Constructing Orthogonal Designs in Powers of Two: Gröbner Bases Meet Equational Unification

Ilias Kotsireas[1], Temur Kutsia[2], and Dimitris E. Simos*[3]

1 **Wilfrid Laurier University**
Waterloo, Ontario, Canada
ikotsire@wlu.ca

2 **RISC, Johannes Kepler University**
Altenbergerstrasse 69, A-4040 Linz, Austria
kutsia@risc.jku.at

3 **SBA Research**
Favoritenstrasse 16, A-1040 Vienna, Austria
dsimos@sba-research.org

Abstract

In the past few decades, design theory has grown to encompass a wide variety of research directions. It comes as no surprise that applications in coding theory and communications continue to arise, and also that designs have found applications in new areas. Computer science has provided a new source of applications of designs, and simultaneously a field of new and challenging problems in design theory. In this paper, we revisit a construction for orthogonal designs using the multiplication tables of Cayley-Dickson algebras of dimension 2^n. The desired orthogonal designs can be described by a system of equations with the aid of a Gröbner basis computation. For orders greater than 16 the combinatorial explosion of the problem gives rise to equations that are unfeasible to be handled by traditional search algorithms. However, the structural properties of the designs make this problem possible to be tackled in terms of rewriting techniques, by equational unification. We establish connections between central concepts of design theory and equational unification where equivalence operations of designs point to the computation of a minimal complete set of unifiers. These connections make viable the computation of some types of orthogonal designs that have not been found before with the aforementioned algebraic modelling.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, F.4.1 Mathematical Logic, G.2.1 Combinatorics

Keywords and phrases Orthogonal designs, unification theory, algorithms, Gröbner bases.

Digital Object Identifier ***

1 Introduction

Orthogonal designs are an important class of combinatorial designs. They are of great interest in applications for wireless communication [15] and in statistics [11]. Even though there exist many combinatorial constructions for orthogonal designs [6], ones that originate from Cayley-Dickson algebras [7, 8] have not been explored enough. In particular, as we exemplify in this work, these algebras can provide a general framework for obtaining orthogonal designs

* The work of the third author was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

submitted to 26nd International Conference on Rewriting Techniques and Applications.
Editor: *** Editor; pp. 1–18

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

for powers of two. Designs in these orders are also of theoretical interest due to their connection to the asymptotic existence of orthogonal designs [6].

Contribution. In this paper, after revisiting past methods we formulate orthogonal design problems in terms of equational unification. In particular, the Cayley-Dickson formulation gives rise to a polynomial system of equations of a specific form, that due to its size cannot be handled by traditional search algorithms. By establishing and proving connections between central concepts of the theory of orthogonal designs and equational unification, we are able to completely tackle these systems of equations, where each solution of them gives rise to an orthogonal design. The efficiency of the unification algorithms needed to solve the corresponding orthogonal design problems is evident also by the fact that we found some types of orthogonal designs, that were not known before with this algebraic modelling of Cayley-Dickson algebras. Our approach not only reports the orthogonal designs, but also constructs the corresponding design matrices. In this way, we *always* give a *constructive* solution to the problem which is not always the case with other approaches used in design theory as we explain in the last section. Last but not least, we would like to emphasize the novel connections we established between base orthogonal designs, a notion introduced in this paper, and minimal complete sets of unifiers, as a means to advance the knowledge in the field of design theory (orthogonal design equivalence among other topics) and also benefit from the algorithmic notions of unification theory as we applied them in this paper.

Structure of the paper. In Section 2 we give some details regarding orthogonal designs and list some of their applications. Afterwards, in Section 3 we detail the algebraic framework for constructing orthogonal desings via computation algebra where we also introduce some new terms for designs. Some first connections with unification theory are also shown. In the subsequent section we give some basic notions of unification theory while in Section 5 we establish additional connections of designs with concepts of unification theory that allow us to formulate orthogonal design problems as unification problems. In Section 6 we describe the unification algorithms we developed for solving the unification problems and in the last section, we translate the solutions obtained via unifiers back to orthogonal designs.

2 Orthogonal Designs

In this section, we give some details regarding orthogonal designs. We provide the necessary definitions and related concepts that will be needed for our approach and list also some applications of orthogonal designs that are of broader interest.

2.1 Definitions and Related Concepts

An *orthogonal design* of order n and type $(s_1, s_2, \ldots, s_u)$ $(s_i > 0)$, denoted $OD(n; s_1, s_2, \ldots, s_u)$, on the commuting variables $x_1, x_2, \ldots, x_u$, is an $n \times n$ matrix D with entries from $\{0, \pm x_1, \pm x_2, \ldots, \pm x_u\}$ such that

$$DD^T = \left(\sum_{i=1}^{u} s_i x_i^2 \right) I_n$$

where by I_n we denote the identity matrix of order n. Alternatively, the rows of D are formally orthogonal and each row has precisely s_i entries of the type $\pm x_i$. The design matrix D, may be considered as a matrix with entries in the field of quotients of the integral domain

$\mathbb{Z}[x_1, x_2, \ldots, x_u]$. In [5], where this was first defined, it was mentioned that

$$D^T D = \left(\sum_{i=1}^{u} s_i x_i^2 \right) I_n$$

and so our alternative description of D applies equally well to the columns of D. It was also shown in [5] that $u \leq \rho(n)$, where $\rho(n)$ (Radon's function) is defined by $\rho(n) = 8c + 2^d$, when $n = 2^a b$, b odd, $a = 4c + d$, $0 \leq d < 4$. D will be called a *full orthogonal design*, if $n = s_1 + s_2 + \ldots + s_u$. Due to the Equating-Killing Lemma, given below, which is of central importance in the theory of Orthogonal Designs, one is interested in full orthogonal designs.

▶ **Lemma 1** (The Equating and Killing Lemma, Geramita and Seberry [6]). *If D is an orthogonal design $OD(n; s_1, s_2, \ldots, s_u)$ on the commuting variables $\{0, \pm x_1, \pm x_2, \ldots, \pm x_u\}$ then there exists an orthogonal design:*

(i) $OD(n; s_1, s_2, \ldots, s_i + s_j, \ldots, s_u)$ *($s_i = s_j$, equating variables)*

(ii) $OD(n; s_1, s_2, \ldots, s_{j-1}, s_{j+1}, \ldots, s_u)$ *($s_j = 0$, killing variables)*

on the $u - 1$ commuting variables $\{0, \pm x_1, \pm x_2, \ldots, \pm x_{j-1}, \pm x_{j+1}, \ldots, \pm x_u\}$.

We also list the Doubling Lemma, which will be needed in the last section of the paper.

▶ **Lemma 2** (The Doubling Lemma, Geramita and Seberry [6]). *If there exists an orthogonal design of order n and type $(s_1, s_2, \ldots, s_u)$, then there exists orthogonal designs of type*

(i) $(e_1 s_1, e_2 s_2, \ldots, e_u s_u)$ *where $e_i = 1$ or 2,*

(ii) $(s_1, s_1, f s_2, \ldots, f s_u)$ *where $f = 1$ or 2.*

▶ **Example 3.** We give an example of some small orthogonal designs, and how we can obtain one from another due to Lemma 1 and related equivalence operations.

$$\begin{bmatrix} x_1 & x_2 \\ x_2 & -x_1 \end{bmatrix}, \begin{bmatrix} x_1 & -x_2 & -x_3 & -x_4 \\ x_2 & x_1 & -x_4 & x_3 \\ x_3 & x_4 & x_1 & -x_2 \\ x_4 & -x_3 & x_2 & x_1 \end{bmatrix}, \begin{bmatrix} x_1 & x_2 & x_2 & x_4 \\ -x_2 & x_1 & x_4 & -x_2 \\ -x_2 & -x_4 & x_1 & x_2 \\ -x_4 & x_2 & -x_2 & x_1 \end{bmatrix}, \begin{bmatrix} x_1 & 0 & -x_3 & 0 \\ 0 & x_1 & 0 & x_3 \\ c & 0 & x_1 & 0 \\ 0 & -x_3 & 0 & x_1 \end{bmatrix}$$

$OD(2; 1, 1)$ $OD(4; 1, 1, 1, 1)$ $OD(4; 1, 1, 2)$ $OD(4; 1, 1)$

- $OD(4; 1, 1, 2)$ can be obtained from $OD(4; 1, 1, 1, 1)$ by setting $x_3 = -x_2$ in its design matrix.
- $OD(4; 1, 1)$ can be obtained from $OD(4; 1, 1, 1, 1)$ by setting $x_2 = x_4 = 0$ in its design matrix.

