Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at
TECHNISCHE UNIVERSITÄT WIEN
The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

Algebraische Methoden in der statistischen Versuchsplanung

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Mathematik

eingereicht von

Bernhard Garn, BSc
Matrikelnummer 0625793

an der Fakultät für Mathematik und Geoinformation

der Technischen Universität Wien

Betreuung: Dr. Dimitrios E. Simos

Wien, 13. Februar 2019

Bernhard Garn	Dimitrios E. Simos

TECHNISCHE UNIVERSITÄT WIEN

Algebraic Methods for Experimental Design Theory

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Technical Mathematics

by

Bernhard Garn, BSc
Registration Number 0625793

to the Faculty of Mathematics and Geoinformation

at the Vienna University of Technology

Advisor: Dr. Dimitrios E. Simos

Vienna, 13th February, 2019

Bernhard Garn	Dimitrios E. Simos

Erklärung zur Verfassung der Arbeit

Bernhard Garn, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. Februar 2019

Bernhard Garn

Acknowledgements

I gratefully acknowledge and deeply thank my advisor Dr. Simos. I gratefully acknowledge and deeply thank my parents for their support. I gratefully acknowledge and deeply thank my colleagues from university, friends and my girlfriend. I gratefully acknowledge and deeply thank all those scientific researchers (including, but not limited to, mathematicians) that came before me and I wish the best of luck to all of those who will come after me. *αιὲν ἀριστεύειν*.

Kurzfassung

Die Integration von algebraischen Methoden in die Statistik in den frühen und mittleren 1990er Jahren [33, 105] hat beide Forschungsgebiete von den entstandenen Synergien profitieren lassen [4, 104]. Diese Arbeit beschäftigt sich mit kombinatorischen Designs, welche im relativ neuen Bereich des kombinatorischen Testen (KT) für Software als Teilgebiet der statistischen Versuchsplanung verwendet werden [81]. Der Begriff der "Abdeckung", der als eine Verallgemeinerung des bekannten λ-fachen Auftretens von t-Tupeln in orthogonalen Arrays angesehen werden kann, steht an zentraler Stelle in dem KT und findet sich auch in den definierenden Eigenschaften der in diesem Bereich betrachteten Strukturen wieder. Zu den betrachteten Strukturen zählen abdeckende Arrays, welche man als spezielle Klasse von kombinatorischen Designs ansehen kann, sowie auch gewisse Klassen von endlichen Sequenzen [28]. Das Ziel dieser Arbeit ist zu analysieren und darstellen, wie algebraische Methoden in der Spezifikation, Erzeugung und der Charakterisierung von Eigenschaften dieser Strukturen verwendet werden können [40]. Die zugrundeliegenden algebraischen Methoden basieren auf Polynomen [16].

Abstract

Since the introduction of algebraic techniques into the field of statistics in the start and middle of the 1990s [33, 105], both fields have immensely benefited from the resulting synergies [4, 104]. This Thesis is concerned with classes of combinatorial designs, that appear in a relatively new subfield called Combinatorial Testing (CT) for Software of Design of Experiments [81]. The notion of "coverage requirement", which represents a generalization of the well established notion of exactly λ-way appearance of t-tuples in orthogonal arrays, is fundamental to the field of CT and is also a fundamental property in the discrete structures that are considered in CT. These structures include covering arrays, which can be regarded as a special class of combinatorial designs, and certain classes of finite sequences [28]. The aim of this Thesis is to analyse and depict how algebraic techniques can help in the specification, generation and property assessment of these structures [40]. Polynomial algebraic techniques are the basic methodologies which are to be applied in this domain [16].

Publications arisen from this Thesis

[40] Bernhard Garn und Dimitris E. Simos. “Algebraic Modelling of Covering Arrays”. In: *Applications of Computer Algebra*. Hrsg. von Ilias S. Kotsireas und Edgar Martínez-Moro. Cham: Springer International Publishing, 2017, S. 149–170. ISBN: 978-3-319-56932-1.

[41] Bernhard Garn und Dimitris E. Simos. “Algebraic Techniques for Covering Arrays and related Structures”. In: *Electronic Notes in Discrete Mathematics* 70 (2018). TCDM 2018 – 2nd IMA Conference on Theoretical and Computational Discrete Mathematics, University of Derby, S. 49–54. ISSN: 1571-0653. DOI: https://doi.org/10.1016/j.endm.2018.11.008. URL: http://www.sciencedirect.com/science/article/pii/S1571065318302038.

[42] Bernhard Garn und Dimitris E. Simos. “Weighted t-way Sequences”. In: *Electronic Notes in Discrete Mathematics* 70 (2018). TCDM 2018 – 2nd IMA Conference on Theoretical and Computational Discrete Mathematics, University of Derby, S. 43–48. ISSN: 1571-0653. DOI: https://doi.org/10.1016/j.endm.2018.11.007. URL: http://www.sciencedirect.com/science/article/pii/S1571065318302026.

Contents

List of Figures

List of Algorithms

CHAPTER 1

Introduction

This Thesis deals primarily with algebraic methods for combinatorial mathematics. Indeed, the usage of transformation of structures and problems from one domain of mathematics (in particular, combinatorial design theory) into an algebraic setting is the means to be able to employ algebraic techniques and methods to reason and (hopefully) solve the transformed problems. The goal is that an algebraic solution – if it exists and if it was constructible – will be transformed back, constituting a solution to the initially posed problem within the initial branch of mathematics.

This work further deals with the field of combinatorial design theory. We focus on special classes of combinatorial designs – in particular, *covering arrays* – and are concerned with their existence, construction and manipulation.

The methods and structures considered have real-world applications in their usage to construct or constitute configurations for "experiments". This might be taken to indicate a connection of this Thesis with a field of mathematics called statistical design of experiments. This is very much not the case, although the design structures considered do have applications in "experimentations/tests", but (at the time of writing) strictly within a deterministic context. It is possible, in this deterministic, discrete context, given "results" of experiments/tests, to define a notion analogous to that of "statistical inference" with properties that one would expect and want for an instantiation of the concept of "inference" to have, but all strictly within a finite, discrete and deterministic context. We briefly comment on this rather recent real-world application of these design structures (in

the form of matrices) in a branch of software testing called *combinatorial testing*. It is important to highlight the notion and applications of such strictly discrete models, since the term *mathematical modelling* is usually understood as referring to a model consisting of (partial) differential equations. However, in this work, the term *modelling* is to be understood within a discrete context meaning as are dealing with a discrete, finite model over a finite domain.

We briefly indicate how all these different domains now come together. The treatment of notions and problems originating from combinatorial design theory within an algebraic framework is the core theme. Certain design structures are used in their matrix (array) formulation, paving the way for an application of linear methods. These methods result in multivariate polynomial expressions, upon which the semantic equivalence between design structure characteristics and zeros of polynomials is established. Here, computational algebra – in particular Gröbner bases – can be used to solve the resulting systems. Assuming that a corresponding variety is nonempty, the immediate next step is the transformation of a point in the variety into a design interpretation as solution to a problem. Then, one can use the generated matrix within computer science where its rows are interpreted as individual test cases to be used in software testing. This requires that the matrix is compatible with some discrete model of a piece of software. Based on this brief description of the approach followed, it should be clear that not only does this Thesis span multiple branches of mathematics, but its results and techniques have a direct impact in applied computer science, specifically, in the domain of test design and test creation. From this purely applied point of view, there is an interest to expand the presented algebraic models with additional structure corresponding to requirements needed in practice.

The primary goal of this work is to show how combinatorial properties can be translated into semantically equivalent algebraic statements. Once this has been achieved, the further development of an algebraic framework for certain classes of combinatorial designs, in particular of covering arrays, follows mechanistically since this step is simply the application of known techniques from computer algebra.

1.1 Motivation and Challenges

The algebraic methods discussed offer (at least) two advantages, which makes them appealing as a way to develop a framework for the treatment of certain design structures

(in particular, covering arrays). Recall that, design structures are usually collections of subsets of finite sets with certain intersection properties. First, with the methods presented, it is possible to algebraically enforce certain properties of interest. Second, "some work" can be given to an "algebraic solver", which will take some choices during its execution (by comparison, in heuristic construction approaches, e.g. in one-test-at-a-time extension strategies, there arises the problem of which test to choose next). Although the scope of this work does not include any stochastic considerations, it should not be overlooked that since the introduction of algebraic techniques into the field of statistics in the start and middle of the 1990s [33, 105], both fields have immensely benefited from the resulting synergies [4, 104].

The three main scientific domains of this Thesis are: combinatorial design theory, commutative and computer algebra and experimental design theory applications (DoE). The focus of this work lies in the presentation of the mathematical treatment of connections between combinatorial design theory and algebra. Nevertheless, we mention that some of the abstract artifacts considered are motivated by their interpretation from the application domain of *combinatorial testing*.

The algebraic framework presented offers "strong and nuanced modelling capabilities", at the price of shifting some "complexity inside" algebraic structures, and, as a consequence, to algebraic solvers. Notwithstanding the existence of sophisticated solvers, the problem remains on the resources needed to complete some computations.

1.2 Structure of this Thesis

This Thesis is structured as follows. In Chapter 2, we introduce the combinatorial designs, some of their properties and constructions that are of interest. In Chapter 3, we review and collect the necessary definitions and techniques from computational algebra. Subsequently, in Chapter 4, we introduce an algebraic framework for the first time and establish some connections between combinatorial design theory and computational algebra. Chapter 5 expands these connections by discussing algebraic means to solve specific design problems. In Chapter 6, we present applications of experimental design theory to *combinatorial testing* and *combinatorial sequence testing*. We conclude this Thesis in Chapter 7.

CHAPTER 2

Discrete Structures

In this chapter, we give the definitions and some background properties of the main discrete structures, in which we are interested in this thesis. All considered structures are finite and have in common, that their defining properties require the appearance of certain pre-determined elements from a finite set. Relaxing or strengthening of some conditions has an immediate impact on whether these structures exist or not for some configurations (i.e., the defining parameters of the design structure). If a structure for some configuration exists, we are particularly interested in determining its optimal size in terms of a minimization problem, where size in the case of arrays is usually understood as the number of rows or columns. The resulting combinatorial optimization problems can be treated with different techniques, and also – as in shown in this Thesis – with algebraic techniques.

We state the primary definitions of the considered structures in Section 2.1 and give alternative formulations and auxiliary structures in Section 2.2. Different approaches for the actual construction for the considered structures that have appeared in the literature are stated in Section 2.3.

2.1 Primary Structures

The following definition is paramount for this Thesis.

2.1.1 Definition ([28, 10.1])**.** *A* covering array $CA_\lambda(N; t, k, v)$ *is an* $N \times k$ *array. In every* $N \times t$ *subarray, each* t*-tuple occurs at least* λ *times. Then* t *is the strength of the*

coverage of interactions, k is the number of components (degree), and v is the number of symbols for each component (order). Only the case when $\lambda = 1$ is treated; the subscript is then omitted in the notation. The size N is omitted when inessential in the context. ◁

2.1.2 Definition ([28, 10.7])**.** *A* mixed level covering array $\mathsf{MCA}_\lambda(N; t, k, (v_1, v_2, \ldots, v_k))$ *is an $N \times k$ array. Let $\{i_1, \ldots, i_t\} \subseteq \{1, \ldots, k\}$, and consider the subarray of size $N \times t$ obtained by selecting columns $i_1, \ldots, i_t$ of the* MCA*. There are $\prod_{i=1}^{t} v_i$ distinct t-tuples that could appear as rows, and an* MCA *requires that each appear at least once.* $\mathsf{CAN}(t, k, (v_1, v_2, \ldots, v_k))$ *denotes the smallest N for which such a mixed covering array exists.* ◁

2.1.3 Terminology. *We call the requirements regarding the appearance of tuples in column selections in Definition 2.1.1 coverage-conditions.* ◁

2.1.4 Definition ([40, Definition 3])**.** *A* configuration *C for a mixed level covering array is a tuple $(t, k, (v_1, v_2, \ldots, v_k))$. When $v_1 = v_2 = \ldots = v_k = v$, then the specified* MCA *is in fact a covering array and we denote its configuration simply by (t, k, v).* ◁

In the scope of this work, we focus on the case of strength two covering arrays over the binary alphabet, i.e., $t = v = 2$.

2.1.5 Notation. *We will, however, denote and use (mixed-level) covering arrays M as their transpose*[1] *$M^\top$ in their matrix notation, following the terminology used in [54]. The later used statement "a matrix M is compatible with an* MCA *configuration $C = (t, k, (v_1, \ldots, v_k))$" is to be understood as implying that the matrix M has k columns and that its elements in the i-th column arise either from the set $\{0, \ldots, v_i - 1\}$ or constitute variables which take values exactly in $\{0, \ldots, v_i - 1\}$.* ◁

2.1.6 Example. *We give some examples*[2] *for covering arrays in the sense of Definition 2.1.1 in Figure 2.1. Note that the dashes (i.e., "-") in Figure 2.1c are a result of the method used to construct this array and denote entries in the matrix that are irrelevant from the standpoint of ensuring the required coverage properties and thus can be filled arbitrarily from the underlying binary alphabet in case a fully instantiated matrix is desired.* ◁

[1] We denote the transpose of a matrix M by $M^\top$.

[2] The *covering array* in Figure 2.1a, was constructed manually.

$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$

(a) $(2, 2, 2)$

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

(b) $(3, 4, 2)$[63]

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ - & - & - & 1 & 1 & - & 0 \\ - & - & - & 0 & 1 & - & 1 \end{pmatrix}$$

(c) $(3, 7, 2)$[64]

Figure 2.1: *Covering array* examples for different configurations given in matrix notation.

2.1.7 Remark. *A branch of software testing called combinatorial testing uses covering arrays as underlying mathematical artifacts to create test sets for software. Assume that a covering array in the sense of Definition 2.1.1 with $N \in \mathbb{N}$ rows is given, which is compatible with a configuration (t, k, v), then in the context of combinatorial testing the k columns are interpreted as k parameters of a system under test (SUT), each taking values in a finite set of cardinality v. The rows of the covering array are then used as tests, where the entries in the individual rows correspond to the parameter values which in their unison constitute a specific "configuration" of the SUT under which it is "run". We elaborate on this topic in Section 6.1.* ◁

The next theorem (Theorem 2.1.8) establishes the universal existence of MCAs and, in particular, the universal existence of *covering arrays*.

2.1.8 Theorem. *For all* MCA *configurations C, there exists a* MCA*, which is compatible with C.* ◁

Proof. Consider the array those rows are exactly the elements of[3] $\prod_{i=1}^{k} [v_i]$, in any

[3] For $n \in \mathbb{N}$, let $[n] = \{0, 1, \ldots, n-1\}$.

enumeration. □

2.1.9 Definition ([83, Problem 1])**.** *Let C be a* CA *configuration. The least value of $N \in \mathbb{N}^{\times}$, such that there exists a* CA *compatible with C with N rows, is called the covering array number (*CAN*) of C.* ◁

2.1.10 Remark. *Definition 2.1.9 is well defined by Theorem 2.1.8.* ◁

2.1.11 Remark. *A similar minimality problem as given in Definition 2.1.9 can – mutatis mutandis – be formulated for* MCAs. ◁

2.1.12 Lemma. *Let M be a matrix with $N \in \mathbb{N}$ rows compatible with a* CA *configuration $C = (t, k, v)$. Then, the following statements hold.*

1. *Let ρ be an element of the symmetric group on N symbols and Ξ a coverage-condition. Then, the matrix M fulfills Ξ iff the matrix obtained by applying ρ to the rows of M fulfills Ξ.*
2. *Let π be an element of the symmetric group on k symbols. Then, a coverage-condition holds for M iff the by π transformed coverage-condition[4] holds for the matrix obtained by applying π to the columns of M.*

In other words, coverage-conditions are invariant under row-permutations and are semantically-equivalently transformed for a permutation of the columns of M. ◁

Proof. The claims follow directly from Definition 2.1.1. □

2.1.13 Corollary. *Let M be a covering array for a configuration C. The result of the application of finitely many row- or column permutations to the array M is again a covering array for the similarly transformed configuration C.* ◁

Proof. Follows by induction and Lemma 2.1.12. □

2.2 Alternative Formulations and Auxiliary Structures

We make some comments to the definition of *covering arrays* in Section 2.2.1 and then look at auxiliary structures which are related to *covering arrays* or represent generalizations in Section 2.2.2.

