{\ell,i+1}}{\vartheta_{\ell,0}}\right)^{2^{n_\ell}} - \frac{\vartheta_{\ell,i+1}}{\vartheta_{\ell,0}} \quad \text{for } i \in \{0, \ldots, n_{\ell+1}/n_\ell - 2\}.$$

Consequently, (4.5) also holds. Finally, $\vartheta'_{\ell,0}, \ldots, \vartheta'_{\ell,n_{\ell+1}/n_\ell-2}$ inherit linear independence over $\mathbb{F}_{2^{n_\ell}}$ from $\vartheta_{\ell,0}, \ldots, \vartheta_{\ell,n_{\ell+1}/n_\ell-1}$, since

$$\sum_{i=0}^{n_{\ell+1}/n_\ell-2} \lambda_{i+1}\vartheta'_{\ell,i} = 0 \quad \text{if and only if} \quad \sum_{i=1}^{n_{\ell+1}/n_\ell-1} \lambda_i \frac{\vartheta_{\ell,i}}{\vartheta_{\ell,0}} \in \mathbb{F}_{2^{n_\ell}}$$

for $\lambda_1, \ldots, \lambda_{n_{\ell+1}/n_\ell-1} \in \mathbb{F}_{2^{n_\ell}}$.

We now proceed by induction on the depth of v, i.e., on the length h of the path $r = v_0, \ldots, v_h = v$ that connects v to the root vertex of the tree. If $v \in V$ is an internal vertex of depth zero, then v is the root of the tree which is covered by the already proved case. Therefore, let h be a positive integer, and suppose that the assertions of the lemma holds for all internal vertices with depth less than h. Let $v \in V$ be an internal vertex of depth h (if no such vertex exists, then we are done) and $r = v_0, \ldots, v_h = v$ be the path that connects v to the root vertex $r \in V$. We may assume that $v_i = (v_{i-1})_\delta$ for some $i \in \{1, \ldots, h\}$. Let j the maximum of all such indices. Then v_{j-1} is an internal vertex. Thus, $d(v_{j-1}) = n_k$ for some $k \in \{0, \ldots, m-1\}$. Consequently, the induction hypothesis and the choice of j imply that there exists a basis $\{\vartheta''_{k,0}, \ldots, \vartheta''_{k,n_{k+1}/n_k-1}\}$ of $\mathbb{F}_{2^{n_{k+1}}}/\mathbb{F}_{2^{n_k}}$ such that

$$\beta_{v,i} = \beta_{v_{h-1},i} = \cdots = \beta_{v_j,i} = \frac{\vartheta_{0,i_0}}{\vartheta_{0,0}} \cdots \frac{\vartheta_{k-1,i_{k-1}}}{\vartheta_{k-1,0}} \vartheta''_{k,i_k} \quad \text{for } i \in \{0, \ldots, |L_v| - 1\}. \tag{4.7}$$

As v is an internal vertex, $d(v) = n_\ell$ for some $\ell \in \{0, \ldots, m-1\}$. Moreover, the definition of the tree $T_n^{(n_0,\ldots,n_m)}$ implies that

$$n_\ell = \max\{n_t \mid n_t < |L_v|\} \leq \max\{n_t \mid n_t < |L_{v_{j-1}}|\} = n_k,$$

since v is descended from v_{j-1}. If $\ell = k$, then (4.7) implies that (4.4) holds with

$$\vartheta'_{\ell,i} = \left(\frac{\vartheta''_{\ell,i+1}}{\vartheta''_{\ell,0}}\right)^{2^{n_\ell}} - \frac{\vartheta''_{\ell,i+1}}{\vartheta''_{\ell,0}} \quad \text{for } i \in \{0, \ldots, n_{\ell+1}/n_\ell - 2\},$$

which inherit linear independence over $\mathbb{F}_{2^{n_\ell}}$ from $\vartheta''_{k,0}, \ldots, \vartheta''_{k,n_{k+1}/n_k-1}$. Moreover, we must have $j = h$, since otherwise the maximality of j implies that v is descended from $(v_j)_\alpha$, which in-turn implies that $n_k = n_\ell < |L_v| \leq d(v_j) \leq n_k$. It follows that $u = v_{h-1}$ satisfies $u_\delta = v$ and $d(u) = n_\ell$. Consequently, we are not required to show that (4.5) holds in this case.