It is important to note here that in the first case the transformation is composed by the equating operation of the Equating and Killing Lemma and also changing the sign of the variable. The last operation leaves invariant the type of the design, however changes the design matrix. We describe more formally *equivalence of orthogonal designs* taken from [18].

Given two designs D_1 and D_2 of the same order, we say that D_2 is a *variant* of D_1, if it is obtained from D_1 by the following operations, performed in any order and any number of times:

1. Multiply one row (one column) by -1.
2. Swap two rows (columns).
3. Rename or negate a variable throughout the design.

It is easy to prove that the relation of being a variant is an equivalence relation. Below we write $D_1 \simeq D_2$ to express this fact. Note also that if $D_1 \simeq D_2$, then D_1 and D_2 have the same type. This follows directly from the definition of orthogonal design.

The general discussion of equivalence of orthogonal designs is very difficult because of the lack of a nice canonical form. It also means that it is quite difficult to decide whether or not two given orthogonal designs of the same order are equivalent. To the best of our knowledge, there has been little effort contributing at this point. In [18], where the above mentioned notion of equivalence was introduced, some designs for small orders have been classified by hand.

The approach proposed in this paper, besides providing a systematic search method for orthogonal designs in order of powers of two, also exhibits some interesting connections between the Equating and Killing Lemma and equivalence of orthogonal designs on the one hand, and fundamental concepts of unification theory such as subsumption and equi-generality on the other hand, as we can see below in Section 5.

2.2 Applications of Orthogonal Designs

We give some references to works describing applications of orthogonal designs. We do not aim to provide a comprehensive, or by all means complete, treatment of the subject, as this is not the purpose of the present paper. We are merely interesting in giving a flavor of the many different application areas involved, in order to exhibit that while orthogonal designs are specialized types of combinatorial structures their applications are of a broader interest.

As first noted in [11], orthogonal designs are used in statistics where they generate optimal statistical designs used in weighing experiments. A special case of orthogonal designs, the so called *Hadamard matrices* play an important role also in coding theory where they have been used to generate the so called Hadamard codes ([10]), i.e. error-correcting codes that correct the maximum number of errors. It is worthwhile to note that, a Hadamard code was used during the 1971 space probe Mariner 9 mission by NASA to correct for picture transmission error. The Mariner 9 mission and the Coding Theory used in that project are the subjects of [12] and [16]. Recently, complex orthogonal designs were used in [15] to generate space-time block codes, a relatively new paradigm for communication over Rayleigh fading channels using multiple transmit antennas. In this case, the orthogonal structure of the space-time block code derived by the orthogonal design gives a maximum-likelihood decoding algorithm which is based only on linear processing at the receiver.

Orthogonal designs are also used in telecommunications where they generate sequences used in digital communications and in optics for the improvement of the quality and resolution of image scanners. More details, regarding their applications in communications and signal/image processing can be found in [6, 13, 17].

3 Orthogonal Designs via Computational Algebra

In this section, we revisit a construction for orthogonal designs based on the multiplication tables of algebras of order n. These multiplication tables are used to construct right multiplication matrices that in the sequel are used to construct orthogonal designs. Using the right multiplication operator is a way to overcome the obstacle of non-associativity of the algebra. Non-associativity is an obstacle, because it is incompatible with the existence of matrix representations, that we could use directly to construct orthogonal designs. To circumvent this obstacle we use the right multiplication operator, as it seems that left multiplication is not suitable for our purposes.

First, we give an account of the classical Williamson construction for orthogonal designs [2], from the point of view of quaternions, following Baumert and Hall to be able to use it as reference in subsequent constructions.

A basis for quaternions is given by the four elements $1, i, j, k$, having the properties

$$i^2 = -1,\ j^2 = -1,\ k^2 = -1,\ ij = k,\ ji = -k,\ ik = -j,\ ki = j,\ jk = i,\ kj = -i.$$

These properties are enough to specify the full multiplication table for the four basis elements. We note that quaternion multiplication is not commutative.

To associate a 4×4 matrix to each basis element, we use the right multiplication operator on the column vector $v = \begin{bmatrix} 1 & i & j & k \end{bmatrix}^t$. Then the right multiplications $v \cdot 1$, $v \cdot i$, $v \cdot j$, $v \cdot k$, give rise to the following four 4×4 matrices respectively:

$$q_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad q_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

$$q_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, \quad q_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}.$$

Let A, B, C, D be commuting variables. Then the sum

$$Aq_1 + Bq_2 + Cq_3 + Dq_4$$

is equal to the classical Williamson array

$$H_4 = \begin{pmatrix} A & B & C & D \\ -B & A & -D & C \\ -C & D & A & -B \\ -D & -C & B & A \end{pmatrix}$$

which has the property

$$H_4 H_4^T = (A^2 + B^2 + C^2 + D^2) I_4.$$

The matrix H is the design matrix of an $OD(4; 1, 1, 1, 1)$.

The Cayley-Dickson process allows us to obtain an algebra of dimension $2n$ from an algebra of dimension n, see [4]. One limitation of this process is that restricts our method to study ODs in powers of two. By applying the Cayley-Dickson process successively to the algebras of quaternions we get octonions and sedenions [7]. Repeating the Cayley-Dickson process to the algebra of sedenions one obtains a Cayley-Dickson algebra of dimension 32 and doing the same for the latter algebra we can obtain a Cayley-Dickson algebra of order 64 [8].

3.1 Cayley-Dickson Orthogonal Designs

It is important to note that the Cayley-Dickson process essentially constructs the multiplication tables we need to model orthogonal designs. Now we describe a generic formulation of

the algebraic modelling with multiplication tables of appropriate Cayley-Dickson algebras of order n to obtain orthogonal designs of order n.

Take a Cayley-Dickson algebra of dimension n with basis $e_0 = 1, e_1, \ldots, e_{n-1}$. To associate an $n \times n$ matrix to each basis element, we use the right multiplication operator on the column vector $v = \begin{bmatrix} 1 & e_1 & \ldots & e_{n-1} \end{bmatrix}^t$. Then the n right multiplications $v \cdot e_0,\ v \cdot e_1,\ \ldots,\ v \cdot e_{n-1}$ give rise to n matrices $q_0, \ldots, q_{n-1}$ of order n. Let $A_1, \ldots, A_n$ be commuting variables. Then the sum $A = \sum_{i=0}^{n-1} A_{i+1} q_i$ is equal to an $n \times n$ matrix with the property that the diagonal elements of AA^T are all equal to $\sum_{i=1}^{n} A_i^2$, but whose other elements are not necessarily all zero.

By requiring that all elements of AA^T (except the diagonal ones) are equal to zero, we obtain a polynomial system of equations in the set of variables $\{A_1, \ldots, A_n\}$. We define this problem as the *Cayley-Dickson Orthogonal Design* (CDOD) problem.

To represent solutions, we introduce a special kind of mapping that we call *substitution mapping* or, simply, a *substitution*. Formally, a substitution from a set S_1 to a set $S_2 \supseteq S_1$ is a mapping from S_1 to S_2 which is identity almost everywhere. We use lower case Greek letters to denote them. The identity substitution is denoted by ε. The *domain* and the *range* of a substitution σ are defined, respectively, as $dom(\sigma) := \{u \mid u \in S_1, u \neq \sigma(u)\}$ and $ran(\sigma) := \cup_{u \in dom(\sigma)} \{\sigma(u)\}$. A substitution, usually, is represented as a function by a finite set of bindings of variables in its domain. For instance, a substitution σ is represented as $\{u \mapsto \sigma(u) \mid u \in dom(\sigma)\}$.