[4] The result of which is simply a relabeling according to the permutation.

2.2.1 Alternative Definitions

We would like to put emphasis on the fact that the existential quantifier (i.e.,, the quantifier inside the scope of the two universal quantifiers) given in Definition 2.1.1 is to be strictly-formally interpreted as an *existential quantifier* and that the enumeration of the rows in the array is (at this point for this abstract view) irrelevant and that a pair appears as part of (at least one) specific row (e.g., in the "first/last/third row") determined by the enumeration of the rows of the matrix is possibly even misleading the essential focus and intention. To put it differently, *where exactly* a certain binary pair appears in some sub-selection of columns of the matrix is irrelevant, the crucial point is that this tuple appears at all, which is semantically also captured by the phrase "at least once". These purely existential criteria are put more into focus in the *set cover formulation* of *covering array* problems given below.

2.2.1 Definition ([69, Definition 2.2])**.** *A* set cover *(SC) of a finite set U, is a set $\mathcal{S}$ of nonempty subsets of U whose union is U. In this context, we call U the* universe*, and refer to the elements of $\mathcal{S}$ as* blocks*. A set cover consisting of pairwise disjoint blocks is called an* exact cover*. A set cover consisting only of blocks of cardinality d is called a* d-set cover*.* ◁

2.2.2 Definition ([69, Problem 2.4])**.** *(Minimal Set Cover (MSC)) Given a finite set U and a set cover $\mathcal{S}$ of U, i.e. $\bigcup \mathcal{S} = U$, find one subset $\mathcal{C}$ of $\mathcal{S}$, of minimal cardinality, such that $\bigcup \mathcal{C} = U$.* ◁

2.2.3 Definition ([69, Definition 3.1])**.** *For positive integers t, k and v with $t \leq k$, we define a* v-ary (k,t)-tuple *as a pair $((x_1 \ldots, x_t), (p_1, \ldots, p_t))$ with the property that $x_i \in \{0, \ldots, v-1\},\ \forall i \in \{1, \ldots, t\}$ and $1 \leq p_1 < \ldots < p_t \leq k$.* ◁

2.2.4 Definition ([69, Definition 3.2])**.** *For positive integers t, k and v with $t \leq k$, we say that a vector $r \in \{0, \ldots, v-1\}^k$* covers *a v-ary (k,t)-tuple $((x_1 \ldots, x_t), (p_1, \ldots, p_t))$, if the entries of r in positions p_i equal x_i for all $i = 1, \ldots, t$. Further we denote with $\varphi_{(v,k,t)}$ the function, which maps each $r \in \{0, \ldots, v-1\}^k$, to the set of $\binom{k}{t}$ v-ary (k,t)-tuples that are covered by r.* ◁

Via Definition 2.2.4 the authors of [69] give a construction how to map problems pertaining *covering arrays* to specialized set cover problems. We note that set-cover formulations for *covering arrays* have also appeared (in various degree of detail) in [125, 115, 54].

2.2.2 Auxiliary Structures

In the literature, there have appeared many combinatorial design structures that can be related to the defining property of covering certain tuples. Some of these structures arise as special cases of *covering arrays*, other emerge as the result of relaxing some requirements in the definition of *covering arrays*. We also observe that the literature is combinatorial designs, their connection between them and their connections with closely related branches of mathematics like graph theory or the theory of error correcting codes is quite intensive and is explored in detail in various specialized scientific literature.

2.2.5 Definition ([56, Definition 1.1])**.** *Let S be a set of s symbols. An $N \times k$ array A with entries from S is said to be an orthogonal array with s levels, strength t and index λ (for some t in the range $0 \leq t \leq k$) if every $N \times t$ subarray of A contains each t-tuple based on S exactly λ times as a row.* ◁

2.2.6 Definition ([93, Definition 1])**.** *A set of vectors with entries from $\mathbb{Z}_g$ are t-qualitatively independent if for any t-subset, $\{v_i\}$, of vectors and any ordered t-tuple of elements $(g_1, g_2, \ldots, g_t) \in \mathbb{Z}_g^t$ there exists a j such that for each vector v_i the jth coordinate $v_{ij} = g_i$.* ◁

2.2.7 Definition ([93, Definition 3])**.** *A covering array on a graph G with alphabet size $g, k = |V(G)|$ is a $k \times n$ array on $\mathbb{Z}_g$. Each row in the array corresponds to a vertex in the graph G. The covering array has the property that pairs of rows which correspond to adjacent vertices in the graph are qualitatively independent.*

A covering array on a graph G will be denoted as $CA(n, G, g)$. The smallest possible covering array on a graph G will be written

$$CAN(G, g) = \min_{l \in \mathbb{N}} \{l \colon \exists CA(l, G, g)\}. \tag{2.1}$$

We call $CAN(G, g)$ either the g-qualitative independence number of G or g-ary covering array number of G depending on the point of view. ◁

2.2.8 Definition ([32, pp.5405-5406 and Definition 2])**.** *Let $G = G_{(g_1,\ldots,g_k)}$ denote a graph with k parts of sizes $g_1, \ldots, g_k$ that is k-partite except for the possible existence of loops. The vertices of G are v_{i,a_i}, indexed by i, a_i where $i \in [1, k]$ and $a_i \in [k]$ and $a_i \in [g_i]$. If $g_1 = \cdots = g_k = g$, then we simplify the notation to $G = G_{k,p}$. We define a graph $G^{|}$ on the same vertex set as G and include the edges from $E(G)$ but also containing all the edges $\{v_{i,a}, v_{i,b}\}$ for $a \neq b \in [g_i]$.*

- *A graph G is said to be factor connected if $G^{|}$ is connected; factor-connected components of G correspond to components of $G^{|}$.*
- *A k-tuple $T = (T_1, \ldots, T_k) \in [g_1] \times \cdots \times [g_k]$ is said to avoid $G = G_{(g_1,\ldots,g_k)}$ if for all $i, j \in [1, k]$, we have $\{v_{i,T_i}, v_{j,T_j}\} \notin E(G)$.*
- *We say that an interaction $\{(i, a), (j, b)\}$, with $i \neq j$ if $a \neq b$, such that $\{v_{i,a}, v_{j,b}\} \notin E(G)$ is consistent with G if there exists a k-tuple T with $T_i = a$ and $T_j = b$ that avoids G.*
- *A graph is consistent if all interactions $\{(i, a), (j, b)\}$, with $i \neq j$ if $a \neq b$, where $\{v_{i,a}, v_{j,b}\} \notin E(G)$ are consistent.*

A covering array with forbidden edges for a graph $G = G_{(g_1,\ldots,g_k)}$ is an $n \times k$ array A with each column i having symbols from the alphabet $[g_i]$, and denoted by $CAFE(n, G)$, such that

1. *each row of A forms a k-tuple avoiding G;*
2. *for all $v_{i,a}, v_{j,b} \in V(G)$ with $i \neq j$, if $\{v_{i,a}, v_{j,b}\} \notin E(G)$, then there exists a row r such that $A_{r,i} = a$ and $A_{r,j} = b$.*

We denote by $CAFEN(G)$ the minimum n for which there exists a $CAFE(n, G)$, if such an object exists, or $+\infty$ otherwise. ◁

2.2.9 Definition ([111, p. 1474])**.** *When $1 \leq m \leq v^t$, a partial m-covering array, $PCA(N; t, k, v, m)$, is an $N \times k$ array A with each entry from $[v]$ so that for each t-set of columns $C \in \binom{[k]}{t}$, at least m distinct tuples $x \in [v]^t$ appear as rows in A_C.* ◁

2.2.10 Definition ([96, Definition 1.1])**.** *Let $H = (V, E)$ be a hypergraph and let $k = |V|$. A variable-strength covering array, denoted $VCA(n; H, v)$, is an $n \times k$ array M filled from $\mathbb{Z}_v$ such that for $e = \{v_0, \ldots, v_{t-1}\} \in E$, the $n \times t$ subarray of columns indexed by e is covered, that is it has every possible t-tuple in $\mathbb{Z}_v$ as a row at least once. The variable-strength covering array, written $VCA(H, v)$, is the smallest n such that a $VCA(n; H, v)$ exists.* ◁

2.2.11 Definition. *In the situation of Definition 2.2.10, additionally assume that the hypergraph is in fact a labeled hypergraph, where the labels are nonempty elements in the*

power set of the following Cartesian product $\prod_{i=1}^{t} v_i$. *We refer to such a structure as a partial coverage system. Compatibility of a matrix with a partial coverage system is defined similarly to the case of covering arrays.* ◁

When working with concrete given matrices, it is (sometimes) necessary to extend Definition 2.2.11 to consider each row of the matrix separately; which will be accomplished in Definition 2.2.12.

2.2.12 Definition ([41, Definition 2.3]). *For* $k, N \in \mathbb{N}$ *with* $2 \leq k$, *let* $C = (2, k, 2)$ *be a configuration and* M *a compatible* $N \times k$ *matrix. We call the function* $\Gamma \colon [N] \times \mathcal{I}_k \longrightarrow \mathcal{P}(\mathcal{T})$ *an* interaction-membership function. *This function is interpreted as assigning, to each unordered selection of two different columns per row, a set of binary 2-tuples that are allowed to appear at this position. Such a function may specify contradicting conditions.* ◁

2.3 Construction Approaches

Although for Theorem 2.1.8 we gave a constructive proof for the universal existence of MCAs, the focus lies on the construction of near-optimal or optimal arrays in terms of their size. There exists considerable literature for attacking this problem with techniques from various fields of mathematics. For more details we refer to [133, 98, 78, 82, 70, 117, 83].

Moreover, there are many software implementations available for the construction of *covering arrays* [67].

2.3.1 Mathematical Construction Methods

Observing that *orthogonal arrays* are in fact also *covering arrays*, we refer to [56] for constructions of *orthogonal arrays* and mentioned references there in.

Group constructions of *covering arrays* are given in [94, 18, 88].

A recursive construction for *covering arrays*, which uses existing *covering arrays* to create a *covering array* for the concatenation of the respective configurations was given in [68].

A construction for *covering arrays* from linear-feedback-shift-register sequences over finite fields was given in [121].

Approaches using augmentation, which is an operation to increase the number of symbols in a covering array, without unnecessarily increasing the number of rows, were given in [25, 27].

2.3.2 One-Test-at-a-Time

The idea behind greedy algorithms following a one-test[5]-at-a-time strategy is simple: start with an empty array and add rows to the array, until all coverage-conditions are fulfilled. During this process, in each step a (local) optimum in terms of newly added coverage for a candidate row is determined[6] and is added to the array. One-test-at-a-time strategies have been proposed in [20, 31, 134, 13, 9]. A framework for the evaluation of a large class of greedy methods that follow a one-test-at-a-time strategy was given in [14].

2.3.3 In Parameter Order Family

The in-parameter[7]-order (IPO) strategy builds first a *covering array* for only a part of a given configuration and then add more columns to the array (i.e., the already constructed array is horizontally concatenated with a greedy generated column of compatible size such that if the resulting matrix does not exhibit full coverage, then for all yet-unsatisfied coverage-conditions a row is added to the array (i.e., horizontally concatenated) which fulfills the coverage-condition. Once all parameters have been added, a *covering array* for the initially given configuration has been constructed. The following works deal with algorithms using an IPO strategy [84, 85, 38, 86, 73].

2.3.4 Evolutionary Computation and Metaheuristics

Metaheuristic search and evolutionary algorithms have been used successfully to construct *covering arrays*. Methods of swarm intelligence have been applied for the generation of *covering arrays* in [1, 2, 19, 92, 114]. Heuristic search techniques were followed in [10, 44, 99]. Simulated annealing has been applied to various problems of *covering arrays*

[5]For covering arrays in the sense of Definition 2.1.1 these algorithms could also be described as "one-row-at-a-time". However, the usually employed terminology is "one-test-at-a-time", because the rows of *covering arrays* in the sense of Definition 2.1.1 can be used to encode configurations for software testing. Each such configuration is colloquially referred to as a test. See Remark 2.1.7 and Section 6.1.

[6]The exact notion of locality and the exact instance of objective function used for in these optimization problems depend on the considered strategy and implementation decisions.

[7]For *covering arrays* in the sense of Definition 2.1.1, the columns of an array are interpreted as parameters of a piece of software in a real-world application of designs in a branch of software testing. See Remark 2.1.7 and we elaborate on this application in Section 6.1

[21, 108, 119]. In [46, 90, 91, 114], genetic algorithms were used for the construction of *covering arrays*. Approaches using tabu search were considered in [48, 47, 100, 122].

2.3.5 Approaches based on Formal Logic

Similar to the translation of coverage-conditions into an algebraic setting there have been works performing an analogous translation, but to a different domain. These target domains include SAT formulations, constraint programming, integer programming and set cover formulations with respective construction methods.

Approaches based on SAT and Constraint Programming

Constraint programming models of the problem of finding an optimal covering array were developed in [57]. SAT solving for the generation of *covering arrays* was employed in [72, 132]. In [89], the authors presented an enhanced backtrack search algorithms for *orthogonal arrays* using a SAT or pseudo-Boolean constraint solver. In [128], incremental SAT solving was considered. In [127], a correspondence between forbidden tuples and unsatisfiable cores is illustrated and integrated into a greedy test case generation approach.

Approaches based on Integer Programming and Set Cover Solvers

Coverage-conditions can be formulated in terms of integer programming problems or in terms of set cover problems, as has been done in [115, 125, 69].

2.3.6 Post-Optimization of Covering Arrays

In the case where for given *covering array* configuration the corresponding CAN is not known[8], instead of creating a new *covering array* for that configuration with the hope that it will be smaller, one can try to permute/change the existing array entries in such a way that full coverage will already be achieved with less than the total number rows. In other words, the goal is that by changing/switching[9] of some array entries at least the last row of the array will contain entries which do not contribute any more to the fulfillment of the coverage requirements meaning that all tuples determined by the last row will have already appeared as part of another row somewhere else. This process can be iterated until some stopping criteria is met.

[8] In general, the values for CAN are not known [83].
[9] Depending on the used strategy.

In [50], a strategy was that reduces the number of rows of a *covering array*. Randomized post-optimization approaches were considered in [97, 87, 29]. In [118], the authors performed metaheuristic post-optimization of the NIST repository of *covering array*. A branch and bound strategy that maximizes the number of wild cards in the profile of an already constructed *covering array* was presented in [49]. A graph-based post-optimization approach was given in [102]. In [71], some criteria were proposed for identifying the best choices for the wild card positions to create covering arrays with highly desirable properties.

CHAPTER 3

Polynomial System Solving and Gröbner Bases

In this chapter, we give the necessary definitions and develop the techniques from commutative and computational algebra, that will be used throughout this Thesis. The algebraic translation of constructive problems arising in design theory discussed in this Thesis are multivariate polynomial systems of equations. Gröbner bases are the means by which such systems can be computationally solved. If such a system has a solution, meaning that the corresponding variety is nonempty, then each point in the variety encodes a complete or part of a design matrix and, in particular, this point (i.e., solution) can be transformed into a design matrix representation, since its coordinates correspond to entries in a (pre-determined) matrix. As a result, algebraic solutions (if they exist) to the algebraic formulation of a constructive design problem give rise to design matrices, which constitute affirmative existential solutions to the constructive problem.