If $\ell < k$, then $|L_{v_\delta}| = |L_v| - n_\ell < n_{\ell+1} \leq n_k$, since $n_\ell = \max\{n_t \mid n_t < |L_v|\}$. Thus, (4.7) implies that (4.6) holds, which we have already shown to be sufficient for the two assertions of the lemma to hold. Hence, the lemma follows by induction. □

Remark 4.11. Lemma 4.10 implies that $\beta_{v_\delta,0} = \vartheta'_{\ell,0}$ lies in the subfield $\mathbb{F}_{2^{n_{\ell+1}}}$ regardless of the choice of bases used in the construction of Theorem 4.3. Consequently, if $\beta_{v_\delta,0}$ cannot be forced to equal one, then it may still be possible to reduce the cost of the multiplications performed in Lines 10–17 of Algorithm 7 (similarly, Lines 4–11 of Algorithm 8) by choosing the representation of the elements in $\mathbb{F}$ so that the cost of multiplication is reduced whenever one of the multiplicands belongs to $\mathbb{F}_{2^{n_{\ell+1}}}$. Such optimisations have previously been shown to be beneficial in practice, particularly for multiplications by elements of small subfields, by Bernstein and Chou [3] and Chen et al. [8]. These considerations also extend to the algorithms for conversion between the LCH and the Newton or Lagrange bases of Sections 3.2–3.4. For these algorithms, if the tower used in the construction of the basis contains small subfields, then Lemma 4.10 can be used to show that some of the precomputed elements $\varphi_v(u, \sigma_{v,i})$ belong to small subfields. Consequently, if the initial shift parameter λ also lies in a small subfield, as is the case when it is zero, then so too do some of the multiplicands in the base cases of the algorithms.

Proof of Theorem 4.9. Suppose there exists $t \in \{0, \ldots, m-1\}$ such that (4.3) holds, and there exists a vertex $v \in V$ such that $d(v) = n_t$. If $v = u_\delta$ for some $u \in V$, then $d(u) \neq n_t$, since otherwise $n_{t+1} = 2n_t < n_t + |L_v| = n_t + (|L_u| - n_t) = |L_u|$, contradicting the maximality of n_t. Therefore, Lemma 4.10 implies that

$$\beta_{v_\delta,0} = \frac{\vartheta_{0,0}}{\vartheta_{0,0}} \cdots \frac{\vartheta_{t-1,0}}{\vartheta_{t-1,0}} \left(\left(\frac{\vartheta_{t,1}}{\vartheta_{t,0}} \right)^{2^{n_t}} - \frac{\vartheta_{t,1}}{\vartheta_{t,0}} \right) = \mathrm{Tr}_{\mathbb{F}_{2^{n_{t+1}}}/\mathbb{F}_{2^{n_t}}} \left(\frac{\vartheta_{t,1}}{\vartheta_{t,0}} \right) = 1,$$

as required. □

4.3. **Generalised Cantor basis.** Gao and Mateer propose a generic method of constructing Cantor bases in the appendix of their paper [14]. We generalise their construction to one that extends an arbitrarily chosen basis of $\mathbb{F}_{2^t}/\mathbb{F}_2$ to a basis of $\mathbb{F}_{2^{2^m t}}/\mathbb{F}_2$ that enjoys properties similar to those offered by Cantor bases. The original construction of Gao and Mateer then corresponds to the case $t = 1$. By generalising their construction, we are able to take advantage of quadratic extensions in a different manner to the previous section in order to provide a greater selection of reduction trees.

Hereafter, we assume that $\mathbb{F}_{q^{2^m}} \subseteq \mathbb{F}$ with $m \in \mathbb{N}$ positive and $q = 2^t$ for some positive $t \in \mathbb{N}$. We also fix a basis $\beta_0, \ldots, \beta_{2^m t-1}$ of $\mathbb{F}_{q^{2^m}}/\mathbb{F}_2$ which is given by the following generalisation of the construction of Gao and Mateer: choose a basis $\{\vartheta_0, \ldots, \vartheta_{t-1}\}$ of $\mathbb{F}_q/\mathbb{F}_2$, choose $\beta_{(2^m-1)t}, \ldots, \beta_{2^m t-1} \in \mathbb{F}_{q^{2^m}}$ such that

$$\mathrm{Tr}_{\mathbb{F}_{q^{2^m}}/\mathbb{F}_q}\left(\beta_{(2^m-1)t+i}\right) = \vartheta_i \quad \text{for } i \in \{0, \ldots, t-1\},$$

and recursively define

$$\beta_i = \beta_{i+t}^q - \beta_{i+t} \quad \text{for } i \in \{0, \ldots, (2^m-1)t-1\}.$$