It is important to highlight that we seek solutions of CDOD's in an endomorphic form, i.e., substitutions from a set S to itself. For a CDOD of order n, this set is $\{A_1, \ldots, A_n\}$. Moreover, the domain and the range of such substitutions should be disjoint, i.e., the substitutions should be idempotent. These requirements are justified by the following:

- Mapping of variables A_i to variables A_j, for $i, j \in \{1, \ldots, n\}$, is due to the fact that we force the matrix A to be an orthogonal design and by definition the diagonal elements give rise to a quadratic form that is a sum of squares.
- In particular, if several variables map to the same variable, it is the analogue of the equating operation of the Equating-Killing Lemma for orthogonal designs for the equations that are produced by the algebraic modelling. It is clear from the context that equating variables in the polynomial system of equations, is the same as equating variables in the design matrix representation.

▶ **Theorem 4.** *Let $n = 2^m$ for some $m > 0$. Any endomorphic idempotent solution to* CDOD *of order n gives rise to an orthogonal design of order n, which we call a Cayley-Dickson orthogonal design of order n.*

Proof. Let σ be an endomorphic idempotent solution of the CDOD of order n over the set of variables $\{A_1, \ldots, A_n\}$. From σ, we associate with each A_i a number s_i as follows:

- If $A_i \in dom(\sigma)$ and $A_i \notin ran(\sigma)$, then $s_i = 0$.
- If $A_i \notin dom(\sigma)$ and $A_i \notin ran(\sigma)$, then $s_i = 1$.
- If $A_i \notin dom(\sigma)$ and $A_i \in ran(\sigma)$, then $s_i = m + 1$, where m is the number of variables that map to A_i by sigma.

These s_i's, together with the corresponding A_i, give a matrix A with the property $AA^T = (\sum_{j=1}^{k} s_j A_j^2) I_n$, for $k \leq \rho(n)$, and $\rho(n)$ is the Radon function that gives an upper bound on the

number of variables that can appear in a design. This is by definition an orthogonal design of order n and type $(s_1, \ldots, s_k)$. ◀

Now, it is important to note that the CDOD problem is instantiated for orders of power of two, since in these orders we are able to construct the multiplication tables of the respective algebras by using successively the Cayley-Dickson process on the construction of designs via quaternions of Baumert and Hall. We are interested in Cayley-Dickson orthogonal designs in orders 16, 32 and 64.

- CDOD16: An instance of the CDOD problem for order 16, consists of a polynomial system of 42 equations in 14 variables.
- CDOD32: An instance of the CDOD problem for order 32, consists of a polynomial system of 252 equations in 30 variables.
- CDOD64: An instance of the CDOD problem for order 64, consists of a polynomial system of 1182 equations in 62 variables.

We emphasize here the computational difficulty of retrieving *all* endomorphic solutions of the previous three problems. We have used Gröbner bases to verify the computations of [7] and [8], for orders 16 and 32, 64, respectively. In particular, we have computed in Magma V2.12-14 a reduced Gröbner basis (for a total degree reverse lexicographical ordering) for the polynomial systems of the CDOD16 and CDOD32 problem. For order 64 we have not managed to compute a Gröbner basis due to its enormous computational cost. Clearly, a solution of the reduced polynomial system obtained by a Gröbner basis corresponds to a solution of the original system. We formulate the CDOD problems in terms of Gröbner bases, below.

- CDODGB16: A reduced Gröbner basis of the CDOD problem for order 16, consists of a polynomial system of 21 equations in 14 variables.
- CDODGB32: A reduced Gröbner basis of the CDOD problem for order 32, consists of a polynomial system of 290 equations in 30 variables.

Gröbner bases give some insight how to locate endomorphic solutions due to the fact that binomial terms of the polynomial system could be written in a canonical form. However, this is not sufficient to compute all required solutions as there is no indication for the structure of substitution of different variables. Moreover, using this property that distills from Gröbner bases in [7] and [8], it was feasible only to compute a handful of solutions and respectively orthogonal designs.

It is clear that a specialized equation solver is needed to retrieve all endomorphic solutions for the previous five problems. Performing some post-processing on the structure of the polynomial systems we obtained for these problems, we observe that each equation consists of the *same* number of positive and negative monomial terms, and within each equation, all monomials have the same degree. This property, together with some statistics for the structure of the equations presented in Table 1, makes the CDOD problems and their Gröbner basis counterpart very suitable to be attacked by equational unification as we later explain in Sections 5 and 6.

4 Equational Unification

Unification theory [1] studies *unification problems:* sets of equations between *terms.* The latter, as usual, are constructed by a set of function symbols $\mathcal{F}$ and a (countably infinite) set of variables $\mathcal{V}$. We denote the set of terms over $\mathcal{F}$ and $\mathcal{V}$ by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. Variables are denoted by x, y, z, function symbols by f, g, and terms by s, t, r.

Our substitutions are a special case of the substitutions defined in Section 3.1, mapping variables to terms. An *application* of a substitution σ to a term t, denoted $t\sigma$, is defined as follows: If $t = x$, then $t\sigma := \sigma(x)$. If $t = f(s_1, \ldots, s_n)$, $n \geq 0$, then $t\sigma := f(s_1\sigma, \ldots, s_n\sigma)$. *Composition* of two substitutions σ and φ, written as $\sigma\varphi$, is defined as $t\sigma\varphi := (t\sigma)\varphi$ for any t.

An *equational theory*, defined by a set equational axioms $E \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathcal{T}(\mathcal{F}, \mathcal{V})$, is the least congruence relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$, that is closed under substitution application and contains E. It is denoted by $\doteq_E$. If $s \doteq_E t$, then we say that s and t are *equal modulo* E. The axioms (i.e., the elements of E) are written as $s \approx t$. For instance, $E = \{f(x, f(y, z)) \approx f(f(x, y), z),\ f(x, y) \approx f(y, x)\}$ defines the equational theory of associativity and commutativity of f.

Given an E and a set of variables $\mathcal{X}$, the substitution σ is *more general modulo* E *on* $\mathcal{X}$ than the substitution φ, written $\sigma \preccurlyeq_E^{\mathcal{X}} \varphi$, iff there exists a substitution ϑ such that $x\sigma\vartheta \doteq_E x\varphi$ for all $x \in \mathcal{X}$. The relation $\preccurlyeq_E^{\mathcal{X}}$ is a quasi-order, and the induced equivalence is denoted by $\simeq_E^{\mathcal{X}}$.

Given an E and a set of function symbols $\mathcal{F}$, an E-unification problem Γ over $\mathcal{F}$ is a finite set of equations between terms over $\mathcal{F}$ and a countable infinite set of variables $\mathcal{V}$, written as $\Gamma := \{s_1 \stackrel{?}{\doteq}_E t_1, \ldots, s_n \stackrel{?}{\doteq}_E t_n\}$. An E-unifier of Γ is a substitution σ such that $s_i\sigma \doteq_E t_i\sigma$ for all $1 \leq i \leq n$.

Let Γ be an E-unification problem over $\mathcal{F}$ and let $\mathcal{X}$ be the set of all variables that occur in Γ. A *minimal complete set of unifiers* (*mcsu,* in short) of Γ, denoted $mcsu(\Gamma)$, is the set of substitutions such that the following three conditions are satisfied:

- Correctness: Each element of $mcsu(\Gamma)$ is an E-unifier of Γ.
- Completeness: For each unifier φ of Γ there exists $\sigma \in mcsu(\Gamma)$ such that $\sigma \preccurlyeq_E^{\mathcal{X}} \varphi$.
- Minimality: For all $\sigma_1, \sigma_2 \in mcsu(\Gamma)$, if $\sigma_1 \preccurlyeq_E^{\mathcal{X}} \sigma_2$, then $\sigma_1 = \sigma_2$.

The *signature* of an equational theory E, denoted by $sig(E)$, is the set of all function symbols that appear in the axioms of E. An E-unification problem Γ over $\mathcal{F}$ is *elementary*, if $\mathcal{F} \setminus sig(E) = \emptyset$. It is a problem with *constants*, if $\mathcal{F} \setminus sig(E)$ is a set of constants. It is called a *general* problem, if $\mathcal{F} \setminus sig(E)$ may contain arbitrary function symbols.

When we are interested in E-unification problems of a special form, we talk about a *fragment* of E-unification. When solutions only of a special form are needed, then we say that a *variant* of E-unification is considered.

5 Orthogonal Designs Meet Equational Unification

In this section, we establish the connections between orthogonal designs and equational unification. In particular, we show that Cayley-Dickson orthogonal designs defined in Section 3.1 can be constructed from unifiers of certain unification problems.