For a general treatment of Gröbner bases theory we refer to the extensive materials provided in [5, 30, 126].

We discuss general reduction relations in Section 3.1 and then turn to polynomial reductions in Section 3.2. We study the computation of Gröbner bases in Section 3.3 and apply them to the problem of polynomial system solving in Section 3.4.

3.1 Reduction Relations

A treatment of the theory of reduction relations was given in [58]. We develop some of their important properties to be able to define the notion of a Gröbner basis using properties for a certain reduction relation on multivariate polynomials.

3.1.1 Definition ([126, Definition 8.1.1])**.** *Let M be a set and $\to$ a binary relation on M, i.e. $\to \subseteq M \times M$. We call $\to$ a* reduction relation *on M. For $(a, b) \in M \times M$, the usual notation $a \to b$ will be used instead of $(a, b) \in \to$ and we say that a* reduces to *b.*

Let $\to$ and $\to'$ be reduction relations on M, then we define the following operations on $M \times M$ for constructing new reduction relations.

1. *$\to \circ \to'$ (or just $\to\to'$), the* composition *of $\to$ and $\to'$, is the reduction relation as $a \to\to' b$ iff $\exists c \in M : a \to c \to' b$;*

2. *$\to^{-1}$ (or just $\leftarrow$), the* inverse relation *of $\to$, is the reduction relation defined as $a \leftarrow b$ iff $b \to a$;*

3. *$\to_{sym}$ (or just $\longleftrightarrow$), the* symmetric closure *of $\to$, is the reduction relation defined as $\to \cup \leftarrow$, i.e., $a \to_{sym} b$ iff $(a \to b \vee a \leftarrow b)$.*

4. *$\to^i$, the i-*th power *of $\to$, is the reduction relation defined inductively for $i \in \mathbb{N}$ as*

   $\to^0 = id$, where id is the identity relation on M; i.e., $a \to^0 b$ iff $a = b$, and

   $\to\to^{i-1}$ for $i \geq 1$.

   This means that $a \to^i b$ iff there exist $c_0, \ldots, c_i$ such that

   $$a = c_0 \to c_1 \to \cdots \to c_i = b. \tag{3.1}$$

   In this case we say that a reduces to *b* in *i* steps*;

5. *$\to^+ = \cup_{i=1}^{\infty}$, the* transitive closure *of $\to$;*

6. *$\to^+ = \cup_{i=1}^{\infty}$, the* reflexive-transitive closure *of $\to$;*

7. *$\to^* = \cup_{i=1}^{\infty}$, the* reflexive-transitive-symmetric closure *of $\to$.*

◁

3.1.2 Proposition ([126, Section 8.1]). *The binary relation $\to^*$ is an equivalence relation on M and we denote with $M/_{\to^*}$ the corresponding partition consisting of equivalence classes modulo $\to^*$.* ◁

We continue with more definitions.

3.1.3 Definition ([126, Definition 8.1.2]). *1. $x \to$ means that x is* reducible*, i.e., $x \to y$ for some $y \in M$;*

2. $\underline{x}_\to$ means x is irreducible *or in* normal form *with regard to $\to$;*

3. $x \downarrow y$ means that x and y have a common successor*, i.e., $\exists z \in M \colon x \to z \leftarrow x$;*

4. $x \uparrow y$ means that x and y have a common predecessor*, i.e., $\exists z \in M \colon x \leftarrow z \to x$;*

*5. x is a $\to$-*normal form *of y iff $y \to^* \underline{x}_\to$.*

◁

3.1.4 Definition ([126, Definition 8.1.3]). *1. $\to$ is* Noetherian *or has the* termination property *iff every reduction sequence terminates, i.e., there is no infinite sequence $x_1, x_2, \ldots$ in M such that $x_1 \to x_2 \to \cdots$.*

2. $\to$ is Church-Rosser *or has the* Church-Rosser property *iff $a \longleftrightarrow^* b$ implies $a \downarrow_* b$.*

◁

3.1.5 Remark ([126, Section 8.1]). *Let M be a set and $\to$ a Noetherian relation. The property of being Noetherian for a relation can be used to define the* principle of Noetherian induction *that can be used to prove that a predicate P holds for all $x \in M$ in the following sense ([23]):*
If for all $x \in M$:

$$[\forall y \in M \colon (x \to y) \implies P(y)] \implies P(x), \tag{3.2}$$

then

$$\forall x \in M \colon P(x). \tag{3.3}$$

◁

3.1.6 Definition ([126, Definition 8.1.4]). *1. $\to$ is* confluent *iff*

$$x \uparrow^* y \implies x \downarrow^* y. \tag{3.4}$$

The implication stated in equation (3.4) *is represented graphically in Figure 3.1.*

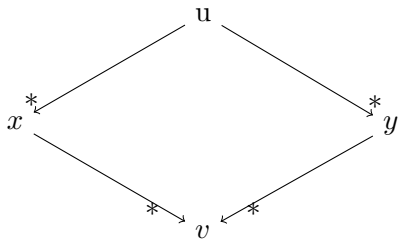

Figure 3.1: Graphical illustration related to the confluence property of a reduction relation.

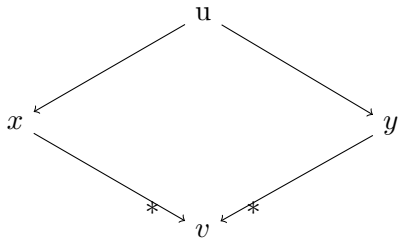

Figure 3.2: Graphical illustration related to the confluence property of a reduction relation.

2. $\rightarrow$ *is* locally confluent *iff*

$$x \uparrow y \implies x \downarrow_* y. \tag{3.5}$$

The implication stated in equation (3.5) *is represented graphically in Figure 3.2.*

◁

3.1.7 Theorem ([126, Theorem 8.1.2])**.** *1.* $\rightarrow$ *is Church-Rosser iff* $\rightarrow$ *is confluent.*

2. (Newman lemma) Let $\rightarrow$ *be Noetherian. Then* $\rightarrow$ *is confluent iff* $\rightarrow$ *is locally confluent.*

◁

Proof. 1: If $\rightarrow$ is Church-Rosser, then it is obviously confluent. For the other direction, assume that $\rightarrow$ is confluent. Suppose that $x \rightarrow^* y$ in n steps, i.e., $x \rightarrow^n y$. We use induction on n. The case $n = 0$ is immediate. For $n > 0$, we distinguish two cases (graphically depicted in Figures 3.3 and 3.4):

$$x \rightarrow z \rightarrow^{n-1} y, \tag{3.6a}$$

$$x \leftarrow z \rightarrow^{n-1} y. \tag{3.6b}$$

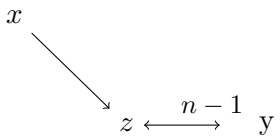

Figure 3.3: Graphical illustration corresponding to case (3.6a).

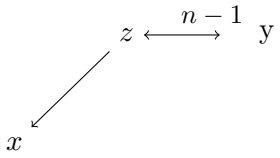

Figure 3.4: Graphical illustration corresponding to case (3.6b).

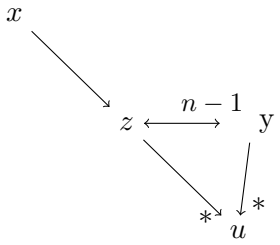

Figure 3.5: Graphical illustration of existence of u, corresponding to case (3.6a).

In case (3.6a), by the induction hypothesis there is a u such that (graphically depicted in Figure 3.5)

$$z \to^* u^* \leftarrow y. \tag{3.7}$$

In case (3.6b), by the induction hypothesis and by confluence there are u, v such that (graphically depicted in Figure 3.6)

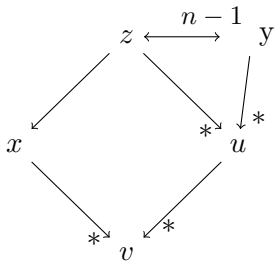

Figure 3.6: Graphical illustration of existence of u, v, corresponding to case (3.6b).

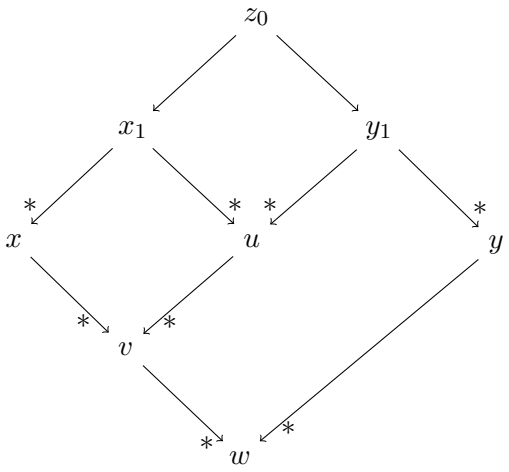

Figure 3.7: Graphical illustration of existence of u, v, w, corresponding to (3.9).

$$(z \to^* u^* \leftarrow y) \wedge (x \to^* v^* \leftarrow u)\,. \tag{3.8}$$

In either case, we have $x \downarrow^* y$.

2: Confluence implies local confluence. For the other direction, assume that $\to$ is locally confluent. To prove the claim, we use Noetherian induction on the Noetherian relation $\to$. Suppose that $x^* \leftarrow z_0 \to^* y$. The cases $x = z_0$ and $y = z_0$ are immediate. Therefore, consider

$$x^* \leftarrow {x_1}^* \leftarrow z_0 \to^* y_1 \to^* y. \tag{3.9}$$

By local confluence ((3.10a)) and the induction hypothesis ((3.10b),(3.10c)) there are u, v, w such that

$$x_1 \to^* u^* \leftarrow y_1 \tag{3.10a}$$
$$x \to^* v^* \leftarrow u \tag{3.10b}$$
$$v \to^* w^* \leftarrow y \tag{3.10c}$$
$$\tag{3.10d}$$

The situation is graphically depicted in Figure 3.7. Therefore, $x \downarrow^* y$. □

3.1.8 Definition ([126, Definition 8.1.5])**.** *Let $\to$ be a reduction relation on the set M and $>$ a partial ordering on M. Suppose that $x, y, z \in M$. x and y are* connected (with regard to $\to$) *below (with regard to $>$) z iff*

$$\exists w_1, \ldots, w_n \in M \left((x = w_1 \longleftrightarrow \cdots \longleftrightarrow w_n = y) \wedge \bigwedge_{i=1}^{n} (w_i < z) \right). \tag{3.11}$$

This property is denoted by $x \longleftrightarrow^*_{(<z)} y$. ◁

3.1.9 Theorem ([126, Theorem 8.1.3])**.** *(Refined Newman lemma) Let* $\to$ *be a reduction relation on* M *and* $>$ *a partial Noetherian ordering on* M *such that* $\to \subseteq >$*. Then,* $\to$ *is confluent iff*

$$\forall x, y, z \in M \left((x \leftarrow z \to y) \implies x \longleftrightarrow^*_{(<z)} y \right). \tag{3.12}$$

◁

Proof. Confluence implies connectedness. For the other direction, assume that the connectedness property holds. We use Noetherian induction on $>$ with the (first) induction hypothesis

$$\forall \tilde{x}, \tilde{y}, \tilde{z}\colon \text{ if } \tilde{z} < z \wedge \tilde{x}^* \leftarrow \tilde{z} \to^* \tilde{y} \text{ then } \tilde{x} \downarrow_* \tilde{y}. \tag{3.13}$$

Consider the situation $x \to^* z \to^* y$. If $x = z$ or $y = z$, then we are done. Otherwise, we have

$$x^* \leftarrow x_1 \leftarrow z \to y_1 \to^* y. \tag{3.14}$$

By the assumption of connectedness, there are $u_1, \ldots, u_n < z$, such that

$$x_1 = u \longleftrightarrow \cdots \longleftrightarrow u_n = y_1. \tag{3.15}$$

We use induction on n to show that for all n and all $u_1, \ldots, u_n \in M$:

$$\left(u_1 \longleftrightarrow \cdots \longleftrightarrow u_n \wedge \bigwedge_{i=1}^{n} (u_i < z) \right) \implies u_1 \downarrow_* u_n. \tag{3.16}$$

The case $n = 1$ is clear. Next, we assume that

$$\text{(3.16) holds for some } n \geq 2, \tag{3.17}$$

and this is the second induction hypothesis. For the induction step, assume $u_1, \cdots, u_{n+1} \in M$ with $u_1 \longleftrightarrow \cdots \longleftrightarrow u_{n+1}$ with $u_i < z$, for $1 \leq i \leq n+1$. We distinguish two cases, where both show the existence of a common successor v to u_1 and u_{n+1}:

$u_n \leftarrow u_{n+1}$ By (3.17), we have

$$\exists v \in M \, (u_1 \to^* v_* \leftarrow u_n) \tag{3.18}$$

This v has the required property and the corresponding situation is depicted in Figure 3.8.

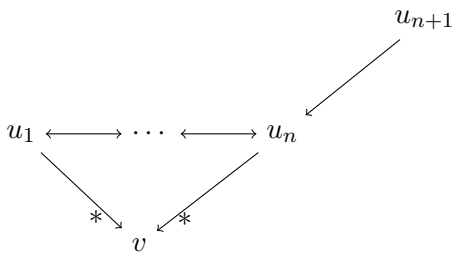

Figure 3.8: Graphical illustration of existence of common successor v to u_1 and u_{n+1} corresponding to case in (3.18).

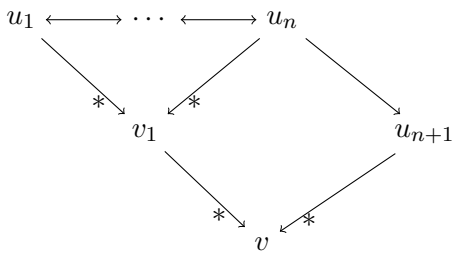

Figure 3.9: Graphical illustration of existence of common successor v to u_1 and u_{n+1} corresponding to case in (3.19b) and (3.19b).

$u_n \to u_{n+1}$ By (3.17) and (3.13), it holds that

$$v_1 \in M\left(u_1 \to^* {v_1}^* \leftarrow u_n\right), \tag{3.19a}$$

$$v \in M\left(v_1 \to^* v^* \leftarrow u_{n+1}\right). \tag{3.19b}$$

This v has the required property and the corresponding situation is depicted in Figure 3.9).

The results of the case distinction together yield that (3.16) is established. Finally, the following three steps complete the proof of the theorem (which are illustrated in Figure 3.10):

$$\exists w_1 \in M\left(u_1 \to^* {w_1}^* \leftarrow u_n\right), \text{ by (3.16)}; \tag{3.20a}$$

$$\exists v \in M\left(x \to^* v^* \leftarrow w_1\right), \text{ by (3.13)}; \tag{3.20b}$$

$$\exists w \in M\left(v \to^* w^* \leftarrow y\right), \text{ by (3.13)}. \tag{3.20c}$$

□

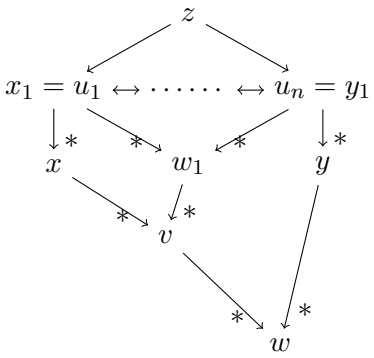

Figure 3.10: Graphical illustration for final steps in the proof.

3.2 Polynomial Reduction

For $n \in \mathbb{N}^{\times}$ and a commutative unary ring R, consider the polynomial ring in n indeterminates over R, denoted as $K[X] = K[x_1, \ldots, x_n]$. For any subset $F \subseteq K[X]$, we denote the generated ideal by $\langle F \rangle$. The monoid under multiplication of the *power products* $x_1^{i_1} \cdots x_n^{i_n}$ in $x_1, \ldots, x_n$ with unit element $1 = x_1^0 \cdots x_n^0$ is denoted by $[X]$; and $\operatorname{lcm}(s, t)$ denotes the least common multiple of the power products $s, t \in [X]$.