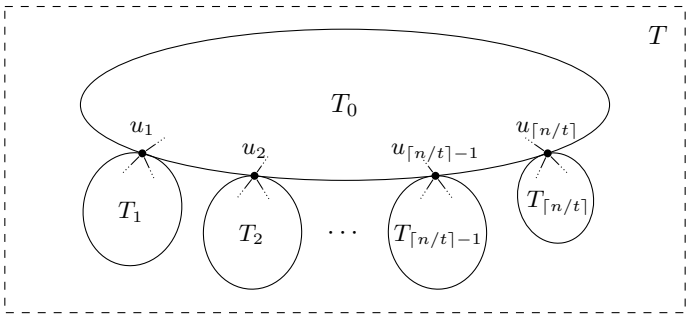

FIGURE 8. Construction of Theorem 4.12.

For this construction, we provide generalisations of the properties of Cantor bases given in Lemma 2.9. The properties are then used to prove the following theorem, which provides a method of constructing reduction trees for the vectors $(\beta_0, \dots, \beta_{n-1})$ for $n \in \{1, \dots, 2^m t\}$.

Theorem 4.12. *Let $n \in \{1, \dots, 2^m t\}$, $T_0 = (V_0, E_0)$ be a full binary tree with $\lceil n/t \rceil$ leaves such that $\mathrm{Im}(d) \subseteq \{0, 2^0, 2^1, \dots, 2^{\lceil \log_2 \lceil n/t \rceil \rceil - 1}\}$, and $T_i = (V_i, E_i)$ be a reduction tree for $(\vartheta_0, \dots, \vartheta_{\min(n-(i-1)t, t)-1})$, for $i \in \{1, \dots, \lceil n/t \rceil\}$. Let $u_1, \dots, u_{\lceil n/t \rceil} \in V_0$ be the leaves of T_0, ordered such that for all $i, j \in \{1, \dots, \lceil n/t \rceil\}$ with $i < j$, there exists an internal vertex $v \in V_0$ with $u_i \in L_{v_\alpha}$ and $u_j \in L_{v_\delta}$. Construct a new tree $T = (V, E)$ by identifying the root vertex of T_i with u_i for $i \in \{1, \dots, \lceil n/t \rceil\}$, as shown in Figure 8. Then T is a reduction tree for $(\beta_0, \dots, \beta_{n-1})$.*

Theorem 4.12 provides greater freedom than Theorem 4.3 by not requiring the inequality $d(v_\delta) \leq d(v)$ to hold for $v \in V$ that are initially internal vertices in the tree T_0. Proposition 2.7 guarantees the existence of trees $T_1, \dots, T_{\lceil n/t \rceil}$ to use in the construction. We can of course provide a better selection for these trees if the methods of Sections 4.1 and 4.2 are used to construct the basis $\{\vartheta_0, \dots, \vartheta_{t-1}\}$.

The remainder of the section is dedicated to the proof of Theorem 4.12.

Lemma 4.13. *The following hold:*

(1) $\beta_i = \sum_{r=0}^{j} \binom{j}{r} \beta_{i+jt}^{q^r}$ *for* $i \in \{0, \dots, (2^m - j)t - 1\}$ *and* $j \in \{0, \dots, 2^m - 1\}$,
(2) $\beta_i = \beta_{i+2^k t}^{q^{2^k}} - \beta_{i+2^k t}$ *for* $i \in \{0, \dots, (2^m - 2^k)t - 1\}$ *and* $k \in \{0, \dots, m-1\}$,
(3) $\beta_i = \vartheta_i$ *for* $i \in \{0, \dots, t-1\}$,
(4) $\beta_0, \dots, \beta_{2^k t - 1} \in \mathbb{F}_{q^{2^k}}$ *for* $k \in \{0, \dots, m-1\}$, *and*
(5) $\beta_0, \dots, \beta_{2^m t - 1}$ *are linearly independent over* $\mathbb{F}_2$.