Recall that, as we observed, each equation in a CDOD consists of an *equal* number of positive and negative monomial terms. Moreover, within an equation, all monomials have the *same* degree. That means that the equations have the form $A_{11} \cdots A_{1n} + \cdots + A_{m1} \cdots A_{mn} - B_{11} \cdots B_{1n} - \cdots - B_{m1} \cdots B_{mn} = 0$ with $n, m > 0$. By changing the design variables with unification variables (A with x, B with y), making the multiplication explicit, and placing negative monomials on the other side of equation, we obtain a unification problem of the form $x_{11} * \cdots * x_{1n} + \cdots + x_{m1} * \cdots * x_{mn} \stackrel{?}{\doteq}_{\mathsf{AC}(+,*)} y_{11} * \cdots * y_{1n} + \cdots + y_{m1} * \cdots * y_{mn}$, where $*$ and $+$ are associative and commutative (and the subscript $\mathsf{AC}(+, *)$ indicates this fact). We refer to the unification problem obtained from an CDOD (of order n) in this way as $\mathtt{CDOD}_{\mathcal{U}}$ (of order n). The important property, that is straightforward to see, is that there is a

direct correspondence between endomorphic solutions of unification equations in the $\mathtt{CDOD}_{\mathcal{U}}$ and those of the corresponding polynomial equations in the given CDOD.

▶ **Theorem 5.** *Let $n = 2^m$ for some $m > 0$. If there exists an endomorphic idempotent unifier for an $\mathtt{CDOD}_{\mathcal{U}}$ problem of order n, then there exists a Cayley-Dickson orthogonal design of order n.*

Proof. $\mathtt{CDOD}_{\mathcal{U}}$ of order n has an idempotent endomorphic unifier iff the corresponding CDOD of order n has an idempotent endomorphic solution. By Theorem 4, the latter implies the existence of an Cayley-Dickson orthogonal design of order n. ◀

For an endomorphic idempotent solution σ of the CDOD, the corresponding Cayley-Dickson orthogonal design is denoted by $\mathtt{CDOD}(\sigma)$.

In the theory of orthogonal designs, as we have already mentioned the Equating-Killing Lemma plays a pivotal role, as it can produce a vast number of orthogonal designs from any given one. It is natural to distinguish between orthogonal designs that can be produced or not by the Equating-Killing Lemma.

Given two ODs of the same order, D_1 and D_2, we say D_1 is more general than D_2 and write $D_1 \trianglerighteq D_2$, if there exists an OD D_3 of the same order as D_1 and D_2 such that $D_1 \simeq D_3$ and D_2 is obtained from D_3 by equating zero or more variables. Strictly more generality relation is written $D_1 \triangleright D_2$ and requires equating one or more variables to get D_3 from D_1.

▶ **Definition 6** (Basis)**.** Let $\mathcal{D}$ be a set of orthogonal designs of order n. A *basis* for $\mathcal{D}$ is a set $\mathcal{B} \subseteq \mathcal{D}$ such that for each $D \in \mathcal{D}$, there is $B \in \mathcal{B}$ such that $B \trianglerighteq D$.

A trivial basis for $\mathcal{D}$ is $\mathcal{D}$ itself. The interesting ones are *reduced bases* defined below:

▶ **Definition 7** (Reduced Basis, Base OD)**.** Let $\mathcal{B}$ be a basis of the set $\mathcal{D}$ of orthogonal designs of the same order. $\mathcal{B}$ is a *reduced basis* of $\mathcal{D}$, written $rb(\mathcal{D})$, if $\mathcal{B}$ does not contain two elements B_1, B_2 such that $B_1 \trianglerighteq B_2$. The elements of $rb(\mathcal{D})$ are called the *base orthogonal designs* for $\mathcal{D}$.

This notion of *base orthogonal designs* introduced here for the first time, exhibits a remarkable connection with unification theory.

▶ **Theorem 8.** *Consider a CDOD problem of order n and the corresponding $\mathtt{CDOD}_{\mathcal{U}}$ unification problem. Let σ be an element of the minimal complete set of endomorphic idempotent unifiers of $\mathtt{CDOD}_{\mathcal{U}}$. Assume $\mathcal{D}$ is a set of Cayley-Dickson orthogonal designs of order n (i.e, the solutions of the CDOD problem). Then $\mathtt{CDOD}(\sigma) \in \mathcal{D}$ is a base orthogonal design for $\mathcal{D}$.*

Proof. Let χ be the set of variables of $\mathtt{CDOD}_{\mathcal{U}}$ and A be the set of variables of CDOD. The theorem follows from the following fact: For two endomorphic idempotent unifiers φ_1 and φ_2 of $\mathtt{CDOD}_{\mathcal{U}}$, if $\varphi_1 \preccurlyeq^{\chi}_{\mathsf{AC}(+,*)} \varphi_2$, then $\mathtt{CDOD}(\varphi_1) \trianglerighteq \mathtt{CDOD}(\varphi_2)$. Since φ_1 and φ_2 are endomorphic, $\varphi_1 \preccurlyeq^{\chi}_{\mathsf{AC}(+,*)} \varphi_2$ means that for some ϑ, $x\varphi_1\vartheta = x\varphi_2$ for all $x \in \chi$. Hence, ϑ is also endomorphic on χ and can be decomposed into $\vartheta_1\vartheta_2$, where ϑ_1 is a permutation (a bijective mapping from $dom(\vartheta)$ to $dom(\vartheta)$), and ϑ_2 is an arbitrary endomorphic substitution. Then from $\mathtt{CDOD}(\varphi_1)$ we first can obtain an OD D by renaming variables that correspond to ϑ_1. It gives $\mathtt{CDOD}(\varphi_1) \simeq D$. Afterwards, from D we can perform variable equating according to ϑ_2, which will give $\mathtt{CDOD}(\varphi_2)$. By the definition of $\trianglerighteq$, we get $\mathtt{CDOD}(\varphi_1) \trianglerighteq \mathtt{CDOD}(\varphi_2)$. ◀

The connections between orthogonal designs and unification theory presented in this section are essential for translating CDOD problems to unification problems, and in addition provide some concrete guidelines on how to efficiently perform a systematic solving of the respective polynomial systems.

6 Solving Unification Problems

Our unification problem Γ contains only equations in the flattened form $x_1^1 * \cdots * x_n^1 + \cdots + x_1^m * \cdots * x_n^m \stackrel{?}{=}_{\mathsf{AC}(+,*)} y_1^1 * \cdots * y_n^1 + \cdots + y_1^m * \cdots * y_n^m$ for some $n, m > 0$, where $+$ and $*$ are the AC symbols. We call it a *balanced* fragment of AC-unification. We are looking for AC-unifiers of Γ that map variables of Γ to variables of Γ, i.e., both domain and range of unifiers should be subsets of $var(\Gamma)$. We call such variants *endomorphic*. Hence, the problem we would like to solve is an *endomorphic variant of a balanced fragment of the elementary AC-unification*. For brevity, we refer to it as an $\mathrm{AC}_{\mathrm{EB}}$-unification problem.

Note that this problem always has a unifier: Just map all variables to one of them, and it will be a solution. What we are looking for is the minimal complete set of unifiers.

AC-unification problems are solved by reducing them to systems of linear Diophantine equations, see, e.g., [3, 14]. However, it is pretty easy to formulate a direct algorithm that computes a complete set of unifiers for $\mathrm{AC}_{\mathrm{EB}}$-unification problems. In fact, as we will see, the four rules below are sufficient to construct it. The rules transform systems (pairs $\Gamma; \sigma$ of an unification problem and a substitution) into systems. The symbol $\cup$ stands for disjoint union. The subscript $\mathsf{AC}(+,*)$ is omitted, as well as the symbol $*$.

T: **Trivial**

$$\{x \stackrel{?}{=} x\} \cup \Gamma';\ \sigma \Longrightarrow \Gamma';\ \sigma.$$

D-sum: **Decomposition for Sums**

$$\{s_1 + \cdots + s_n \stackrel{?}{=} t_1 + \cdots + t_n\} \cup \Gamma';\ \sigma \Longrightarrow \{s_1 \stackrel{?}{=} \pi(t_1), \ldots, s_n \stackrel{?}{=} \pi(t_n)\} \cup \Gamma';\ \sigma,$$

where $n > 1$ and π is a permutation of the multiset $\{t_1, \ldots, t_n\}$.