3.2.1 Definition ([126, Section 8.2])**.** *The* basis condition *for a commutative unary ring states that every ideal has a finite basis.* ◁

3.2.2 Definition ([126, Section 8.2])**.** *Commutative unary rings, in which the basis condition holds, are called Noetherian ring.* ◁

3.2.3 Lemma ([126, Lemma 8.2.1])**.** *In a Noetherian ring, there are no infinitely ascending chains of ideals.* ◁

3.2.4 Theorem ([126, Theorem 8.2.2])**.** *(Hilbert's basis theorem) If R is a Noetherian ring, then also the univariate polynomial ring $R[x]$ is Noetherian.* ◁

From *Hilbert's basis theorem*, Theorem 3.2.4, it follows that the multivariate polynomial ring $K[X]$ is Noetherian, if K is a field [126, Section 8.2].

3.2.5 Definition ([126, Definition 8.2.1])**.** *Let $<$ be a total ordering on $[X]$ that is compatible with the monoid structure in the following sense:*

1. $\forall t \in [X] \setminus \{(1\}\colon 1 < t)$, *and*

2. $\forall s,t,u \in [X]\ (s < t \implies su < tu)$.

A total ordering on $[X]$*, that satisfies both above conditions, is called an* admissible ordering. ◁

3.2.6 Example ([126, Example 8.2.1]). *1. For a a permutation* π *of* $[n]$*, the* lexicographic ordering *with* $x_{\pi(1)} > x_{\pi(2)} > \cdots > x_{\pi(n)}$*, is defined as follows:*

$$x_1^{i_1}x_2^{i_2}\cdots x_n^{i_n} <_{\mathsf{lex},\pi} x_1^{j_1}x_2^{j_2}\cdots x_n^{j_n} \Leftrightarrow \exists k \in [n]\left(\left(\forall l < k\colon i_{\pi(l)} = j_{\pi(l)}\right) \wedge i_{\pi(k)} < j_{\pi(k)}\right). \tag{3.21}$$

In the case that $\pi = id$*, then the lexicographic ordering* $<_{\mathsf{lex},\pi}$ *is the usual lexicographic ordering* $<_{\mathsf{lex}}$*.*

2. The graduated lexicographic ordering *with regard to the permutation* π *and the weight function* $w\colon \{1,\ldots,n\} \longrightarrow \mathbb{R}^+$*: for* $s = x_1^{i_1}\cdots x_n^{i_n}, t = x_1^{j_1}\cdots x_n^{j_n}$ *we define* $s <_{glex,\pi,w} t$ *iff*

$$\left(\sum_{k=1}^{n} w(k)i_k < \sum_{k=1}^{n} w(k)j_k\right) \vee \left(\sum_{k=1}^{n} w(k)i_k = \sum_{k=1}^{n} w(k)j_k \wedge s <_{lex,\pi} t\right). \tag{3.22}$$

In the case that $\pi = id$ *and* w *is constant and equal to one, then one obtains the usual graduated lexicographic ordering* $<_{glex}$*.*

3. The graduated reverse lexicographic ordering is defined as follows: $s <_{grlex} t$ *iff*

$$(\deg(s) < \deg(t)) \vee (\deg(s) = \deg(t) \wedge t <_{lex,\pi} s, \pi(j) = n - j + 1). \tag{3.23}$$

4. The product ordering with regard to $i \in \{1,\ldots,n-1\}$ *and the admissible orderings* $<_1$ *on* $X_1 = [x_1,\ldots,x_i]$ *and* $<_2$ *on* $X_2 = [x_{i+1},\ldots,x_n]$*: for* $s = s_1s_2, t = t_1t_2$*, where* $s_1,t_1 \in X_1, s_2,t_2 \in X_2$*, we define* $s <_{prod,i,<_1,<_2} t$ *iff*

$$(s_1 <_1 t_1) \vee (s_1 = t_1 \wedge s_2 <_2 t_2). \tag{3.24}$$

◁

For an in-depth treatment of admissible orderings, we refer to [107, 106].

3.2.7 Lemma ([126, Lemma 8.2.3]). *Let* $<$ *be an admissible ordering on* $[X]$*.*

1. If $s,t \in [X]$ *and* s *divides* t*, then* $s \leq t$*.*

2. $>$ is Noetherian, and consequently every subset of $[X]$ has a smallest element.

◁

Proof. 1: For some u we have $su = t$. By the admissibility of $<$, $s = 1s \leq us = t$.
2: Let $s_1 > s_2 > \cdots$ be a sequence of decreasing elements in $[X]$. Let K be any field. So the sequence of ideals $\langle s_1 \rangle \subset \langle s1, s2 \rangle \subset \cdots$ in $K[X]$ is increasing. But $K[X]$ is *Noetherian*, thus the sequence has to be finite. □

3.2.8 Definition ([126, Definition 8.2.2])**.** *The following list contains, for $s \in [X], f \in K[X] \setminus \{0\}$ and $F \subseteq K[X]$, both definitions and terminology in unison.*

1. $\mathsf{coeff}(f, s)$ *denotes the coefficient of s in f;*
2. $\mathsf{lpp}(f) = \max_<\{t \in [X] : \mathsf{coeff}(f, t) \neq 0\}$*, the leading power product of f;*
3. $\mathsf{lc}(f) = \mathsf{coeff}(f, \mathsf{lpp}(f))$*, the leading coefficient of f;*
4. $\mathsf{in}(f) = \mathsf{lc}(f)\,\mathsf{lpp}(f)$*, the initial of f;*
5. $\mathsf{red}(f) = f - \mathsf{in}(f)$*, the reductum of f;*
6. $\mathsf{lpp}(F) = \{\mathsf{lpp}(f) : f \in F \setminus \{0\}\}$*;*
7. $\mathsf{lc}(F) = \{\mathsf{lc}(f) : f \in F \setminus \{0\}\}$*;*
8. $\mathsf{in}(F) = \{\mathsf{in}(f) : f \in F \setminus \{0\}\}$*;*
9. $\mathsf{red}(F) = \{\mathsf{red}(f) : f \in F \setminus \{0\}\}$.

◁

3.2.9 Definition ([126, Definition 8.2.3])**.** *Any admissible ordering $<$ on $[X]$ induces a partial ordering $\ll$ on $K[X]$, the* induced ordering*, in the following way:*

$$f \ll g \Leftrightarrow (f = 0 \wedge g \neq 0) \vee \tag{3.25a}$$
$$(f \neq 0 \wedge g \neq 0 \wedge \mathsf{lpp}(f) < \mathsf{lpp}(g)) \vee \tag{3.25b}$$
$$(f \neq 0 \wedge g \neq 0 \wedge \mathsf{lpp}(f) = \mathsf{lpp}(g) \wedge \mathsf{red}(f) \ll \mathsf{red}(g))\,. \tag{3.25c}$$

◁

3.2.10 Lemma ([126, Lemma 8.2.4]). *$\gg$ is a Noetherian partial ordering on $K[X]$.* ◁

A pivotal notion in the theory of Gröbner bases is the concept of polynomial reduction (see Definition 3.2.11).

3.2.11 Definition ([126, Definition 8.2.4]). *Let $f, g, h \in K[X]$ and $F \subseteq K[X]$. We say that g* reduces to *h with regard to f (denoted as $g \longrightarrow_f h$) iff $\exists s, t \in [X]$ such that s has a non-vanishing coefficient c in g (i.e. $\mathsf{coeff}(g, s) = c \neq 0$), $s = \mathsf{lpp}(f) \cdot t$, and*

$$h = g - \frac{c}{\mathsf{lc}(f)} \cdot t \cdot f. \tag{3.26}$$

To be precise in notation and indicate which power product and coefficient are used in the reduction, we write

$$g \longrightarrow_{f,b,t} h, \text{ where } b = \frac{c}{\mathsf{lc}(f)}. \tag{3.27}$$

We say that g reduces to h with regard to *F (denoted as $g \longrightarrow_F h$) iff $\exists f \in F \colon g \longrightarrow_f h$.* ◁

3.2.12 Lemma ([126, Lemma 8.2.5]). *Let $\in K^\times, s \in [X], F \subseteq [X], g_1, g_2, h \in K[X]$.*

1. *$\longrightarrow_F \subseteq \gg$,*
2. *$\longrightarrow_F$ is Noetherian,*
3. *if $g_1 \longrightarrow_F g_2$, then $a \cdot s \cdot g_1 \longrightarrow_F a \cdot s \cdot g_2$,*
4. *if $g_1 \longrightarrow_F g_2$, then $g_1 + h \downarrow_F^* g_2 + h$.*

◁

3.2.13 Theorem ([126, Theorem 8.2.6]). *Let $F \subseteq K[X]$. The ideal congruence modulo $\langle F \rangle$ equals the reflexive-transitive-symmetric closure of $\to^F$, i.e., $\equiv_{\langle F \rangle} = \longleftrightarrow_F^*$.* ◁

3.2.14 Definition ([126, Definition 8.2.5]). *A subset F of $K[X]$ is a Gröbner basis (for $\langle F \rangle$) iff $\longrightarrow_F$ is Church-Rosser.* ◁

It should be noted that many generating sets of an ideal can have the property stated in Definition 3.2.14 and, as a result, are **a** Gröbner basis of the ideal.

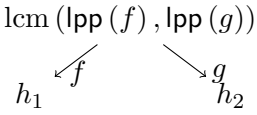

Figure 3.11: Critical pairs of elements of F describe exactly the essential branchings of the polynomial reduction relation $\to_F$.

3.3 Computation of Gröbner Bases

3.3.1 Definition ([126, Definition 8.3.3]). *Let $f, g \in K[X]$ and $t = \operatorname{lcm}(\mathsf{lpp}(f), \mathsf{lpp}(g))$. Then,*

$$cp(f,g) = \left(t - \frac{1}{\mathsf{lc}(f)} \cdot \frac{t}{\mathsf{lpp}(f)} \cdot f, t - \frac{1}{\mathsf{lc}(g)} \cdot \frac{t}{\mathsf{lpp}(g)} \cdot g\right) \tag{3.28}$$

is the critical pair *of f and g. The difference of the elements of $cp(f,g)$ is called the S-polynomial of f and g.* ◁

3.3.2 Remark ([126, Section 8.3]). *Figure 3.11 illustrates the situation graphically in the of $cp(f,g) = (h_1, h_2)$.* ◁

3.3.3 Theorem ([126, Theorem 8.3.1]). *Let $F \subseteq K[X]$.*

1. *(Buchberger's Theorem) F is a Gröbner basis iff $g_1 \downarrow_F^* g_2$ for all critical pairs (g_1, g_2) of elements of F.*
2. *F is a Gröbner basis iff $\forall f, g \in F\colon spol(f,g) \to_F^* 0$.*

◁

Proof. 1: If F is a Gröbner basis, then $g_1 \downarrow_F^* g_2$ for all critical pairs (g_1, g_2) of F. For the other direction, assume that $g_1 \downarrow_F^* g_2$ for all critical pairs (g_1, g_2) of F. By Theorem 3.1.9, it suffices to show that $h_1 \longleftrightarrow_{F(\ll h)}^* h_2$ for all h, h_1, h_2 such that $h_1{}_F \leftarrow h \to_F h_2$.

Let s_1, s_2 be the power products that are eliminated in the reduction of h to h_1 and h_2, respectively. That means, there are polynomials $f_1, f_2 \in F$, coefficients $c_1 = \mathsf{coeff}(h, s_1) \neq 0, c_2 = \mathsf{coeff}(h, s_2) \neq 0$, and power product t_1, t_2 such that

$$s_1 = t_1 \mathsf{lpp}(f_1), h_1 = h - \frac{c_1}{\mathsf{lc}(f_1)} t_1 f_1 \wedge s_2 = t_2 \mathsf{lpp}(f_2), h_2 = h - \frac{c_2}{\mathsf{lc}(f_2)} t_2 f_2. \tag{3.29}$$

We continue with a case distinction, depending on whether $s_1 = s_2$ holds.

$s_1 \neq s_2$**:** We can assume that $s_1 > s_2$, w.l.o.g.. Let $a = \mathsf{coeff}\left(-(c_1/\mathsf{lc}\left(f_1\right) t_1 f_1, s_2\right)$, then $\mathsf{coeff}\left(h_1, s_2\right) = c_2 + a$, leading to

$$h_1 \to_F h_1 - \frac{c_2 + a}{\mathsf{lc}\left(f_2\right)} t_2 f_2 = h - \frac{c_1}{\mathsf{lc}\left(f_1\right)} t_1 f_1 - \frac{c_2 + a}{\mathsf{lc}\left(f_2\right)} t_2 f_2. \tag{3.30}$$

We also have

$$h_2 \to_F h_2 - \frac{c_1}{\mathsf{lc}\left(f_1\right)} t_1 f_1 \to_F h_2 - \frac{c_1}{\mathsf{lc}\left(f_1\right)} t_1 f_1 - \frac{a}{\mathsf{lc}\left(f_2\right)} t_2 f_2 = \tag{3.31a}$$

$$h - \frac{c_1}{\mathsf{lc}\left(f_1\right)} t_1 f_1 - \frac{c_2 + a}{\mathsf{lc}\left(f_2\right)} t_2 f_2. \tag{3.31b}$$

It follows that $h_1 \longleftrightarrow^*_{F(\ll h)} h_2$, and in fact $h_1 \downarrow^*_F h_2$.

$s_1 = s_2$**:** Let $s = s_1 = s_2, c = \mathsf{coeff}\left(h, s\right)$ and $h' = h - cs$. There exists a power product t, such that

$$s = t \operatorname{lcm}\left(\mathsf{lpp}\left(f_1\right), \mathsf{lpp}\left(f_2\right)\right) \wedge h_1 = h' + ctg_1, h_2 = h' + ctg_2, \tag{3.32}$$

where $(g_1, g_2) = cp\left(f_1, f_2\right)$. By assumption $g_1 \downarrow^*_F g_2$, i.e.,, there exist $p_1, \ldots, p_k, q_1, \ldots, q_l$ such that

$$g_1 = p_1 \to_F \cdots \to_F p_k = q_{l\,F} \leftarrow \cdots {}_F \leftarrow q_1 = g_2. \tag{3.33}$$

By Lemma 3.2.12, 3, it follows that

$$ctg_1 = ctp_1 \to_F \cdots \to_F ctp_k = ctq_{l\,F} \leftarrow \cdots {}_F \leftarrow ctq_1 = ctg_2. \tag{3.34}$$

By Lemma 3.2.12, 4, it follows that

$$h_1 = h' + ctp_1 \downarrow^*_F \cdots \downarrow^*_F h' + ctp_k = h' + ctq_l \downarrow^*_F \cdots \downarrow^*_F h' + ctq_1 = h_2. \tag{3.35}$$

All the intermediate polynomials in these reductions are less than h with regard to $\ll$. Thus, $h_1 \longleftrightarrow^*_{F(\ll)} h_2$.