Our proof of Lemma 4.13 generalises arguments found in Cantor's paper [7] and the paper of Gao and Mateer [14].

Proof. We prove (1) by induction on j. It is clear that (1) holds if $j = 0$, regardless of the value of i. Therefore, suppose that (1) holds for some $j \in \{0, \dots, 2^m - 2\}$ and each $i \in \{0, \dots, (2^m - j)t - 1\}$. Then

$$\beta_{i+t} = \sum_{r=0}^{j} \binom{j}{r} \beta_{i+t+jt}^{q^r} = \sum_{r=0}^{j} \binom{j}{r} \beta_{i+(j+1)t}^{q^r}$$

for $i \in \{0, \dots, (2^m - j - 1)t - 1\}$. As $(2^m - j - 1)t - 1 \leq (2^m - 1)t - 1$, it follows that

$$\begin{aligned}
\beta_i &= \beta_{i+t}^q - \beta_{i+t} \\
&= \sum_{r=0}^{j} \binom{j}{r} \left(\beta_{i+(j+1)t}^{q^{r+1}} - \beta_{i+(j+1)t}^{q^r} \right) \\
&= \beta_{i+(j+1)t} + \sum_{r=1}^{j+1} \left(\binom{j}{r-1} + \binom{j}{r} \right) \beta_{i+(j+1)t}^{q^r} \\
&= \beta_{i+(j+1)t} + \sum_{r=1}^{j+1} \binom{j+1}{r} \beta_{i+(j+1)t}^{q^r} \\
&= \sum_{r=0}^{j+1} \binom{j+1}{r} \beta_{i+(j+1)t}^{q^r}
\end{aligned}$$

for $i \in \{0, \dots, (2^m - j - 1)t - 1\}$. Thus, property (1) holds.

For $i, j \in \mathbb{N}$, Lucas' lemma [22, p. 230] (see also [13]) implies that $\binom{i}{j} \equiv 1 \pmod 2$ if and only if $[j]_k \leq [i]_k$ for $k \in \mathbb{N}$. Using the lemma, property (2) follows from property (1) by setting $j = 2^k$. Similarly, Lucas' lemma and property (1) with $j = (2^{m-k} - 1)2^k$ implies that

$$\begin{aligned}
\beta_{(2^k-1)t+i} &= \sum_{r=0}^{(2^{m-k}-1)2^k} \binom{(2^{m-k} - 1)2^k}{r} \beta_{(2^m-1)t+i}^{q^r} \\
&= \sum_{r=0}^{2^{m-k}-1} \beta_{(2^m-1)t+i}^{q^{2^k r}} \\
&= \mathrm{Tr}_{\mathbb{F}_{q^{2^m}}/\mathbb{F}_{q^{2^k}}} \left(\beta_{(2^m-1)t+i} \right)
\end{aligned}$$

for $i \in \{0, \dots, t-1\}$ and $k \in \{0, \dots, m-1\}$. Setting $k = 0$, property (3) follows by the choice of $\beta_{(2^m-1)t}, \dots, \beta_{2^m t - 1}$. Moreover, the trace formula implies that $\beta_{(2^k-1)t}, \dots, \beta_{2^k t - 1} \in \mathbb{F}_{q^{2^k}}$ for $k \in \{0, \dots, m-1\}$, after-which the recursive definition of $\beta_0, \dots, \beta_{(2^m-1)t-1}$ implies that property (4) holds.

Property (3) implies that $\beta_0, \dots, \beta_{t-1}$ are linearly independent over $\mathbb{F}_2$, and belong to the kernel of the $\mathbb{F}_2$-linear map $\varphi : \mathbb{F} \to \mathbb{F}$ given by $\omega \mapsto \omega^q - \omega$. Thus, for $i \in \{2, \dots, 2^m\}$, any nontrivial $\mathbb{F}_2$-linear relation amongst $\beta_0, \dots, \beta_{it-1}$, which necessarily involves at least one of $\beta_{(i-1)t}, \dots, \beta_{it-1}$, translates under φ to a nontrivial relation amongst $\beta_0, \dots, \beta_{(i-1)t-1}$. It follows that property (5) holds by induction on i. □

Lemma 4.13 provides generalisations of the properties of Cantor bases given in Lemma 2.9. The following lemma may be viewed as a partial generalisation Proposition 2.8.