D-prod: **Decomposition for Products**

$$\{x_1 \cdots x_n \stackrel{?}{=} y_1 \cdots y_n\} \cup \Gamma';\ \sigma \Longrightarrow \{x_1 \stackrel{?}{=} \pi(y_1), \ldots, x_n \stackrel{?}{=} \pi(y_n)\} \cup \Gamma';\ \sigma,$$

where $n > 1$ and π is a permutation of the multiset $\{t_1, \ldots, t_n\}$.

S: **Solve**

$$\{x \stackrel{?}{=} y\} \cup \Gamma';\ \sigma \Longrightarrow \Gamma'\{x \mapsto y\};\ \sigma\{x \mapsto y\}, \qquad \text{where } x \neq y.$$

We call a system $\Gamma; \sigma$ a *balanced system*, if Γ is a balanced AC-unification problem. By inspecting the rules, it is easy to see that the rules transform balanced systems into balanced systems. Note that any balanced system $\Gamma; \sigma$, where $\Gamma \neq \emptyset$, can be transformed, and each selected equation can be transformed by only one rule.

To solve an unification problem Γ, we create the initial system $\Gamma; \varepsilon$ and apply the rules exhaustively. Let **BF** denote this algorithm, to indicate that it is a brute force approach, i.e., **BF** := (T | D-sum | D-prod | S)*, where | stands for choice and * for iteration. The terminal systems have the form $\emptyset; \sigma$. We say in this case that the algorithm computes σ. Given a balanced Γ, the set of all substitutions computed by **BF** is denoted by $\Sigma_{\mathbf{BF}}(\Gamma)$. This set is finite, because there can be finitely many terminal systems (since rules that produce all possible permutations lead to finite branching).

▶ **Theorem 9.** *Given a balanced AC-unification problem Γ, the algorithm **BF** terminates and computes $\Sigma_{\mathbf{BF}}(\Gamma)$, which is a complete set of endomorphic idempotent AC-unifiers of Γ.*

Proof. (Sketch) To prove termination, we first define the size of an equation as the number of symbol occurrences in it (including the $*$ that is omitted in the rules). Next, we associate to each AC-unification problem its measure: the multiset of sizes of equations in it. Then we

can see that each rule strictly decreases this measure. For the rules T and S it is obvious. For the other two rules it follows from the condition $n > 1$, which implies that the resulting set of equations reduces the number of occurrences of $+$ or $*$, while the rest does not increase. These facts, together with the observation that the number of branching alternatives the rules produce is finite, imply termination.

The S rule guarantees that the computed substitutions are endomorphic and idempotent. Each rule preserves the set of unifiers for the problems it transforms. Hence the computed substitutions are endomorphic idempotent unifiers. Completeness is implied by the fact that the permutations in the decomposition rules generate all possible branchings in the search tree. ◀

The set $\Sigma_{\mathbf{BF}}(\Gamma)$ is not minimal, in general. This is not surprising, since $\mathrm{AC}_{\mathrm{EB}}$-unification is, in fact, variadic commutative (aka orderless) unification [9]. No algorithm is known that would directly compute minimal complete set of unifiers for commutative unification problems. There is an additional minimization step required.

Our main challenge, however, was related to the size of the problem. The unification problems contain hundreds of equations and the brute-force approach of **BF** usually is not feasible. We need to keep the alternatives as small as possible. For this purpose, we elaborated several heuristics. Two of them concern equation selection, and two more unification problem simplification:

Sel1: For transformation, select an equation with the minimal number of arguments. For instance, if the unification problem is $\{x_1x_2 + y_3x_3 \stackrel{?}{\doteq} x_3y_3 + y_2y_1, x_1 \stackrel{?}{\doteq} y_1\}$, the equation $x_1 \stackrel{?}{\doteq} y_1$ will be selected and transformed by the rule Solve.

Sel2: In the decomposition rules, permute that side of the selected equation that generates fewer permutations (i.e., the side that has more repeated arguments). It reduces the branching factor, but completeness is not violated, since equality is symmetric.

Simp1: Given an ordering on variables that is extended lexicographically to products and sums, rearrange unordered subterms in equations in the ordered form. For instance, if $x_1 > x_2 > x_3 > y_1 > y_2 > y_3$, then the equation $x_1x_2 + y_3x_3 \stackrel{?}{\doteq} x_3y_3 + y_2y_1$ would be transformed in one step to, e.g., $x_1x_2 + x_3y_3 \stackrel{?}{\doteq} x_3y_3 + y_2y_1$ and in two steps to $x_1x_2 + x_3y_3 \stackrel{?}{\doteq} x_3y_3 + y_1y_2$.

Simp2: Remove all common arguments from both sides of equations. This is a well-known technique used in AC-unification. It reduces, for instance, the equation $x_1x_2 + x_3y_3 \stackrel{?}{\doteq} x_3y_3 + y_1y_2$ in one step to $x_1x_2 \stackrel{?}{\doteq} y_1y_2$. Two applications of this strategy would reduce the equation $x_1x_2y_1 \stackrel{?}{\doteq} y_1x_2y_2$ to $x_1 \stackrel{?}{\doteq} y_2$.

Let T-s and S-s be the variations of the T and S rules, respectively, where equations are selected according to Sel1. Similarly, D-sum-s and D-prod-s stand for the variants of D-sum and D-prod rules, where the equation is selected according to Sel1, and the permutation side is selected according to Sel2. Simp1 should be used in combination with Simp2 to detect common arguments in the sides of equations. Then we define the refined algorithm **Ref** with the following strategy ($\circ$ stands for composition, | for choice, * for iteration):

$$\mathbf{Ref} := \big((\mathsf{Simp1} \mid \mathsf{Simp2})^* \circ (\mathsf{T\text{-}s} \mid \mathsf{S\text{-}s} \mid \mathsf{D\text{-}sum\text{-}s} \mid \mathsf{D\text{-}prod\text{-}s})\big)^*.$$

In words, it means that **Ref** works with a set of systems, selects one of them nondeterministically, normalizes it with respect to the Simp1 and Simp2, transforms the obtained system into new ones with one of the rules T-s, S-s, D-sum-s, or D-prod-s, and iterates.

Since unification problems are sets, simplification steps may decrease the number of equations, when several equations simplify to the same one. It is not hard to see that the

selection and simplification heuristics affect neither soundness nor completeness. Therefore, based on Theorem 9 we have that $\Sigma_{\mathsf{Ref}}(\Gamma)$ is a complete set of endomorphic idempotent unifiers of Γ.

As it turns out, Simp2 plays an important role in reducing the number of computed unifiers. For instance, for an unification problem Γ originated from `CDODGB16`, $\Sigma_{\mathsf{Ref}}(\Gamma)$ contains 7 unifiers. For a Γ coming from `CDODGB32`, this number is 33. If we skipped the Simp2 step in **Ref**, then we would get 45 unifiers for `CDODGB16`, and 1574 for `CDODGB32`. Similarly, for an unification problem Γ originated from `CDOD16`, $\Sigma_{\mathsf{Ref}}(\Gamma)$ contains 65 unifiers. For a Γ coming from `CDOD32`, this number is 6935. If we again skip the Simp2 step in **Ref**, then we get 264 unifiers for `CDOD16`.

The set computed by **Ref** is complete but not minimal. A minimal and complete algorithm **ACEB** for $\mathrm{AC}_{\mathrm{EB}}$-unification problems can be formulated as

$$\mathsf{ACEB}(\Gamma) := \mathit{minimize}\big(\Sigma_{\mathsf{Ref}}(\Gamma)\big),$$

where *minimize* is a function that minimizes a set of substitutions. Therefore, we have the following theorem:

▶ **Theorem 10.** $\mathit{ACEB}(\Gamma) = \mathit{mcsu}(\Gamma)$.