2: All S-polynomial are congruent to zero modulo $\langle F \rangle$ and Theorem 3.2.13 yields $spol\left(f, g\right) \to^*_F 0$. If F is a Gröbner basis, then $spol\left(f, g\right) \to^*_F 0$. For the other direction, assume that $spol\left(f, g\right) \to^*_F 0, \forall f, g \in F$. We not only continue with the same notation as in 1, but also observe that the whole proof is analogous, except for the case $s_1 = s_2 = s$. Therefore, for $h_1 = h' + ctg_{1\,F} \leftarrow h \to_F h' + ctg_2 = h_2$, we have to show $h_1 \longleftrightarrow^*_{F(\ll h)} h_2$. $g_1 - g_2$ is the S-polynomial of $f_1, f_2 \in F$, so by our assumption $g_1 - g_2 \to^*_F 0$. An

application of Lemma 3.2.12 yields $h_1 - h_2 = ct\,(g_1 - g_2) \to_F^* 0$, meaning there exist $p_1, \ldots, p_k$ such that

$$h_1 - h_2 = p_1 \to_F \cdots \to_F p_k = 0. \tag{3.36}$$

Another application of Lemma 3.2.12 yields

$$h_1 = p_1 + h_2 \to_F^* \cdots \to_F^* p_k + h_2 = h_2, \tag{3.37}$$

which implies $h_1 \longleftrightarrow_{F(\ll h)}^* h_2$. □

An algorithm[1] for computing Gröbner bases based on Buchberger's theorem (Theorem 3.3.3, 1) is given in Algorithm 1.

Algorithm 1 Buchberger's algorithm.

procedure GB-BB: (returns a finite generator G of I, which is a Gröbner basis for I.)
Require: Finite generator F of an ideal I.
  $G \leftarrow F$
  $C \leftarrow \{\{g_1, g_2\} \colon g_1, g_2 \in G, g_1 \neq g_2\}$
  while not all pairs $\{g_1, g_2\}$ are marked **do**
    Choose an unmarked pair $\{g_1, g_2\}$
    Mark $\{g_1, g_2\}$
    $h \leftarrow$ normal form of $spol\,(g_1, g_2)$ w.r.t. $\to_G$
    if $h \neq 0$ **then**
      $C \leftarrow C \cup \{\{g, h\} \colon g \in G\}$
      $G \leftarrow G \cup \{h\}$
    end if
  end while
  return G
end procedure

3.3.4 Theorem ([126, Theorem 8.3.2])**.** *(Dickson's lemma) Every $A \subseteq [X]$ contains a finite subset B, such that every $t \in A$ is a multiple of some $s \in B$.* ◁

Proof. The proof uses induction on the number of variables in $[X]$. The case $n = 1$ trivially holds. Therefore, assume that $n \geq 2$. Choose

$$A \ni t_0 = x_1^{e_1} \cdots x_n^{e_n}. \tag{3.38}$$

[1] At this point, we already call the computational steps given in Algorithm 1 in their unity an *algorithm*, although this terminology will only be justified by Proposition 3.3.5.

For $i \in \{1, \ldots, n\}$ and $j \in \{0, \cdots, e_i\}$, consider the set of power products

$$A_{i,j} = \{t \in [X] : t \in A \wedge \underset{x_i}{deg}\,(t) = j\}, \tag{3.39}$$

and

$$A'_{i,j} = \{t/x_i^j : t \in A_{i,j}\}. \tag{3.40}$$

The variable x_i does not occur – by construction of the set $A'_{i,j}$ – in the elements of $A'_{i,j}$ any more. By the induction hypothesis, there exist finite subsets $B'_{i,j} \subseteq A'_{i,j}$ such that every power product in $A'_{i,j}$ is a multiple of some power product in $B'_{i,j}$. Let

$$B_{i,j} = \{t \cdot x_i^j : t \in B'_{i,j}\}. \tag{3.41}$$

It follows that every element of A is a multiple of some element of the finite set

$$B = \{t_0\} \cup \bigcup_{i,j} B_{i,j} \subseteq A. \tag{3.42}$$

□

3.3.5 Proposition. *The computational steps given in Algorithm 1 indeed form an algorithm, in particular it terminates for any valid input with a Gröbner basis.* ◁

Proof. The polynomials h constructed in Algorithm[2] 1 in line 7 are all elements of $\langle F \rangle$, therefore all throughout the execution of the steps in Algorithm[3] 1 it holds that $\langle G \rangle = \langle F \rangle$.

Regarding the termination of the computational steps given in Algorithm[4] 1, we note that Theorem 3.3.4 implies that in $[X]$ there is no infinite chain of elements $s_1, s_2, \ldots$ such that $s_i \nmid s_j$, for all $1 \leq i < j$. The leading power product of the polynomials added to the basis form such a sequence in $[X]$, hence this sequence must be finite.

It is now established that the computational steps given in Algorithm[5] 1 form indeed a proper algorithm[6] and it follows by Buchberger's Theorem, Theorem 3.1.5, 1, that the output of Algorithm 1 is indeed a Gröbner basis for $\langle F \rangle$. □

3.3.6 Theorem ([126, Theorem 8.3.3])**.** *Every ideal I in $K[X]$ has a Gröbner basis.* ◁

[2]For the used terminology in this reference, see Footnote 1 on page 31.
[3]For the used terminology in this reference, see Footnote 1 on page 31.
[4]For the used terminology in this reference, see Footnote 1 on page 31.
[5]For the used terminology in this reference, see Footnote 1 on page 31.
[6]From this point onwards, the terminology used – the word algorithm – is justified.

Proof. A constructive proof is given by Algorithm 1. □

There are many characterizations of Gröbner bases, however, we only list a few of them in Theorem 3.3.7.

3.3.7 Theorem ([126, Theorem 8.3.4])**.** *Let I be an ideal in $K[X], F \subseteq K[X]$ and $\langle F \rangle = I$. Then, the following statements are equivalent.*

1. *F is a Gröbner basis for I;*
2. *$f \to_F^* 0, \forall f \in I$;*
3. *$f \to_F, \forall f \in I \setminus \{0\}$;*
4. *$\forall g \in I, h \in K[X]\ (g \to_F^* \underline{h} \implies h = 0)$;*
5. *$\forall g, h_1, h_2 \in K[X]\ ((g \to_F^* \underline{h}_1 \wedge g \to_F^* \underline{h}_2) \implies h_1 = h_2)$;*
6. *$\langle \mathit{in}(F) \rangle = \langle \mathit{in}(I) \rangle$.*

◁

3.3.8 Theorem ([126, Definition 8.3.5])**.** *Let G be a Gröbner basis for an ideal I in $K[X]$, $g, h \in G, g \neq h$ and $h' \in K[X]$.*

1. *If $\mathit{lpp}(g) \mid \mathit{lpp}(h)$, then $G' = G \setminus \{h\}$ is also a Gröbner basis for I.*
2. *If $h \to_g h'$, then $G' = (G \setminus \{h\}) \cup \{h'\}$ is also a Gröbner basis for I.*

◁

Proof. 1: Clearly, we have $\langle G' \rangle \subseteq I$. For $f \in I$, we have $f \to_G^* 0$, but in fact we have $f \to_{G'}^* 0$, because whenever we could reduce by h we can instead also reduce by g.
2: $\langle G' \rangle = \langle G \rangle$. If $\mathsf{lpp}(h)$ is reduced, then the result follows from 1. Otherwise, $\langle \mathsf{in}(G') \rangle = \langle G \rangle = \langle \mathsf{in}(I) \rangle$. □

3.3.9 Definition.

- *G is* minimal *iff $\mathit{lpp}(g) \nmid \mathit{lpp}(h), \forall g, h \in G, g \neq h$.*
- *G is* reduced *iff for all $g, h \in G$ with $g \neq h$ we cannot reduce h by g.*

- G *is* normed *iff* $\forall g \in G\colon \mathsf{lc}(g) = 1$.

◁

3.3.10 Theorem ([126, Theorem 8.3.6])**.** *Every ideal in* $K[X]$ *has a unique finite normed reduced Gröbner basis.* ◁

Proof. The existence of such a basis follows from Theorem 3.3.8. Assume that

$$G = \{g_1, \ldots, g_m\}, G' = \{g'_1, \ldots, g'_{m'}\} \tag{3.43}$$

are two normed reduced Gröbner basis for the ideal I. By the assumptions on G', it holds that $g_1 \to^*_{G'} 0$, in particular $\mathsf{lpp}(g_1)$ can be reduced by some polynomial in G', w.l.o.g. let g'_1 be this polynomial, i.e., $\mathsf{lpp}(g'_1) \mid \mathsf{lpp}(g_1)$. Also, by assumption on G, it holds that $g'_1 \to^0_G 0$ and therefore there exists k with $1 \leq k \leq m$ such that $\mathsf{lpp}(g_k) \mid \mathsf{lpp}(g'_1)$. By assumption G is reduced, yielding $k = 1$, i.e., $\mathsf{lpp}(g_1) = \mathsf{lpp}(g'_1)$. Continuing in this way and possibly after a reordering of the elements of G' we obtain $m = m'$ and $\mathsf{lpp}(g_i) = \mathsf{lpp}(g'_i)$ for all $1 \leq i \leq m$.
Let $i \in \{1, \ldots, m\}$, then $g_i \to^*_{G'} 0$. Assume, towards a contradiction, that $g_i \neq g'_i$. The only way to eliminate $\mathsf{lpp}(g_i)$ is to use g'_i. We have $g_i - g'_i \neq 0$, however, none of the power products in $g_i - g'_i$ can be reduced modulo G', contradiction. Hence, $\forall 1 \leq i \leq m\colon g_i = g'_i$. □

3.3.11 Remark ([126, Section 8.3])**.** *We note that the unique finite normed reduced Gröbner basis of an ideal given in Theorem 3.3.10 depends on the chosen admissible ordering. Different orderings can result in different Gröbner bases. This lead to the consideration of* universal *Gröbner bases [124, 95].* ◁

3.4 Polynomial System Solving

Polynomial system solving includes the problems and ways to solve systems of multivariate polynomial equations. The problem of polynomial system solving is defined as follows for some field K:

$$f_1(x_1, \ldots, x_n) = 0, \tag{3.44a}$$
$$f_2(x_1, \ldots, x_n) = 0, \tag{3.44b}$$
$$\vdots \tag{3.44c}$$
$$f_m(x_1, \ldots, x_n) = 0, \tag{3.44d}$$

where $f_1, f_2, \ldots, f_m \in K[X]$ and we seek – if any exist – common solutions (i.e, zeros) of the system given in (3.44a) - (3.44d). Let $I = \langle f_1, \ldots, f_m \rangle$.

3.4.1 Theorem ([126, Theorem 8.4.2]). *(Hilbert's Nullstellensatz) Let I be an ideal in $K[X]$, where K is an algebraically closed field. Then, the radical of I consists of exactly those polynomials in $K[X]$, which vanish on all common roots of I.* ◁

In [15], the following result was given:

3.4.2 Theorem ([126, Theorem 8.4.3]). *Let G be a normed Gröbner basis of I. The system given in* (3.44),(3.44a) - (3.44d) *is unsolvable in $\bar{K}$ if and only if $1 \in G$.* ◁

Proof. If $1 \in G$, then $1 \in \langle G \rangle = I$, so every solution of the system given in (3.44) is also a solution of $1 = 0$, yielding that no solutions exist.
For the other direction, assume that the system given in (3.44) is unsolvable. Then, the polynomial 1 vanishes on every common root of (3.44). So, by Theorem 3.4.1, $1 \in \mathit{radical}\,(I)$ and therefore also $1 \in I$. Since G is a normed Gröbner basis of I, it holds that $1 \rightarrow_G 0$, but this is only possible if $1 \in G$. □

Although we are only interested in solutions over the binary field – necessarily resulting in only finitely many solutions – we also mention an important result, namely that a Gröbner basis can be used to determine whether or not the corresponding ideal is zero-dimensional.

3.4.3 Theorem ([126, Theorem 8.4.4]). *Let G be a Gröbner basis of I. Then,* (3.44) *has finitely many solutions (i.e., I is zero-dimensional) if and only if for every i, $1 \leq i \leq n$, there is a polynomial $g_i \in G$ such that $lpp(g_i)$ is a pure power of x_i. Moreover, if I is zero-dimensional then the number of zeros of I (counted with multiplicities) is equal to $dim(K[X]/I)$.* ◁

The part that triangulation plays in solving systems of linear equations is in the case of multivariate polynomial system solving the role of the Gröbner basis algorithm. Gröbner bases with the lexicographic ordering provide the means to derive the solutions of such systems, a fact that was first stated in [120]. Some of the necessary conditions have been collected jointly and can be found in the literature under the name "elimination orderings", however, we restrict ourselves to the case of the lexicographic ordering given

in the next theorem (Theorem 3.4.4), which states the *elimination property of Gröbner bases.*

3.4.4 Theorem ([126, Theorem 8.4.5])**.** *Get G be a Gröbner basis of I w.r.t. the lexicographic ordering $x_1 < \cdots < x_n$. Then*

$$I \cap K[x_1, \ldots, x_i] = \langle G \cap K[x_1, \ldots, x_i] \rangle, \tag{3.45}$$

where the ideal on the right-hand side is generated over the ring $K[x_1, \ldots, x_i]$. ◁

Proof. The right hand side is obviously contained in the left-hand side.
For the other inclusion, let $f \in K[x_1, \ldots, x_n]$ and obviously $f \to_G^* 0$. All polynomials occurring in this reduction depend only on variables in the set $\{x_1, \ldots, x_i\}$, resulting in a representation of f as a linear combination $\sum_j h_j g_j$, where $g_j \in G \cap K[x_1, \ldots, x_i]$ and $h_j \in K[x_1, \ldots, x_i]$. □

The elimination property, in the form given in Theorem 3.4.4, yields that a Gröbner basis with regard to the lexicographic ordering of a zero-dimensional ideal has always the following structure:

$$\begin{cases} g_1(x_1) & = 0, \\ g_2(x_1, x_2) & = 0, \\ & \vdots \end{cases} \tag{3.46}$$

Once a Gröbner basis with regard to the lexicographic ordering has been computed for a polynomial system and if we assume that we can compute the solutions of all univariate polynomials over the algebraic closure of the underlying field, then we can successively eliminate variables by computing the solutions for the resulting univariate polynomials and back-substituting, propagating partial solutions until we have solved the entire system.

CHAPTER 4

Applications of Computer Algebra to Design Theory

We apply algebraic techniques with the goal to reason about combinatorial designs in an algebraic context. The algebraic approach presented is built upon commutative algebra and the corresponding methodologies are developed within a linear algebra setting and also employ symbolic computation techniques for constructive problems. We translate combinatorial properties of tuples into an algebraic formulation and this will serve as the building block to derive an algebraic framework for the study of certain combinatorial structures, most importantly, covering arrays. In particular, in this chapter, we show how to model, characterize and construct binary strength two covering arrays within an algebraic framework.

A brief, informal description of our approach for constructing *covering arrays* can be given as follows: for given covering array configuration $C = (2, k, 2)$, and depending on the covering array problem considered, we construct a matrix, where some entries are variables x_i from a suitable multivariate polynomial ring over the field of rational numbers $\mathbb{Q}$. The concept of row-selectors with specific entries and transformations of coverage conditions into an algebraic formulation are the means to arrive at a multivariate polynomial system of equations over a specific multivariate polynomial ring. Subsequently, we rely on the theory of Gröbner bases to compute the corresponding variety. In the case that there are solutions, each point in the computed variety corresponds to a matrix, which constitutes a covering array for the given configuration C.

We develop fundamental algebraic distinguishers for binary pairs in Section 4.1, which are used in different contexts to derive characterizations of combinatorial designs in Section 4.2. In Section 4.3 we turn to constructive design problems and show how they can be tackled using a computational algebra formalism.

4.1 Algebraic Distinguishers for multiple existentially-quantified Combinatorial Designs

The kind of equations, which will be derived, model the logical `or` operator (in the "not exclusively" sense) applied to existential conditions. The main theoretical focus of this Thesis is on the translation of these logical statements into an algebraic context.

Algebraic Tuple Modelling with Coverage Equations

We will derive algebraic means to decide about the appearance of binary pairs for certain submatrix selections of a given matrix. In particular, we establish a connection between existential conditions in combinatorial design theory to zeros of multivariate polynomials in specific rings.