Lemma 4.14. *Let (V, E) be a full binary tree with $n \leq 2^m t$ leaves that satisfies the following conditions:*

(1) *if $v \in V$ such that $|L_v| > t$, then $d(v) \in \{2^k t \mid k < \lceil \log_2 \lceil n/t \rceil \rceil\}$, and*

(2) *if $v \in V$ such that $|L_v| \leq t$, and v is either the root of the tree or a child of a vertex $v' \in V$ with $|L_{v'}| > t$, then the subtree of (V, E) rooted on v is a reduction tree for $(\beta_0, \ldots, \beta_{|L_v|-1})$.*

Then (V, E) is a reduction tree for $(\beta_0, \ldots, \beta_{n-1})$.

The following technical lemma is required for the proof of Lemma 4.14.

Lemma 4.15. *Let $\mu \in \mathbb{F}^n$ have linearly independent entries over $\mathbb{F}_2$, (V, E) be a reduction tree for μ, and $\omega \in \mathbb{F}$ be nonzero. Then (V, E) is a reduction tree for $\omega\mu$.*

Proof. We prove the lemma by induction on n. The lemma holds trivially for $n = 1$. Therefore, let $n \geq 2$ and suppose that the lemma holds for all smaller values of n. Let $\mu = (\mu_0, \ldots, \mu_{n-1}) \in \mathbb{F}^n$ have linearly independent entries over $\mathbb{F}_2$, (V, E) be a reduction tree for μ, $r \in V$ be the root vertex of the tree, and $\omega \in \mathbb{F}$ be nonzero. Then $\mu_i/\mu_0 = \omega\mu_i/(\omega\mu_0) \in \mathbb{F}_{2^{d(r)}}$ for $i \in \{0, \ldots, d(r) - 1\}$, the induction hypothesis implies that the subtree rooted on r_α is a reduction tree for $\omega\alpha(\mu, d(r)) = \alpha(\omega\mu, d(r))$, and the subtree rooted on r_δ is a reduction tree for $\delta(\mu, d(r)) = \delta(\omega\mu, d(r))$. Therefore, (V, E) is a reduction tree for $\omega\mu$. Hence, the lemma follows by induction. □

Proof of Lemma 4.14. We prove the lemma by induction of n. Condition (2) implies that the lemma holds trivially if $n \leq t$. Therefore, let $n \in \{t + 1, \ldots, 2^m t\}$ and suppose that the lemma is true for all smaller values of n. Let (V, E) be a full binary tree with n leaves that satisfies conditions (1) and (2) of the lemma. Let $\beta = (\beta_0, \ldots, \beta_{n-1})$ and $r \in V$ be the root vertex of the tree. Then $|L_r| = n > t$. Thus, (1) implies that $d(r) = 2^k t$ for some $k < \lceil \log_2 \lceil n/t \rceil \rceil \leq m$. Therefore, property (4) of Lemma 4.13 implies that $\beta_i/\beta_0 \in \mathbb{F}_{2^{d(r)}}$ for $i \in \{0, \ldots, d(r) - 1\}$. Moreover, properties (2) and (4) of the lemma imply that

$$\alpha(\beta, d(r)) = \big(\beta_0, \ldots, \beta_{d(r)-1}\big) \quad \text{and} \quad \delta(\beta, d(r)) = \frac{1}{\beta_0}\big(\beta_0, \ldots, \beta_{n-d(r)-1}\big). \tag{4.8}$$

The subtrees of (V, E) rooted on r_α and r_δ satisfy the conditions of the lemma through inheritance. Thus, the induction hypothesis and (4.8) imply that the subtree rooted on r_α is a reduction tree for $\alpha(\beta, d(r))$. Similarly, the induction hypothesis, Lemma 4.15 and (4.8) imply that the subtree root on r_δ is a reduction tree for $\delta(\beta, d(r))$. Therefore, (V, E) is a reduction tree for β. Hence, the lemma follows by induction. □

We now complete the proof of Theorem 4.12 by showing that its construction produces binary trees that satisfy the conditions of Lemma 4.14.

Proof of Theorem 4.12. For $i \in \{1, \ldots, \lceil n/t \rceil\}$, (V_i, E_i) is a full binary tree with $\min(n-(i-1)t, t)$ leaves. Therefore, it is clear that (V, E) is a full binary tree with

$$\sum_{i=1}^{\lceil n/t \rceil} \min(n - (i-1)t, t) = n$$

leaves. We show that (V, E) satisfies the conditions of Lemma 4.14.