For efficiency reasons, it makes sense to have an incremental version of the algorithm **ACEB**: Instead of working with the entire set of equations at once, we split this set into smaller subsets of some fixed size size. After **ACEB** computes an mcsu $\mathcal{U}$ of one such subset, we generate all possible instances of the next subset with respect to the unifiers in $\mathcal{U}$, and proceed further in a similar way for each new set. Such early minimization efforts reduce the number of redundant potential solutions. This method is sensitive to the choice of size. It should be not too small not to trigger frequent calls of the expensive *minimize* function, and not too big not to postpone minimization too much. As experiments showed, a good strategy for the unification problems originated from the original polynomials is, for instance, to set size close to the number of equations of the smallest size. For instance, in `CDOD32`, the polynomials of the smallest size are those that contain 4 monomials, each of degree 2. There are 42 such polynomials (out of 252) there. Setting size to 42 led to the fastest computation of the result. However, for equations coming from the polynomials in Gröbner bases, we could not observe such a pattern.

Now we give the elements of $\mathsf{ACEB}(\Gamma)$ for unification problems Γ that originate from `CDOD16`, `CDODGB16`, `CDOD32`, `CDODGB32` and `CDOD64` problems:

`CDOD16` and `CDODGB16`:

$$\sigma_1^{16} = \{x_{10} \to x_2,\, x_{11} \to x_3,\, x_{12} \to x_4,\, x_{13} \to x_5,\, x_{14} \to x_6,\, x_{15} \to x_7,\, x_{16} \to x_8\}$$

$$\begin{aligned}\sigma_2^{16} = \{&x_2 \to x_8,\, x_3 \to x_8,\, x_4 \to x_8,\, x_5 \to x_8,\, x_6 \to x_8,\, x_7 \to x_8,\, x_{10} \to x_{16},\\ &x_{11} \to x_{16},\, x_{12} \to x_{16},\, x_{13} \to x_{16},\, x_{14} \to x_{16},\, x_{15} \to x_{16}\}\end{aligned}$$

`CDOD32` and `CDODGB32`:

$$\begin{aligned}\sigma_1^{32} = \{&x_2 \to x_8,\, x_3 \to x_8,\, x_4 \to x_8,\, x_5 \to x_8,\, x_6 \to x_8,\, x_7 \to x_8,\, x_{10} \to x_{32},\\ &x_{11} \to x_{32},\, x_{12} \to x_{32},\, x_{13} \to x_{32},\, x_{14} \to x_{32},\, x_{15} \to x_{32},\, x_{16} \to x_{32},\\ &x_{18} \to x_8,\, x_{19} \to x_8,\, x_{20} \to x_8,\, x_{21} \to x_8,\, x_{22} \to x_8,\, x_{23} \to x_8,\, x_{24} \to x_8,\\ &x_{25} \to x_9,\, x_{26} \to x_{32},\, x_{27} \to x_{32},\, x_{28} \to x_{32},\, x_{29} \to x_{32},\, x_{30} \to x_{32},\\ &x_{31} \to x_{32}\}\end{aligned}$$

$$\sigma_2^{32} = \{x_2 \to x_{26},\, x_{10} \to x_{26},\, x_{11} \to x_3,\, x_{12} \to x_4,\, x_{13} \to x_5,\, x_{14} \to x_6,\, x_{15} \to x_7,$$

$x_{16} \to x_8$, $x_{18} \to x_{26}$, $x_{19} \to x_3$, $x_{20} \to x_4$, $x_{21} \to x_5$, $x_{22} \to x_6$, $x_{23} \to x_7$,
$x_{24} \to x_8$, $x_{25} \to x_9$, $x_{27} \to x_3$, $x_{28} \to x_4$, $x_{29} \to x_5$, $x_{30} \to x_6$, $x_{31} \to x_7$,
$x_{32} \to x_8\}$

$\sigma_3^{32} = \{x_2 \to x_9$, $x_3 \to x_9$, $x_4 \to x_9$, $x_5 \to x_9$, $x_6 \to x_9$, $x_7 \to x_9$, $x_8 \to x_9$,
$x_{10} \to x_9$, $x_{11} \to x_9$, $x_{12} \to x_9$, $x_{13} \to x_9$, $x_{14} \to x_9$, $x_{15} \to x_9$, $x_{16} \to x_9$,
$x_{18} \to x_{32}$, $x_{19} \to x_{32}$, $x_{20} \to x_{32}$, $x_{21} \to x_{32}$, $x_{22} \to x_{32}$, $x_{23} \to x_{32}$,
$x_{24} \to x_{32}$, $x_{25} \to x_{32}$, $x_{26} \to x_{32}$, $x_{27} \to x_{32}$, $x_{28} \to x_{32}$, $x_{29} \to x_{32}$,
$x_{30} \to x_{32}$, $x_{31} \to x_{32}\}$

CDOD64:

$\sigma_1^{64} = \{x_2 \to x_9$, $x_3 \to x_9$, $x_4 \to x_9$, $x_5 \to x_9$, $x_6 \to x_9$, $x_7 \to x_9$, $x_8 \to x_9$,
$x_{10} \to x_9$, $x_{11} \to x_9$, $x_{12} \to x_9$, $x_{13} \to x_9$, $x_{14} \to x_9$, $x_{15} \to x_9$, $x_{16} \to x_9$,
$x_{17} \to x_9$, $x_{18} \to x_9$, $x_{19} \to x_9$, $x_{20} \to x_9$, $x_{21} \to x_9$, $x_{22} \to x_9$, $x_{23} \to x_9$,
$x_{24} \to x_9$, $x_{25} \to x_9$, $x_{26} \to x_9$, $x_{27} \to x_9$, $x_{28} \to x_9$, $x_{29} \to x_9$, $x_{30} \to x_9$,
$x_{31} \to x_9$, $x_{32} \to x_9$, $x_{34} \to x_{64}$, $x_{35} \to x_{64}$, $x_{36} \to x_{64}$, $x_{37} \to x_{64}$,
$x_{38} \to x_{64}$, $x_{39} \to x_{64}$, $x_{40} \to x_{64}$, $x_{41} \to x_{64}$, $x_{42} \to x_{64}$, $x_{43} \to x_{64}$,
$x_{44} \to x_{64}$, $x_{45} \to x_{64}$, $x_{46} \to x_{64}$, $x_{47} \to x_{64}$, $x_{48} \to x_{64}$, $x_{49} \to x_{64}$,
$x_{50} \to x_{64}$, $x_{51} \to x_{64}$, $x_{52} \to x_{64}$, $x_{53} \to x_{64}$, $x_{54} \to x_{64}$, $x_{55} \to x_{64}$,
$x_{56} \to x_{64}$, $x_{57} \to x_{64}$, $x_{58} \to x_{64}$, $x_{59} \to x_{64}$, $x_{60} \to x_{64}$, $x_{61} \to x_{64}$,
$x_{62} \to x_{64}$, $x_{63} \to x_{64}\}$

$\sigma_2^{64} = \{x_2 \to x_8$, $x_3 \to x_8$, $x_4 \to x_8$, $x_5 \to x_8$, $x_6 \to x_8$, $x_7 \to x_8$, $x_{10} \to x_{64}$,
$x_{11} \to x_{64}$, $x_{12} \to x_{64}$, $x_{13} \to x_{64}$, $x_{14} \to x_{64}$, $x_{15} \to x_{64}$, $x_{16} \to x_{64}$,
$x_{17} \to x_{49}$, $x_{18} \to x_8$, $x_{19} \to x_8$, $x_{20} \to x_8$, $x_{21} \to x_8$, $x_{22} \to x_8$, $x_{23} \to x_8$,
$x_{24} \to x_8$, $x_{25} \to x_9$, $x_{26} \to x_{64}$, $x_{27} \to x_{64}$, $x_{28} \to x_{64}$, $x_{29} \to x_{64}$,
$x_{30} \to x_{64}$, $x_{31} \to x_{64}$, $x_{32} \to x_{64}$, $x_{34} \to x_8, x_{35} \to x_8$, $x_{36} \to x_8$,
$x_{37} \to x_8$, $x_{38} \to x_8$, $x_{39} \to x_8$, $x_{40} \to x_8$, $x_{41} \to x_9$, $x_{42} \to x_{64}$, $x_{43} \to x_{64}$,
$x_{44} \to x_{64}$, $x_{45} \to x_{64}$, $x_{46} \to x_{64}$, $x_{47} \to x_{64}$, $x_{48} \to x_{64}$, $x_{50} \to x_8$,
$x_{51} \to x_8$, $x_{52} \to x_8$, $x_{53} \to x_8$, $x_{54} \to x_8$, $x_{55} \to x_8$, $x_{56} \to x_8$, $x_{57} \to x_9$,
$x_{58} \to x_{64}$, $x_{59} \to x_{64}$, $x_{60} \to x_{64}$, $x_{61} \to x_{64}, x_{62} \to x_{64}$, $x_{63} \to x_{64}\}$