4.1.1 Definition ([40, Definition 4])**.** *Let P be a ring and $a, b \in P$. We say that the triple (P, a, b) has the* pairwise binary tuple distinguishing property*, if and only if,*

1. *P is a unary ring;*
2. *P is an integral domain;*
3. *The elements $0, a, b$ and $a + b$ are pairwise different.*

◁

4.1.2 Terminology. *A distinguisher for a set A is an injective function with domain A. The image of this function is chosen according to individual requirements of the problem of interest.* ◁

We begin with the inference of an "algebraic distinguisher" for binary pairs. The first observation is a (trivial) computation.

4.1.3 Proposition. *[40, Remark 1] Assume that P is a ring, $a, b \in P$, (P, a, b) has the pairwise binary tuple distinguishing property, and $\mathcal{T} = \{\begin{pmatrix}0, & 0\end{pmatrix}, \begin{pmatrix}1, & 0\end{pmatrix}, \begin{pmatrix}0, & 1\end{pmatrix}, \begin{pmatrix}1, & 1\end{pmatrix}\}$. Then, the function $\varphi\colon \mathcal{T} \longrightarrow \{0, 1, a, b\}$, given by*

$$\varphi(\begin{pmatrix}0, & 0\end{pmatrix}) = 0,\ \varphi(\begin{pmatrix}1, & 0\end{pmatrix}) = a,\ \varphi(\begin{pmatrix}0, & 1\end{pmatrix}) = b,\ \varphi(\begin{pmatrix}1, & 1\end{pmatrix}) = a + b, \tag{4.1}$$

is a bijection. Furthermore, it can be expressed as a matrix matrix multiplication[1] *as follows:*

$$\begin{pmatrix}a, & b\end{pmatrix} \cdot \begin{pmatrix}0, & 0\end{pmatrix}^{\mathsf{T}} = 0 = \varphi(\begin{pmatrix}0, & 0\end{pmatrix}), \tag{4.2a}$$
$$\begin{pmatrix}a, & b\end{pmatrix} \cdot \begin{pmatrix}1, & 0\end{pmatrix}^{\mathsf{T}} = a = \varphi(\begin{pmatrix}1, & 0\end{pmatrix}), \tag{4.2b}$$
$$\begin{pmatrix}a, & b\end{pmatrix} \cdot \begin{pmatrix}0, & 1\end{pmatrix}^{\mathsf{T}} = b = \varphi(\begin{pmatrix}0, & 1\end{pmatrix}), \tag{4.2c}$$
$$\begin{pmatrix}a, & b\end{pmatrix} \cdot \begin{pmatrix}1, & 1\end{pmatrix}^{\mathsf{T}} = a + b = \varphi(\begin{pmatrix}1, & 1\end{pmatrix}). \tag{4.2d}$$

◁

Proof. From the assumption that (P, a, b) has the pairwise binary tuple distinguishing property follows that φ is a bijection. The claimed representation of φ as a matrix matrix multiplication follows from the computations in equations (4.2a) – (4.2d). □

4.1.4 Remark. *[40, Remark 1] From Corollary 4.1.3 it follows that we can use the evaluation of a matrix matrix multiplication of a tuple having only zero and one as elements with the "symbolic" vector $\begin{pmatrix}a, & b\end{pmatrix}$ to determine a tuple from the set $\mathcal{T}$ uniquely. We state this result explicitly in the following Lemma 4.1.5.* ◁

4.1.5 Lemma. *[40, Lemma 1] Assume that P is a ring, $a, b \in P$, (P, a, b) has the pairwise binary tuple distinguishing property, let $\mathcal{T} = \{\begin{pmatrix}0, & 0\end{pmatrix}, \begin{pmatrix}1, & 0\end{pmatrix}, \begin{pmatrix}0, & 1\end{pmatrix}, \begin{pmatrix}1, & 1\end{pmatrix}\}$ and $\begin{pmatrix}t_1, & t_2\end{pmatrix} \in \mathcal{T}$. Then, the equivalences in equations* (4.3a), (4.3b), (4.3c) *and* (4.3d) *hold.*

$$\begin{pmatrix}t_1, & t_2\end{pmatrix} = \begin{pmatrix}0, & 0\end{pmatrix} \iff \begin{pmatrix}a, & b\end{pmatrix} \cdot \begin{pmatrix}t_1, & t_2\end{pmatrix}^{\mathsf{T}} = 0, \tag{4.3a}$$
$$\begin{pmatrix}t_1, & t_2\end{pmatrix} = \begin{pmatrix}1, & 0\end{pmatrix} \iff \begin{pmatrix}a, & b\end{pmatrix} \cdot \begin{pmatrix}t_1, & t_2\end{pmatrix}^{\mathsf{T}} - a = 0, \tag{4.3b}$$
$$\begin{pmatrix}t_1, & t_2\end{pmatrix} = \begin{pmatrix}0, & 1\end{pmatrix} \iff \begin{pmatrix}a, & b\end{pmatrix} \cdot \begin{pmatrix}t_1, & t_2\end{pmatrix}^{\mathsf{T}} - b = 0, \tag{4.3c}$$
$$\begin{pmatrix}t_1, & t_2\end{pmatrix} = \begin{pmatrix}1, & 1\end{pmatrix} \iff \begin{pmatrix}a, & b\end{pmatrix} \cdot \begin{pmatrix}t_1, & t_2\end{pmatrix}^{\mathsf{T}} - a - b = 0. \tag{4.3d}$$

[1] We regard pairs as 2×1 matrices.

◁

Proof. For all four cases, the direction "⇒" follows from the computations in Proposition 4.1.3 and the direction "⇐" from the fact that the function φ is by Proposition 4.1.3 a bijection. □

It is well known, that given a nonempty finite product of elements of an integral domain, the product is zero if and only if at least one factor is zero[30, Appendix A, Definition 3]. This paves the way for making a connection between the requirement of "appearance at least once" in the coverage conditions in the theory of covering arrays and zeros of nonempty finite products of elements in certain integral domains (multivariate polynomial rings).

To extend the technique presented so far to being capable to reason about matrices, it is necessary to introduce "row selectors" as a means to transform the statement "for any selection of t distinct rows" into our algebraic approach of linear operations.

4.1.6 Remark. *[40, Remark 2] The definition of covering arrays, Definition 2.1.1, requires coverage of all t-tuples for all $N \times t$ subarrays. In the sequel, we will (sometimes) work with transposed matrices and will therefore be interested in coverage properties of selections of t distinct rows, i.e., $t \times N$ subarrays of the transposed matrix.* ◁

4.1.7 Definition. *[40, Definition 5] For $k, i, j \in \mathbb{N}, i, j \geq 1, k \geq 2$, let the function*

$$e^{k,i,j} : R \times R \longrightarrow R^{1\times k} \tag{4.4}$$

map a pair of elements from a ring R to a row vector of length k, where the first component is mapped to the i-th position, the second component to the j-th position in the vector, and all other entries in the vector are zero. ◁

4.1.8 Example. *[40, Example 1] We will be particular interested in $e^{k,i,j}(a, b)$, for example,*

$$e^{6,2,5}(a, b) = (0, a, 0, 0, b, 0). \tag{4.5}$$

◁

4.1.9 Remark. *[40, Remark 3] Let P be a ring and $\epsilon_i, \epsilon_j \in P$. For any matrix M defined over P with k rows and given $1 \leq i < j \leq k$, let the matrix $\widetilde{M}$ consist of the vertical concatenation of the i-th and j-th row in this order of the matrix M. Then,*

$$e^{k,i,j}(\epsilon_i, \epsilon_j)\, M = \begin{pmatrix} \epsilon_i, & \epsilon_j \end{pmatrix} \widetilde{M}. \tag{4.6}$$

In equation (4.6)*, both sides of the equality result from matrix matrix multiplications.* ◁

4.1.10 Theorem ([40, Theorem 1])**.** *Assume that P is a ring and that (P, a, b) has the pairwise binary tuple distinguishing property. For any given $k \times N$ matrix M defined over P containing only zero and one as entries, and any $1 \leq i < j \leq k$, let $\widetilde{M}$ denote the vertical concatenation of the i-th and j-th row in this order of the matrix M. Let s_a denote the $1 \times N$ row vector with all components equal to a, and let*

$$h = (h_\ell)_{1 \leq \ell \leq N} = e^{k,i,j}(a, b)M - s_a. \tag{4.7}$$

Then, the following statements are equivalent:

(i) The tuple $\left(1, 0\right)^\intercal$ appears at least once as a column in the matrix $\widetilde{M}$.

(ii) The vector h contains at least one component equal to zero.

(iii) $\prod_{\ell=1}^{N} h_\ell = 0$.

A similar statement holds for the tuples $\left(0, 0\right)^\intercal, \left(0, 1\right)^\intercal, \left(1, 1\right)^\intercal$. ◁

Proof. The equivalence of (i) and (ii) follows from Lemma 4.1.5. The equivalence of (ii) and (iii) follows from the defining property of an integral domain. □

4.1.11 Notation ([40, Definition 6])**.** *We call the equations appearing in Theorem 4.1.10, (iii),* coverage equations *and using the notation from Theorem 4.1.10, define the following notation for $1 \leq i < j \leq k, \tau \in \{\left(0, \quad 0\right)^\top, \left(1, \quad 0\right)^\top, \left(0, \quad 1\right)^\top, \left(1, \quad 1\right)^\top\}$:*

$$coveq_\tau^{(i,j)}(M) = \prod_{\ell=1}^{N} h_\ell. \tag{4.8}$$

◁

4.1.12 Remark. *[40, Remark 4] The coverage equations are modelled after the coverage conditions appearing in the definition of covering arrays. The coverage equations are formulated in such a way that they are semantically equivalent to the pairwise coverage conditions for binary covering arrays. This statement is the main result of [40] and will be formulated in Theorem 4.2.3.* ◁

Membership Equations for covering Tuples in CAs

We will derive algebraic means to enforce the appearance of some value pair out of a list of pre-determined choices for two selected positions of a matrix. Note that this makes it possible to impose *more structure* on the array than what can be achieved with only modelling of existential quantifiers (cf. Section 2.2.1).

4.1.13 Theorem ([40, Theorem 2.1])**.** *Assume that P is a ring, $a, b \in P$, and that (P, a, b) has the pairwise binary tuple distinguishing property, $\begin{pmatrix} t_1, & t_2 \end{pmatrix}$ is a tuple with $t_i \in \{0, 1\}, i = 1, 2$, $\mathcal{T} = \{\begin{pmatrix} 0, & 0 \end{pmatrix}, \begin{pmatrix} 1, & 0 \end{pmatrix}, \begin{pmatrix} 0, & 1 \end{pmatrix}, \begin{pmatrix} 1, & 1 \end{pmatrix}\}$, and let $\mathcal{S}$ be a subset of $\mathcal{T}$, i.e. $\mathcal{S} \subseteq \mathcal{T}$.*

Then, the following statements (1)*,* (2) *and* (3) *are equivalent.*

1.
$$\begin{pmatrix} t_1, & t_2 \end{pmatrix} \in \mathcal{S}; \tag{4.9}$$

2.
$$0 \in \{\begin{pmatrix} a, & b \end{pmatrix} \begin{pmatrix} t_1, & t_2 \end{pmatrix}^\top - \varphi(\tau) \mid \tau \in \mathcal{S}\}; \tag{4.10}$$

3.
$$\prod_{\tau \in \mathcal{S}} \left(\begin{pmatrix} a, & b \end{pmatrix} \begin{pmatrix} t_1, & t_2 \end{pmatrix}^\top - \varphi(\tau) \right) = 0. \tag{4.11}$$

◁

Proof. First, assume that $\mathcal{S} \neq \emptyset$. The equivalence of (1) and (2) follows from the fact that the function φ is a bijection. The equivalence of (2) and (3) follows from the defining property of an integral domain. Second, assume that $\mathcal{S}$ is empty. Then, the claimed equivalences trivially hold. □

4.1.14 Remark ([40, Remark 2.2])**.** *An algebraic distinguisher for* 1*-tuples; i.e., for individual elements, under the same conditions and with similar properties as given in Theorem 4.1.13 for pairs, can be constructed .* ◁

4.1.15 Definition ([40, Definition 2.3])**.** *For $k, N \in \mathbb{N}$ with $2 \leq k$, let $C = (2, k, 2)$ be a configuration and M a compatible $N \times k$ matrix. We call*[2,3] *the function $\Gamma \colon [N] \times \mathcal{I}_k \longrightarrow \mathcal{P}(\mathcal{T})$ an* interaction-membership function*.* ◁

[2] For $k \in \mathbb{N}$, let $\mathcal{I}_k = \{(i, j) \mid 1 \leq i < j \leq k\} \subseteq [k] \times [k]$.
[3] For a set S, we denote with $\mathcal{P}(S)$ its power set.

4.1.16 Remark ([40, Definition 2.3 and Remark 2.4])**.** *An interaction-membership function is interpreted as assigning, to each unordered selection of two different columns per row, a set of binary pairs that are allowed to appear at this position. Note that such a function may specify contradicting conditions. Note that the non-appearance of certain value pairs at a position can be specified by considering their complement with respect to the set $\mathcal{T}$ and encoding this condition within an interaction-membership function.* ◁

Combined Models

So far, we have seen two different approaches, which demand certain structural properties of arrays, namely:

1. Existence of pairs in subarrays in the sense of coverage conditions (see Definition 2.2.11),
2. Constraints on the appearance of pairs of elements from an array in the sense of an interaction-membership function (see Definition 4.1.15).

It is, of course, possible to combine these two requirements and consider them together. The resulting structures – if they exist – will satisfy both requirements. The joint consideration of these two structural properties has been motivated by and continues to have an impact in real-world applications of combinatorial designs in software testing (see Chapter 6).

4.2 Algebraic Characterizations for specific Design Structures

Using Theorem 4.1.10, it is now possible to translate coverage-conditions and membership requirements into an algebraic setting.

Partial Coverage Systems

The next corollary (Corollary 4.2.1) establishes this connection in a very general setting for the notion of partial coverage systems.

4.2.1 Corollary. *For given partial coverage system and compatible matrix M containing only zero and one as entries, there exists a system of multivariate polynomial equations $\mathfrak{S}$,*

such that M fulfills the conditions specified by the partial coverage system iff its elements satisfy $\mathfrak{S}$. ◁

Proof. Analogously to the proof of Corollary 4.2.3. □

4.2.2 Remark. *Observe that Corollary 4.2.1 holds, for example, for the special case of covering arrays on graphs.* ◁

Covering Arrays

We highlight an important special instance of Corollary 4.2.1 for the specific case of *covering arrays* in Corollary 4.2.3 explicitly.

4.2.3 Corollary ([40, Corollary 1])**.** *Assume that P is a ring, $a, b \in P$, and that (P, a, b) has the pairwise binary tuple distinguishing property. Let M be a $k \times N$ matrix with $2 \leq k$ defined over P, containing only zero and one as entries.*

Then, the statements 1, 2 and 3 are equivalent.