Suppose there exists a vertex $v \in V$ such that $|L_v| > t$. Then v is not descended from or equal to u_i for $i \in \{1, \ldots, \lceil n/t \rceil\}$, since (V_i, E_i) has at most t leaves. By the choice of (V_0, E_0), it follows that 2^k of the vertices u_i are descended from v_α for some $k < \lceil \log_2 \lceil n/t \rceil \rceil$. Let $i_1, \ldots, i_{2^k}$ be the indices of these vertices. Then

$i_1, \ldots, i_{2^k} < \lceil n/t \rceil$, since the ordering of the vertices u_i implies that $u_{\lceil n/t \rceil}$ must be equal to v_δ or one of its descendants. It follows that the subtrees of (V, E) rooted on $u_{i_1}, \ldots, u_{i_{2^k}}$ each have t leaves. Thus, $d(v) = 2^k t$ for some $k < \lceil \log_2 \lceil n/t \rceil \rceil$. Therefore, (V, E) satisfies condition (1) of Lemma 4.14.

Suppose there exists a vertex $v \in V$ such that $|L_v| \leq t$, and v is either the root of the tree or the child of a vertex $v' \in V$ with $|L_{v'}| > t$. Then v is descended from or equal to u_i for some $i \in \{1, \ldots, \lceil n/t \rceil\}$, since the subtrees rooted on $u_1, \ldots, u_{\lceil n/t \rceil - 1}$ each have t leaves, while the subtree rooted on $u_{\lceil n/t \rceil}$ has at least one leaf. If v is the root of (V, E), then $(V, E) = (V_i, E_i)$. If v is the child of a vertex $v' \in V$ such that $|L_{v'}| > t$, and thus $|L_{v'}| > |L_{u_i}|$, then u_i is a descendant of v'. In either case, v is equal to u_i. Thus, the choice of (V_i, E_i) and property (3) of Lemma 4.13 imply that the subtree of (V, E) rooted on v is a reduction tree for $(\beta_0, \ldots, \beta_{|L_v|-1})$. Hence, (V, E) satisfies condition (2) of Lemma 4.14. □

References

1. Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza, *Computational integrity with a public random string from quasi-linear PCPs*, Advances in cryptology—EUROCRYPT 2017. Part III, Lecture Notes in Comput. Sci., vol. 10212, Springer, Cham, 2017, pp. 551–579.
2. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev, *Scalable, transparent, and post-quantum secure computational integrity*, Cryptology ePrint Archive, Report 2018/046, 2018, `https://eprint.iacr.org/2018/046`.
3. Daniel J. Bernstein and Tung Chou, *Faster binary-field multiplication and faster binary-field MACs*, Selected areas in cryptography—SAC 2014, Lecture Notes in Comput. Sci., vol. 8781, Springer, Cham, 2014, pp. 92–111.
4. Daniel J. Bernstein, Tung Chou, and Peter Schwabe, *McBits: Fast constant-time code-based cryptography*, Cryptographic Hardware and Embedded Systems—CHES 2013, Lecture Notes in Comput. Sci., vol. 8086, Springer, Berlin, 2013, pp. 250–272.
5. James R. Bitner, Gideon Ehrlich, and Edward M. Reingold, *Efficient generation of the binary reflected Gray code and its applications*, Comm. ACM **19** (1976), no. 9, 517–521.
6. Richard P. Brent, Pierrick Gaudry, Emmanuel Thomé, and Paul Zimmermann, *Faster multiplication in* GF(2)[x], Algorithmic number theory—ANTS 2008, Lecture Notes in Comput. Sci., vol. 5011, Springer, Berlin, 2008, pp. 153–166.
7. David G. Cantor, *On arithmetical algorithms over finite fields*, J. Combin. Theory Ser. A **50** (1989), no. 2, 285–300.
8. Ming-Shing Chen, Chen-Mou Cheng, Po-Chun Kuo, Wen-Ding Li, and Bo-Yin Yang, *Faster multiplication for long binary polynomials*, 2017, `arXiv:1708.09746 [cs.SC]`.
9. ———, *Multiplying boolean polynomials with Frobenius partitions in additive fast Fourier transform*, 2018, `arXiv:1803.11301 [cs.SC]`.
10. Tung Chou, *McBits revisited*, Cryptographic Hardware and Embedded Systems—CHES 2017, Lecture Notes in Comput. Sci., vol. 10529, Springer, Cham, 2017, pp. 213–231.
11. Nicholas Coxon, *Fast systematic encoding of multiplicity codes*, `arXiv:1704.07083 [cs.IT]`, Apr 2017.
12. ———, *Fast Hermite interpolation and evaluation over finite fields of characteristic two*, `arXiv:1807.00645 [cs.SC]`, July 2018.
13. N. J. Fine, *Binomial coefficients modulo a prime*, Amer. Math. Monthly **54** (1947), 589–592.
14. Shuhong Gao and Todd Mateer, *Additive fast Fourier transforms over finite fields*, IEEE Trans. Inform. Theory **56** (2010), no. 12, 6265–6272.
15. David Harvey, *A cache-friendly truncated FFT*, Theoret. Comput. Sci. **410** (2009), no. 27-29, 2649–2658.
16. Donald E. Knuth, *The art of computer programming. Vol. 4, Fasc. 2*, Addison-Wesley, Upper Saddle River, NJ, 2005.