$\sigma_3^{64} = \{x_2 \to x_{58}$, $x_3 \to x_{59}$, $x_4 \to x_{60}$, $x_5 \to x_{61}$, $x_6 \to x_{62}$, $x_{10} \to x_{58}$, $x_{11} \to x_{59}$,
$x_{12} \to x_{60}$, $x_{13} \to x_{61}$, $x_{14} \to x_{62}$, $x_{15} \to x_7$, $x_{16} \to x_8$, $x_{17} \to x_{49}$, $x_{18} \to x_{58}$,
$x_{19} \to x_{59}$, $x_{20} \to x_{60}$, $x_{21} \to x_{61}$, $x_{22} \to x_{62}$, $x_{23} \to x_7$, $x_{24} \to x_8$, $x_{25} \to x_9$,
$x_{26} \to x_{58}$, $x_{27} \to x_{59}$, $x_{28} \to x_{60}$, $x_{29} \to x_{61}$, $x_{30} \to x_{62}$, $x_{31} \to x_7$, $x_{32} \to x_8$,
$x_{34} \to x_{58}$, $x_{35} \to x_{59}$, $x_{36} \to x_{60}$, $x_{37} \to x_{61}$, $x_{38} \to x_{62}$, $x_{39} \to x_7$, $x_{40} \to x_8$,
$x_{41} \to x_9$, $x_{42} \to x_{58}$, $x_{43} \to x_{59}$, $x_{44} \to x_{60}$, $x_{45} \to x_{61}$, $x_{46} \to x_{62}$, $x_{47} \to x_7$,
$x_{48} \to x_8$, $x_{50} \to x_{58}$, $x_{51} \to x_{59}$, $x_{52} \to x_{60}$, $x_{53} \to x_{61}$, $x_{54} \to x_{62}$, $x_{55} \to x_7$,
$x_{56} \to x_8$, $x_{57} \to x_9$, $x_{63} \to x_7$, $x_{64} \to x_8\}$

$\sigma_4^{64} = \{x_2 \to x_9$, $x_3 \to x_9$, $x_4 \to x_9$, $x_5 \to x_9$, $x_6 \to x_9$, $x_7 \to x_9$, $x_8 \to x_9$, $x_{10} \to x_9$,
$x_{11} \to x_9$, $x_{12} \to x_9$, $x_{13} \to x_9, x_{14} \to x_9$, $x_{15} \to x_9$, $x_{16} \to x_9$, $x_{17} \to x_{49}$,
$x_{18} \to x_{64}$, $x_{19} \to x_{64}$, $x_{20} \to x_{64}$, $x_{21} \to x_{64}$, $x_{22} \to x_{64}$, $x_{23} \to x_{64}$,
$x_{24} \to x_{64}$, $x_{25} \to x_{64}$, $x_{26} \to x_{64}$, $x_{27} \to x_{64}$, $x_{28} \to x_{64}$, $x_{29} \to x_{64}$,
$x_{30} \to x_{64}$, $x_{31} \to x_{64}$, $x_{32} \to x_{64}$, $x_{34} \to x_9$, $x_{35} \to x_9$, $x_{36} \to x_9$, $x_{37} \to x_9$,
$x_{38} \to x_9$, $x_{39} \to x_9$, $x_{40} \to x_9$, $x_{41} \to x_9$, $x_{42} \to x_9$, $x_{43} \to x_9$, $x_{44} \to x_9$,
$x_{45} \to x_9$, $x_{46} \to x_9$, $x_{47} \to x_9$, $x_{48} \to x_9$, $x_{50} \to x_{64}$, $x_{51} \to x_{64}$, $x_{52} \to x_{64}$,

$$x_{53} \to x_{64},\ x_{54} \to x_{64},\ x_{55} \to x_{64},\ x_{56} \to x_{64},\ x_{57} \to x_{64},\ x_{58} \to x_{64},$$
$$x_{59} \to x_{64},\ x_{60} \to x_{64},\ x_{61} \to x_{64},\ x_{62} \to x_{64},\ x_{63} \to x_{64}\}$$

We remark that as expected the elements of the **ACEB**(Γ) for the unification problems Γ that originate from `CDOD16` and `CDODGB16`, are the same. This also applies for `CDOD32` and `CDODGB32`.

7 New Cayley-Dickson Orthogonal Designs via Equational Unification

In this section, we translate back from unifiers to solutions of the polynomial systems that give rise to Cayley-Dickson orthogonal designs and list their types. As noted before the elements of **ACEB**(Γ) correspond to base orthogonal designs from Corollary 8, which implies that the designs we list below are sufficient to give all Cayley-Dickson orthogonal designs for orders 16, 32 and 64. Therefore, we provide a complete solution to the `CDOD` problem for these orders.

1. For order 16 we obtain the following *two* base Cayley-Dickson orthogonal designs:
   - From σ_1^{16}: $OD(16; 1, 1, 2, 2, 2, 2, 2, 2, 2)$.
   - From σ_2^{16}: $OD(16; 1, 1, 7, 7)$.
2. For order 32 we obtain the following *three* base Cayley-Dickson orthogonal designs:
   - From σ_1^{32}: $OD(32; 1, 1, 2, 14, 14)$.
   - From σ_2^{32}: $OD(32; 1, 1, 2, 4, 4, 4, 4, 4, 4, 4)$.
   - From σ_3^{32}: $OD(32; 1, 1, 15, 15)$.
3. For order 64 we obtain the following *four* base Cayley-Dickson orthogonal designs:
   - From σ_1^{64}: $OD(64; 1, 1, 31, 31)$.
   - From σ_2^{64}: $OD(64; 1, 1, 2, 4, 28, 28)$.
   - From σ_3^{64}: $OD(64; 1, 1, 2, 4, 8, 8, 8, 8, 8, 8, 8)$.
   - From σ_4^{64}: $OD(64; 1, 1, 2, 30, 30)$.

It is important to note here that from the previous list of orthogonal designs, some Cayley-Dickson orthogonal designs appear here for the *first time*. In particular, the $OD(32; 1, 1, 2, 14, 14)$ and $OD(64; 1, 1, 2, 30, 30)$, $OD(64; 1, 1, 2, 4, 28, 28)$ have not been reported in [7] and [8], respectively.

However, these types of orthogonal designs are not new in the literature of orthogonal designs, as they can be obtained by other methods. In particular, the existence of $OD(32; 1, 1, 2, 14, 14)$ is attributed to a result of Robinson (p. 358, Corollary D.2., [6]) which states that all orthogonal designs of type $(1, 1, a, b, c)$, $a+b+c = 2^t - 2$ exist in order $2^t, t \geq 3$, for $a = 2$, $b = 14$, $c = 14$ and $t = 5$. Again from Robinson's result the $OD(64; 1, 1, 2, 30, 30)$ is known for $a = 2$, $b = 30$, $c = 30$ and $t = 6$. Finally by applying the Doubling Lemma (c.f. Lemma 2) to $OD(32; 1, 1, 2, 14, 14)$ we can get $OD(64; 1, 1, 2, 4, 28, 28)$.

From the previous discussion three patterns for the orthogonal designs that are modelled by Cayley-Dickson algebras and obtained via equational unification are visible.

- The four variable designs are of the form $OD(2^n; 1, 1, 2^{n-1} - 1, 2^{n-1} - 1)$, for orders 2^n where $n = 4, 5, 6$. These types of orthogonal designs can also be obtained via simple Paley matrices [6].
- The five variable designs are of the form $OD(2^n; 1, 1, a, b, c)$ where $a = 2$, $b = 2^n - 2$, $c = 2^n - 2$ for $n = 5, 6$. As we already noted these types of orthogonal designs can be obtained from Robinson's results.