1. *For every selection of two different rows of M, each possible binary tuple appears at least once as a column of the selected $2 \times N$ submatrix of M.*
2. *$M^\top$ is a covering array for the configuration $(2, k, 2)$ in the sense of Definition 2.1.1, i.e., the strength two coverage conditions of covering arrays hold for $M^\top$.*
3. *For every i and j with the property $1 \leq i < j \leq k$ and for all $\tau \in \{\begin{pmatrix}0, & 0\end{pmatrix}^\top, \begin{pmatrix}1, & 0\end{pmatrix}^\top, \begin{pmatrix}0, & 1\end{pmatrix}^\top, \begin{pmatrix}1, & 1\end{pmatrix}^\top\}$:*

$$\mathrm{coveq}_\tau^{(i,j)}(M) = 0. \tag{4.12}$$

◁

Proof. The equivalence of 1 and 2 follows from Definition 2.1.1. The equivalence of 1 and 3 follows from Theorem 4.1.10. □

Membership Constraints

4.2.4 Corollary ([40, Corollary 2.5])**.** *For $k, N \in \mathbb{N}$ with $2 \leq k$, let $C = (2, k, 2)$ be a configuration, M a compatible $N \times k$ matrix and Γ an interaction-membership function. Then, there exists a system of multivariate polynomial equations $\mathcal{E}$, such that M fulfills the conditions given by Γ if and only if its elements satisfy $\mathcal{E}$. We call these equations membership-equations.* ◁

Proof. By Theorem 4.1.13, the condition expressed by every element in $ran(\Gamma)$ holds if and only if the respective values satisfy the corresponding equation (4.11). Denote by $\mathcal{E}$ the set of all these equations, and the statement follows. □

Combined Models

4.2.5 Corollary. *Let combined requirements in the sense of Section 4.1 and a compatible matrix M containing only zero and one as entries be given. Then, the following statements are equivalent.*

1. *M fulfills all combined conditions (i.e., coverage-conditions and all requirements of the interaction-membership function).*
2. *M fulfills the system consisting of the respective coverage-equations and membership-equations.*

◁

Proof. Follows from Corollary 4.2.3 and Corollary 4.3.4. □

4.3 Constructive Design Theory with Polynomial System Solving

Based on the algebraic modelling presented in Sections 4.1 and 4.2, we are now in a position to investigate existential and constructive problems pertaining the design structures of interest in an algebraic setting using methods from symbolic computation. In particular, through the presented *algebraic modelling* the problems of constructing and computing covering arrays will be formulated as instances of algebraic systems, where each solution of them corresponds to a design matrix [40].

4.3.1 Candidate Matrices

We turn now to matrices defined over some appropriate multivariate polynomial ring. Specifically for constructive problems, the goal is to derive a *semantically equivalent* system of multivariate polynomial equations in the sense that any solution – if it exists – will have all the properties required in the initial constructive design problem. To this end, we consider matrices where at least one element is equal to an indeterminate from the underlying polynomial ring. When the corresponding systems of multivariate polynomial equations has a solution, any solution can be used to *instantiate* the original matrix, when the values of the solution are substituted into the matrix in the corresponding positions resulting in a design structure, which exhibits all the initially required properties [40].

Depending on the problem under consideration, we obtain a matrix in which in some entries variables x_i appear. Assume that there are exactly $\gamma \in \mathbb{N}$ variables appearing in the matrix. In order to apply our distinguisher-based approach, we will regard this matrix as being defined over the multivariate polynomial ring $\mathbb{Q}[x_1, x_2, \ldots, x_\gamma, a, b]$ of rank $\gamma + 2$. A detailed description of the used polynomial ring is given in the next section. In the next lemma (Lemma 4.3.1), we prove that the results of the previous section do hold for the case of candidate matrices:

4.3.1 Lemma. *[40, Lemma 2] Let P be a multivariate polynomial ring in at least two variables defined over the rational field and assume that a and b are two different indeterminates, then (P, a, b) has the pairwise binary tuple distinguishing property.* ◁

Proof. The requirements for the pairwise binary tuple distinguishing property hold due to the properties of P. □

4.3.2 Types of Equations

We discuss three types of equations, which model/enforce different kinds of properties that we are interested in. First, we consider *binary conditions*, which are used to restrict the values that a variable can take. Second, we consider *coverage equations*, which are used to enforce the appearance of tuples and third, we consider *membership-equations* as a means to restrict what kind of tuples may appear at certain pairwise selections of matrix positions.

Binary Conditions

We are interested in binary covering arrays, therefore we have to ensure that all entries in the considered matrices are either zero or one. For all variables x_i in a matrix, we enforce the binary condition via an equation of the form ([40, Section 3.1]):

$$x_i\,(x_i - 1)\,. \tag{4.13}$$

Necessary binary conditions will be added to all considered systems of multivariate polynomial equations.

Coverage-equations

By Theorem 4.1.10, coverage-equations can be used to enforce coverage-conditions. Depending on the considered structure (partial coverage system or *covering array*), the corresponding coverage-equations are added to the system (cf. Corollary 4.2.3).

Membership-equations

By Corollary 4.2.4, the fulfillness of the requirements specified by an interaction-membership function is equivalent to being a solution of a polynomial system. Therefore, for given interaction-membership function, the corresponding membership-equations are added to the system.

4.3.3 Solving the Systems: Treating the Parameters

We speak of those variables appearing in a matrix as X-variables $(x_1, \ldots, x_\gamma)$, whereas we think of a and b as A-variables. So far, all matrices are defined over

$$\mathbb{Q}\,[x_1, x_2, \ldots, x_\gamma, a, b]$$

of rank $\gamma + 2$. Concerning the A-variables, they do not appear in the solutions of the modelled matrices. Note that all X-variables take values in $\{0, 1\}$. Since we are only interested in the solutions w.r.t. X-components, we want to project the variety in the subspace spanned by X [40, Section 3.4].

Gathering all the polynomials mentioned, we obtain an algebraic description. This algebraic description is an ideal in $\mathbb{Q}\,[x_1, x_2, \ldots, x_\gamma, a, b]$, which is called the *coverage ideal* of the candidate matrix. From the theory of Gröbner bases we know that the

Gröbner basis is a full description of an ideal, but has a better form than a random set of generators (as the ones we obtained by our analysis of the problem) [40, Section 3.4].

Given a constructive design problem, we consider the corresponding candidate matrix and compute the multivariate polynomial system consisting of equations obtained according to the explanations in Section 4.3.2. By Lemma 4.3.1, we can use this system to reason about coverage statements concerning the matrix. We would like to explicitly point out that there are no binary conditions computed for a and b [40, Section 3.3].

For solving the resulting system of multivariate polynomial equations, we rely on the theory of Gröbner bases (cf. Chapter 3). There exist efficient algorithms for computing Gröbner bases, such as the F4 [36] and F5 algorithms [35] and we refer to [34] for a survey on signature-based algorithms.

Given a set of equations forming a coverage ideal I in $\mathbb{Q}[x_1, x_2, \ldots, x_\gamma, a, b]$, to solve the system, we first choose random values for a and b. We evaluate the polynomials in this set with the chosen values for a and b and interpret them as elements of $R = \mathbb{Q}[x_1, x_2, \ldots, x_\gamma]$. These equations define an ideal I_R restricted to R, whereas we compute a Gröbner basis (GB) of this ideal in the MAGMA computer algebra system [8]. When $\mathrm{GB}(I_R) \neq \{1\}$ then the resulting variety will entail all points (solutions of the algebraic system) that correspond to actual covering arrays upon replacing the values of X-variables into the entries of the candidate matrices. Otherwise, when $\mathrm{GB}(I_R) = \{1\}$ there is no solution to the *specific* algebraic system. We would like to note that whenever a nontrivial variety is obtained, the corresponding matrices are covering arrays by Corollary 4.2.3 [40, Section 3.5].

4.3.4 Constructing Combinatorial Designs with Algebraic Methods

We formulate candidate matrices for the various combinatorial structures discussed previously. Again, we start with partial coverage systems, continue with *covering arrays*, followed by properties corresponding to an interaction-membership function and lastly consider the case of combined models.

Partial Coverage Systems

4.3.2 Corollary. *Let a partial coverage system and compatible matrix M containing X-variables be given. Then, the following statements are equivalent.*

1. *The existence of a matrix, obtained from M by replacing all X-variables in M with a binary value, fulfilling all the conditions given by the partial coverage system.*

2. *The non-emptiness of a variety corresponding to a system of multivariate polynomial equations derived from M, consisting of the resulting binary conditions and coverage-equations determined by the partial coverage system.*

◁

Proof. The equivalence follows from Corollary 4.2.1. □

Covering Arrays

4.3.3 Corollary. *For $2 \leq k \in \mathbb{N}$, let $C = (2, k, 2)$ be a configuration, M a compatible candidate matrix containing X-variables. Then, the following statements are equivalent.*

1. *The existence of a matrix, obtained from M by replacing all X-variables in M with a binary value, such that it constitutes a* covering array *for the configuration C.*

2. *The non-emptiness of a variety corresponding to a system of multivariate polynomial equations derived from M, consisting of the resulting binary conditions and all coverage-equations.*

◁

Proof. Follows from 4.2.3. □

Membership Constraints

4.3.4 Corollary ([40, Corollary 2.6])**.** *For $2 \leq k \in \mathbb{N}$, let $C = (2, k, 2)$ be a configuration, M a compatible candidate matrix containing X-variables and Γ an interaction-membership function. Then, the following statements are equivalent.*

1. *The existence of a matrix, obtained from M by replacing all X-variables in M with a binary value, fulfilling all the conditions given by Γ.*

2. *The non-emptiness of a variety corresponding to a system of multivariate polynomial equations derived from M consisting of the resulting binary conditions and equations from the set $\mathcal{E}$ of membership-equations.*

◁

Proof. Follows from Corollary 4.2.4. □

Combined Models

We also look at the case, where we require full coverage and have to fulfill the requirements of an interaction-membership function. It is clear that we can consider all these requirements jointly, however, we note that coverage-conditions and membership requirements may contradict each other.

4.3.5 Corollary. *For $2 \leq k \in \mathbb{N}$, let $C = (2, k, 2)$ be a configuration, M a compatible candidate matrix containing X-variables and Γ an interaction-membership function. Then, the following statements are equivalent.*

1. *The existence of a matrix, obtained from M by replacing all X-variables in M with a binary value, fulfilling all the conditions given by Γ.*
2. *The non-emptiness of a variety corresponding to a system of multivariate polynomial equations derived from M consisting of the resulting binary conditions, all coverage-equations and equations from the set $\mathcal{E}$ of membership-equations.*

◁

Proof. Follows from Corollary 4.3.3 and Corollary 4.3.4. □

CHAPTER 5

Algebraic Algorithms for Problems of Covering Arrays

In this chapter, we present different problems that arise in the generation and computation of covering arrays. Since, by Theorem 2.1.8, there exists for any given MCA configuration at least one mixed level covering array compatible with that configuration – the respective Cartesian product – the most important challenge lies in the construction of optimal or near optimal mixed level covering arrays in terms of their size[1]. In particular, considerable effort has been put into developing theoretical upper and lower bounds for the covering array numbers $\mathsf{CAN}(t, k, (v_1, v_2, \ldots, v_k))$ [109, 26, 131, 116, 119].

This chapter follows [40].

In Section 5.1, we reformulate some of the problems found in [26, 55, 85, 84] to a proper computational or decisional version (in terms of computational complexity) and introduce some more that will be needed in the course of this Thesis. In Section 5.2, we formulate algorithmic approaches and exemplify how they can be used to obtain *covering arrays* for some of the considered problems.

[1] The size of a *covering array* in the sense of Definition 2.1.1 is its number of rows.

5.1 Problems for Covering Arrays

Most of the problems listed below can also be formulated for general partial coverage systems or for a combined model with an interaction-membership function, however, we follow the general focus of this work on *covering arrays* and only state them for *covering arrays*.

5.1.1 Problem ([40, Problem 1]). *(Decisional Existence) For given* MCA *configuration C and given $N \in \mathbb{N}$, decide whether a* MCA *for the configuration C with N rows exists.* ◁

5.1.2 Problem ([40, Problem 2]). *(Computational Existence (one solution), version 1) For given* MCA *configuration C and given $N \in \mathbb{N}$, construct one* MCA *for the configuration C with exactly N rows or terminate with an error.* ◁

5.1.3 Problem ([40, Problem 3]). *(Computational Existence (one solution), version 2) For given* MCA *configuration C and given $N \in \mathbb{N}$ with* $\mathsf{CAN}(C) \leq N$*, construct one* MCA *for the configuration C with exactly N rows.* ◁

5.1.4 Problem ([40, Problem 4]). *(Computational Existence (all solutions), version 1) For given* MCA *configuration C and given $N \in \mathbb{N}$, construct all* MCAs *for the configuration C with exactly N rows or terminate with an error.* ◁

5.1.5 Problem ([40, Problem 5]). *(Computational Existence (all solutions), version 2) For given* MCA *configuration C and given $N \in \mathbb{N}$ with* $\mathsf{CAN}(C) \leq N$*, construct all* MCAs *for the configuration C with exactly N rows.* ◁

5.1.6 Problem ([40, Problem 6]). *(Decisional Parameter Extension) Given a* MCA *M of strength t and given an alphabet size v, decide whether it is possible to extend the given matrix M with one additional column corresponding to a new parameter taking v values such that the extended matrix constitutes a* MCA *of strength t without adding additional rows.* ◁

5.1.7 Problem ([40, Problem 7]). *(Computational Parameter Extension (one solution), version 1) Given a* MCA *of strength t, given an alphabet size v, construct one new additional column such that the extended matrix constitutes a* MCA *of strength t with the additional parameter taking v values without adding new rows or terminate with an error.* ◁

5.1.8 Problem ([40, Problem 8]). *(Computational Parameter Extension (one solution), version 2) Given a* MCA *of strength t, given an alphabet size v and assume an affirmative*

parameter extension decision, construct one new additional column such that the extended matrix constitutes a MCA *of strength t with the additional parameter taking v values without adding new rows.* ◁

5.1.9 Problem ([40, Problem 9])**.** *(Computational Parameter Extension (all solutions), version 1) Given a* MCA *of strength t, given an alphabet size v, construct all possible new additional columns such that the extended matrices constitute* MCAs *of strength t with the additional parameter taking v values without adding new rows or terminate with an error.* ◁

5.1.10 Problem ([40, Problem 10])**.** *(Computational Parameter Extension (all solutions), version 2) Given a* MCA *of strength t, given an alphabet size v and assume an affirmative parameter extension decision, construct all possible new additional columns such that the extended matrices constitute* MCAs *of strength t with the additional parameter taking v values without adding new rows.* ◁

5.1.11 Problem ([40, Problem 11])**.** *(Decisional Vertical Extension) Given a* MCA *configuration C, a compatible matrix M and an integer r, decide whether it is possible to extend the given matrix M with exactly r rows, such that after the extension the new matrix constitutes a* MCA *for the given configuration C.* ◁

5.1.12 Problem ([40, Problem 12])**.** *(Computational Vertical Extension (one solution), version 1) Given a* MCA *configuration C, a compatible matrix M and an integer r, construct one vertical extension for M of exactly r rows such that the extended matrix constitutes a* MCA *for the given configuration C or terminate with an error.* ◁

5.1.13 Problem ([40, Problem 13])**.** *(Computational Vertical Extension (one solution), version 2) Given a* MCA *configuration C, a compatible matrix M, an integer r and assume an affirmative vertical extension decision for r, construct one vertical extension of exactly r rows such that the extended matrix constitutes a* MCA *for the given configuration C.* ◁

5.1.14 Problem ([40, Problem 14])**.** *(Computational Vertical Extension (all solutions), version 1) Given a* MCA *configuration C, a compatible matrix M and an integer r, construct all possible vertical extensions for M of exactly r rows such that the extended matrices constitute* MCAs *for the given configuration C or terminate with an error.* ◁

5.1.15 Problem ([40, Problem 15])**.** *(Computational Vertical Extension (all solutions), version 2) Given a* MCA *configuration C, a compatible matrix M, an integer r and*

assume an affirmative vertical extension decision for r, construct all possible vertical extensions of exactly r rows such that the extended matrices constitute MCAs *for the given configuration C.* ◁

5.1.16 Problem ([40, Problem 16])**.** *(Decisional Minimal Vertical Extension) Given a* MCA *configuration C, compatible matrix M and integer r, decide whether r is the least positive integer such that an r vertical extension of M for C is possible.* ◁

5.1.17 Problem ([40, Problem 17])**.** *(Computational Minimal Vertical Extension) Given a* MCA *configuration C and compatible matrix M, construct the least positive integer r such that there is an affirmative minimal vertical extension decision for r.* ◁

5.1.18 Problem ([40, Problem 18])**.** *(Decisional Coverage Verification) Given a* MCA *configuration C and a compatible matrix M, decide whether it constitutes a* MCA *for the given configuration C.* ◁

5.2 Algorithmic Approaches using Algebraic Methods

We would like to point out that most constructions operate on the transpose of a covering array, i.e., meaning that rows are corresponding to parameters.