17. Robin Larrieu, *The truncated Fourier transform for mixed radices*, ISSAC'17—Proceedings of the 2017 ACM International Symposium on Symbolic and Algebraic Computation, ACM, New York, 2017, pp. 261–268.
18. Wen-Ding Li, Ming-Shing Chen, Po-Chun Kuo, Chen-Mou Cheng, and Bo-Yin Yang, *Frobenius additive fast Fourier transform*, 2018, `arXiv:1802.03932 [cs.SC]`.
19. Sian-Jheng Lin, Tareq Y. Al-Naffouri, and Yunghsiang S. Han, *FFT algorithm for binary extension finite fields and its application to Reed-Solomon codes*, IEEE Trans. Inform. Theory **62** (2016), no. 10, 5343–5358.
20. Sian-Jheng Lin, Tareq Y. Al-Naffouri, Yunghsiang S. Han, and Wei-Ho Chung, *Novel polynomial basis with fast Fourier transform and its application to Reed–Solomon erasure codes*, IEEE Trans. Inform. Theory **62** (2016), no. 11, 6284–6299.
21. Sian-Jheng Lin, Wei-Ho Chung, and Yunghsiang S. Han, *Novel polynomial basis and its application to Reed-Solomon erasure codes*, 55th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2014, IEEE Computer Soc., Los Alamitos, CA, 2014, pp. 316–325.
22. Edouard Lucas, *Théorie des Fonctions Numériques Simplement Périodiques. [Continued]*, Amer. J. Math. **1** (1878), no. 3, 197–240.
23. Todd Mateer, *Fast Fourier Transform algorithms with applications*, ProQuest LLC, Ann Arbor, MI, 2008, Ph.D. thesis–Clemson University.
24. J. van der Hoeven, *Notes on the Truncated Fourier Transform*, Tech. Report 2005-5, Université Paris-Sud, Orsay, France, 2005.
25. Joris van der Hoeven, *The truncated Fourier transform and applications*, ISSAC 2004—Proceedings of the 2004 international symposium on Symbolic and algebraic computation, ACM, New York, 2004, pp. 290–296.
26. Joris van der Hoeven and Éric Schost, *Multi-point evaluation in higher dimensions*, Appl. Algebra Engrg. Comm. Comput. **24** (2013), no. 1, 37–52.
27. Joachim von zur Gathen, *Functional decomposition of polynomials: the tame case*, J. Symbolic Comput. **9** (1990), no. 3, 281–299.
28. Joachim von zur Gathen and Jürgen Gerhard, *Arithmetic and factorization of polynomial over $\mathbb{F}_2$ (extended abstract)*, ISSAC '96—Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation, ACM, New York, 1996, pp. 1–9.
29. Y. Wang and X. Zhu, *A fast algorithm for the Fourier transform over finite fields and its VLSI implementation*, IEEE Journal on Selected Areas in Communications **6** (1988), no. 3, 572–577.

INRIA Saclay–Île-de-France & Laboratoire d'Informatique, École polytechnique, 91128 Palaiseau Cedex, France

E-mail address: `nicholas.coxon@inria.fr`