- It is clear that there is an analogy between the Cayley-Dickson process and the Doubling Lemma. In particular, by applying the doubling lemma to the nine variable base orthogonal design in order 16 we obtain the ten variable base orthogonal design in order 32. Repeating the process to the latter design, we obtain the eleven variable base orthogonal design in order 64. In addition, as we have shown earlier the six variable design in order 64 can also obtained by doubling of the five variable design in order 32.

Moreover, we would like to explicitly state that these designs are new with respect to the algebraic modelling of Cayley-Dickson algebras (in the class of Cayley-Dickson orthogonal designs), however the corresponding types of ODs have been reported in the literature also with other techniques. To make our contribution in this section more precise, we can say the following:

1. It was not known before that most of the ODs we found belong also to the class of Cayley-Dickson orthogonal designs.
2. Our approach not only reports the ODs, but also constructs the corresponding design matrices. In this way, we *always* give a *constructive* solution to the problem. It is not always the case with the other approaches. In some cases, there are semi-constructive techniques (doubling method), but in some other, there is only the existential, non-constructive method (Robinson's Lemma). (The doubling method is semi-constructive in the sense that one needs to know the design matrix of the initial OD in order to build design matrices of the ODs the doubling method gives.)
3. The design matrices are of interest for the applications of ODs, since in that case it is not enough to know that the design type exists. For example, in weighing experiments you need the design matrix to perform the actual experiment.
4. The fact that the class of Cayley-Dickson orthogonal designs contains the previous types of orthogonal designs is of interest also to the asymptotic existence of orthogonal designs [6] and will be studied further in future work.

8 Conclusion

In this paper, we presented a algebraic framework for modelling orthogonal designs in order of powers of two via Cayley-Dickson algebras of same orders. This framework gives rise to a polynomial system of equations that is unfeasible to be tackled with traditional search algorithms, as the order increases. We exhibited that the structural properties of this algebraic framework can be written in terms of unification theory by establishing important connections between orthogonal designs and unifiers. These connections enabled the development of unification algorithms that can solve the problems arising from the algebraic modelling of orthogonal designs and find solutions that were not known before with this algebraic modelling of Cayley-Dickson algebras.

9 Acknowledgements

The first author is supported by an NSERC Discovery grant. The second author has been supported by the Austrian Science Fund (FWF) under the project SToUT (P 24087-N18). The third author has been funded in part by the Austrian COMET Program from the Austrian Research Promotion Agency (FFG).

References

1 F. Baader and W. Snyder. Unification theory. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 445–532. Elsevier and MIT Press, 2001.

2 L. D. Baumert and J. M. Hall. Hadamard matrices of the Williamson type. *Math. Comput.*, 19:442–447, 1965.

3 A. Boudet, E. Contejean, and H. Devie. A new AC unification algorithm with an algorithm for solving systems of Diophantine equations. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*, pages 289–299. IEEE Computer Society, 1990.

4 G. M. Dixon. *Division Algebras: Octonions, Quaternions, Complex Numbers and the Algebraic Design of Physics*, volume 290 of *Mathematics and its Applications*. Kluwer Academic Publishers Group, Dordrecht, 1994.

5 A. V. Geramita, J. M. Geramita, and J. S. Wallis. Orthogonal designs. *Linear and Multilinear Algebra*, 3:281–306, 1976.

6 A. V. Geramita and J. Seberry. *Orthogonal Designs. Quadratic Forms and Hadamard Matrices*, volume 45 of *Lecture Notes in Pure and Applied Mathematics*. Marcel Dekker, Inc., New York, NY, 1979.

7 I. S. Kotsireas and C. Koukouvinos. Orthogonal designs via computational algebra. *J. Combin. Designs*, 14:351–362, 2006.

8 I. S. Kotsireas and C. Koukouvinos. Orthogonal designs of order 32 and 64 via computational algebra. *Australasian Journal of Combinatorics*, 39:39–48, 2007.

9 T. Kutsia. *Solving and Proving in Equational Theories with Sequence Variables and Flexible Arity Symbols*. PhD thesis, Johannes Kepler University, Linz, Austria, 2002.

10 F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, The Netherlands, Amsterdam, 1997.

11 R. L. Plackett and J. Burman. The design of optimum multifactorial experiments. *Biometrika*, 33:305–325, 1946.

12 E. Posner. Combinatorial structures in planetary reconnaissance. In H. Mann, editor, *Proceedings of the 1968 Symposium on Error Correcting Codes*, pages 15–46. Wiley, New York, 1968.

13 J. Seberry and R. Craigen. Orthogonal designs. In C. Colbourn and J. Dinitz, editors, *The CRC Handbook of Combinatorial Designs*, pages 400–406. CRC Press, Boca Raton, Fla., 1996.

14 M. E. Stickel. A unification algorithm for associative-commutative functions. *J. ACM*, 28(3):423–434, 1981.

15 V. Taroch, H. Jafarkhani, and A. R. Calderbank. Space-time block codes from orthogonal designs. *EEE Trans. Inf. Theory*, 45:1456–1467, 1999.

16 J. Van Lint. Coding, decoding and combinatorics. In R. Wilson, editor, *Applications of Combinatorics*. Shiva, Cheshire, 1982.

17 R. Yarlagadda and J. Hershey. *Hadamard Matrix Analysis and Synthesis: With Applications to Communications and Signal/Image Processing*. Kluwer Acad. Pub., Boston, 1997.

18 Y. Zhao, Y. Wang, and J. Seberry. On amicable orthogonal designs of order 8. *Australasian Journal of Combinatorics*, 34:321–329, 2006.

A Polynomials in CDOD

In this section we give figures that characterize polynomials in CDOD or orders 16, 32, and 64. The word *length* refers to the number of monomials in a polynomial. Half of them is positive and the other half is negative. *degree* stands for the degree that each monomial has. In the column #, the number of polynomials are shown.

Problem	Length	Degree	# polynomials
CDOD16	4	2	42
			Total: 42
CDOD32	4	2	84
	8	2	42
	12	2	126
			Total: 252
CDOD64	4	2	252
	8	2	84
	12	2	168
	16	2	42
	20	2	168
	24	2	126
	28	2	342
			Total: 1182
CDOD16GB	2	2	21
			Total: 21
CDOD32GB	2	2	105
	4	2	21
	4	3	137
	4	4	27
			Total: 290

Table 1 Statistics of the equation structure

For instance, the first row for CDOD32 indicates that there are 84 polynomials with the length 4, where all monomials have degree 2. An example of such a polynomial is $-A_{13}A_{24} + A_4A_{25} + A_8A_{29} - A_9A_{20}$.

The longest polynomials can be found in the CDOD64 problem: There are 342 there with the length 28. For instance,

$$A_6A_{48} - A_2A_{44} + A_{28}A_{50} + A_{21}A_{63} + A_{19}A_{57} - A_{16}A_{38} - A_{10}A_{36} - A_{31}A_{53} + A_5A_{47} + A_{22}A_{64} - A_{17}A_{59} - A_{15}A_{37} + A_{12}A_{34} - A_{26}A_{52} - A_{32}A_{54} + A_4A_{42} + A_{29}A_{55} + A_{30}A_{56} + A_{27}A_{49} - A_{25}A_{51} - A_{23}A_{61} - A_{24}A_{62} + A_{20}A_{58} - A_{18}A_{60} + A_{13}A_{39} + A_{14}A_{40} - A_7A_{45} - A_8A_{46}$$

is such a polynomial.

B Performance Statistics

The unification algorithms have been implemented in Mathematica 9.0 and their performance has been measured on a Linux Laptop, Intel Core i7 CPU M 640, 2.80GHz × 4, 3.7 GiB memory.

The numbers in the columns for the algorithm stand for the number of computed unifiers and the time spent on it. For instance, 264/1.46 means that the **Ref** without Simp2 computed 264 unifiers in 1.46 seconds.

Problem	# polynomials	**Ref** without Simp2	**Ref**	**ACEB**
CDOD16	42	264 / 1.46	65 / 0.78	2 / 0.91
CDOD32	252		6935 / 188.65	3/ 53.32
CDOD64	1182			4 / 7368.29
CDOD16GB	21	45 / 0.25	7/ 0.07	2 / 0.06
CDOD32GB	290	1574 / 202.90	33 / 1.94	3 / 1.60

Table 2 Performance Statistics