5.2.1 An Algorithmic Approach to the Vertical Extension Problem

In the first example we present how to extend a given matrix, which is compatible with, but not a covering array for, a covering array configuration C, with one additional column such that all missing tuples will appear in the extended matrix (relates to Problem 5.1.11, Problem 5.1.12, Problem 5.1.13, Problem 5.1.16).

Specifically, we consider the configuration $C = (2, 2, 2)$, i.e., two binary parameters for strength two, and the following matrix

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.1}$$

We extend with one additional column, therefore we will be working in $P = \mathbb{Q}[x_1, x_2, a, b]$, i.e., in the multivariate polynomial ring in four variables over the rational field. The extension column consists of the first two indeterminates, $\begin{pmatrix} x_1, & x_2 \end{pmatrix}^\top$,

and is added at the end of the given matrix M:

$$Mext = \begin{pmatrix} 0 & 1 & 0 & x_1 \\ 0 & 0 & 1 & x_2 \end{pmatrix}. \tag{5.2}$$

We select the first and second row of the matrix $Mext$ and create the coverage equations. We start with deriving the coverage equation for the $\begin{pmatrix} 0, & 0 \end{pmatrix}^\top$ tuple.

$$v_{00} = \begin{pmatrix} a, & b \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & x_1 \\ 0 & 0 & 1 & x_2 \end{pmatrix} = \tag{5.3a}$$

$$\begin{pmatrix} 0, & a, & b, & x_1 a + x_2 b \end{pmatrix} \tag{5.3b}$$

Taking the product of the elements of the vector v_{00} leads to the first coverage equation $coveq^{(1,2)}_{(0,0)^\top}(Mext) = 0$. As the tuple $\begin{pmatrix} 0, & 0 \end{pmatrix}^\top$ appears in the matrix M, we expect the respective coverage equation to hold which it does as can derived from equation (5.3b). Similarly, the coverage equations for the tuples $\begin{pmatrix} 1, & 0 \end{pmatrix}^\top$ and $\begin{pmatrix} 0, & 1 \end{pmatrix}^\top$ hold as well. The only nontrivial coverage equation arises for the $\begin{pmatrix} 1, & 1 \end{pmatrix}^\top$ tuple:

$$-x_1a^3b - x_1a^2b^2 - x_2a^2b^2 - x_2ab^3 + a^3b + 2a^2b^2 + ab^3. \tag{5.4}$$

Next, we add the binary conditions for the variables x_1 and x_2:

$$x_1^2 - x_1,\, x_2^2 - x_2. \tag{5.5}$$

We now substitute random values for the A-variables

$$a = -13400/112, \tag{5.6a}$$

$$b = 290349/125, \tag{5.6b}$$

and denote by `alleqnoab` the set consisting of the polynomials occurring in the only nontrivial coverage equation (cf. equation (5.4)) and in the binary conditions (cf. equation (5.5)). All further computations take place in $R = \mathbb{Q}[x_1, x_2]$. We compute the Gröbner basis of the following ideal in MAGMA:

```
I_R = ideal < R | alleqnoab >,
```

where `I_R` denotes the restricted ideal I_R as defined in Section 4.3.3. The following computation returns the Gröbner basis polynomials (where *rank* refers to the number of

variables in a multivariate polynomial ring) in MAGMA:

Ideal of Polynomial ring of rank 2 over Rational Field, Lexicographical Order, Variables x_1, x_2, Dimension 0, Groebner basis:

$$(x_1 - 1, x_2 - 1) \tag{5.7}$$

The variety consists of only one point, corresponding to the tuple $\begin{pmatrix}1, & 1\end{pmatrix}^\top$. Substituting this solution into the extended matrix $Mext$ leads to the transpose of a covering array in the sense of Definition 2.1.1 for the configuration $(2, 2, 2)$:

$$\begin{pmatrix}0 & 1 & 0 & 1\\ 0 & 0 & 1 & 1\end{pmatrix}. \tag{5.8}$$

The pseudo-code for the Vertical Extension procedure, that has been used in this example, is given in Algorithm 2.

5.2.2 An Algorithmic Approach to the Parameter Extension Problem

In this example, we are given a covering array with k parameters and we want to extend it to a covering array with one additional parameter by extending the given matrix with a new row and in particular without adding more columns (relates to Problem 5.1.6, Problem 5.1.7, Problem 5.1.8). Consider the following matrix M, which is a covering array for the configuration $C = (2, 2, 2)$:

$$M^\top = \begin{pmatrix}0 & 1 & 0 & 1\\ 0 & 0 & 1 & 1\end{pmatrix}. \tag{5.9}$$

We will add a row vector of length four to the matrix, whose entries are the first four indeterminates of the multivariate polynomial ring $P = \mathbb{Q}[x_1, x_2, x_3, x_4, a, b]$, leading to the matrix

$$Mext = \begin{pmatrix}0 & 1 & 0 & 1\\ 0 & 0 & 1 & 1\\ x_1 & x_2 & x_3 & x_4\end{pmatrix}. \tag{5.10}$$

The next step is to derive the coverage equations. We begin by selecting the first and second row of the matrix $Mext$, the respective row-selector vector is $e^{3,1,2}(a, b) = \Big(a, b, 0\Big)$. All resulting coverage equations for this pair of selected rows hold, since by choice we

Algorithm 2 Vertical Extension

procedure VERT-EXT(M)
Require: matrix M ▷ rows corresponding to parameters
$k \leftarrow \texttt{NumberOfRows}(M)$
$P \leftarrow \mathbb{Q}[x_1, \ldots, x_k, a, b]$
$E \leftarrow (x_1, \ldots, x_k)^\top$
$Mext \leftarrow \texttt{HorizontalConcatenation}(M, E)$
$eqall \leftarrow \emptyset$
for $i = 1, 2, \ldots, k$ **do**
  for $j = i + 1, \ldots, k$ **do**
    for $\tau \in \{\begin{pmatrix}0 & 0\end{pmatrix}^\top, \begin{pmatrix}1 & 0\end{pmatrix}^\top, \begin{pmatrix}0 & 1\end{pmatrix}^\top, \begin{pmatrix}1 & 1\end{pmatrix}^\top\}$ **do**
      $eqall \leftarrow eqall \cup \{coveq_\tau^{(i,j)}(Mext)\}$
    end for
  end for
end for

SetOfBinaryConditions ← Compute binary equations
$eqall \leftarrow eqall \cup$ SetOfBinaryConditions
Randomly replace a and b in $eqall$
Regard polynomials in $eqall$ as elements of a set s over $R = \mathbb{Q}[x_1, \ldots, x_k]$
$I_R \leftarrow \texttt{ideal} < R | s >$
$GB \leftarrow \texttt{Gröbner Basis}(I_R)$
if $GB \neq \{1\}$ **then**
  $V \leftarrow \texttt{Variety}(GB)$
  Print "Non-empty set of solutions (CAs) found."
  return V
else
  Print "No solutions found."
  return $\emptyset$
end if
end procedure

started with a matrix that is already a covering array of strength two for two parameters. Therefore, we only have to consider row-selection pairs which include the newly added row. The four coverage equations for the selection of the first and third row are:

$$x_1x_2x_3x_4b^4 + x_1x_2x_3ab^3 + x_1x_3x_4ab^3 + x_1x_3a^2b^2 \tag{5.11a}$$

$$\begin{aligned} &x_1x_2x_3x_4b^4 + x_1x_2x_3ab^3 - x_1x_2x_3b^4 - x_1x_2x_4b^4 - x_1x_2ab^3 + x_1x_2b^4 + \\ &x_1x_3x_4ab^3 - x_1x_3x_4b^4 + x_1x_3a^2b^2 - 2x_1x_3ab^3 + x_1x_3b^4 - x_1x_4ab^3 + \\ &x_1x_4b^4 - x_1a^2b^2 + 2x_1ab^3 - x_1b^4 - x_2x_3x_4b^4 - x_2x_3ab^3 + x_2x_3b^4 + \\ &x_2x_4b^4 + x_2ab^3 - x_2b^4 - x_3x_4ab^3 + x_3x_4b^4 - x_3a^2b^2 + \\ &2x_3ab^3 - x_3b^4 + x_4ab^3 - x_4b^4 + a^2b^2 - 2ab^3 + b^4 \end{aligned} \tag{5.11b}$$

$$x_1x_2x_3x_4b^4 - x_1x_2x_4ab^3 - x_2x_3x_4ab^3 + x_2x_4a^2b^2 \tag{5.11c}$$

$$\begin{aligned} &x_1x_2x_3x_4b^4 - x_1x_2x_3b^4 - x_1x_2x_4ab^3 - x_1x_2x_4b^4 + x_1x_2ab^3 + x_1x_2b^4 - \\ &x_1x_3x_4b^4 + x_1x_3b^4 + x_1x_4ab^3 + x_1x_4b^4 - x_1ab^3 - x_1b^4 - \\ &x_2x_3x_4ab^3 - x_2x_3x_4b^4 + x_2x_3ab^3 + x_2x_3b^4 + x_2x_4a^2b^2 + \\ &2x_2x_4ab^3 + x_2x_4b^4 - x_2a^2b^2 - 2x_2ab^3 - x_2b^4 + x_3x_4ab^3 + x_3x_4b^4 - \\ &x_3ab^3 - x_3b^4 - x_4a^2b^2 - 2x_4ab^3 - x_4b^4 + a^2b^2 + 2ab^3 + b^4. \end{aligned} \tag{5.11d}$$

We follow again the random replacement approach for the indeterminates a and b and substitute the random values into the equations and interpret them as members of $R = \mathbb{Q}[x_1, x_2, x_3, x_4]$. In MAGMA, we compute a Gröbner basis of the respective ideal and the following computation is returned (where *rank* refers to the number of indeterminates in a multivariate polynomial ring):

Ideal of Polynomial ring of rank 4 over Rational Field,Lexicographical Order, Variables: x_1, x_2, x_3, x_4*, Dimension 0, Groebner basis:*

$$(x_1 - x_4, x_2 + x_4 - 1, x_3 + x_4 - 1, x_4^2 - x_4). \tag{5.12}$$

The variety consists of the following two points,

```
(<0, 1, 1, 0> , <1, 0, 0, 1>),
```

meaning that there are two possible ways two extend to a covering array with three binary parameters of strength two while using the given matrix M as a 'seed'.

The pseudo-code for the Parameter Extension procedure, that has been used in this example, is given in Algorithm 3.

Algorithm 3 Parameter Extension

procedure PARA-EXT(M)
Require: covering array M ▷ rows corresponding to parameters
$N \leftarrow$ `NumberOfColumns`(M)
$k \leftarrow$ `NumberOfRows`(M)
$P \leftarrow \mathbb{Q}[x_1, \ldots, x_N, a, b]$
$E \leftarrow (x_1, \ldots, x_N)$
$Mext \leftarrow$ `VerticalConcatenation`(M, E)
$j \leftarrow k + 1$
$eqall \leftarrow \emptyset$
for $i = 1, 2, \ldots, k$ **do**
  for $\tau \in \{\begin{pmatrix}0, & 0\end{pmatrix}^\top, \begin{pmatrix}1, & 0\end{pmatrix}^\top, \begin{pmatrix}0, & 1\end{pmatrix}^\top, \begin{pmatrix}1, & 1\end{pmatrix}^\top\}$ **do**
    $eqall \leftarrow eqall \cup \{coveq_\tau^{(i,j)}(Mext)\}$
  end for
end for

SetOfBinaryConditions ← Compute binary equations
$eqall \leftarrow eqall \cup$ SetOfBinaryConditions
Randomly replace a and b in $eqall$
Regard polynomials in $eqall$ as elements of a set s over $R = \mathbb{Q}[x_1, \ldots, x_N]$
$I_R \leftarrow$ `ideal` $< R|s >$
$GB \leftarrow$ `GröbnerBasis`(I_R)
if $GB \neq \{1\}$ **then**
  $V \leftarrow$ `Variety`(GB)
  Print "Parameter extension successful."
  return V
else
  Print "Parameter extension not possible."
  return $\emptyset$
end if
end procedure

5.2.3 An Algorithmic Approach to the Computational Existence of Covering Arrays

Given a configuration $C = (t, k, v)$ of a covering array with a chosen value of k (i.e., number of parameters), one may "guess" the number of rows N required for a covering

array in the sense of Definition 2.1.1 for C (relates to Problem 5.1.1, Problem 5.1.2, Problem 5.1.3, Problem 5.1.18). Clearly, in the case $\mathsf{CAN}(C) \leq N$ there is at least one solution, whereas in the case $\mathsf{CAN}(C) > N$ there are no solutions. In the first case, our algebraic modelling provides the means to actually construct such a matrix. The idea is detailed in an approach called *Guess*. There exists a 4×2 covering array for the configuration $(2, 2, 2)$, and assume that we "guess" that there exists a 4×3 matrix which forms a covering array for the configuration $(2, 3, 2)$. In contrast to Section 5.2.2, the Guess approach constructs the complete matrix. While in this example an exhaustive search based approach is still feasible, this might no longer be the case for a greater number of parameters or rows. Continuing the example, we will work in

$$P = \mathbb{Q}\left[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, a, b\right],$$

while the initialization of the candidate matrix is described in the following MAGMA code:

```
S:=RationalField();
P:=PolynomialRing(S,k*N+2);
R:=PolynomialRing(S,k*N);
M := ZeroMatrix(P,k,N);
for i in [1..k] do
    for j in [1..N] do
        M[i][j] := P.((i-1)*N+j); // P_i variable is P.i in MAGMA
    end for;
end for;
```

MAGMA code for Guess candidate matrix generation.

In the next step, we create all coverage equations and all binary conditions. It follows, that in the Guess approach, there arise

$$\binom{k}{2} \cdot 2^2 + kN \tag{5.13}$$

equations in total. In our example, we have 12 coverage equations and 12 binary conditions, yielding a total of 24 equations. So far, these equations are defined over the polynomial ring P in 14 variables.

Again, we choose random values for a and b, evaluate the polynomials, and interpret the resulting polynomials as elements of

$$R = \mathbb{Q}\left[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}\right].$$

We compute a Gröbner Basis of the respective ideal comprised of the new 24 equations in MAGMA and the computation returns that the basis consists in 17 polynomials.

The corresponding variety has 48 points, which means we have computed 48 different 3×4 matrices, those transposes constitute covering arrays in the sense of Definition 2.1.1 of strength two for three binary parameters. With an independent exhaustive search and simple tuple counting approach, we have verified that the transposes of these 48 matrices are in fact *all* 4×3 matrices, which constitute covering arrays in the sense of Definition 2.1.1 of strength two for three binary parameters.

The pseudo-code for the Guess procedure, that has been used in this example, is given in Algorithm 4.

5.3 Comparison with Greedy Algorithms

We give some cases where the presented algebraic methodology compares favorably to the IPOG algorithm, one of the most known greedy algorithms. These cases are merely used for illustration of the algebraic method's potential rather than a benchmark.

The NIST tables of covering arrays [65] are a publicly accessible source of covering arrays for various covering array configurations that have been constructed using the IPOG-F algorithm.

At [62], a covering array with 6 rows for 9 binary parameters is available. It is possible to take this covering array and apply the Parameter Extension procedure. A successful extension of the initially chosen matrix is possible in two ways to a covering array for 10 binary parameters of strength two, while keeping 6 rows. The best covering array for the configuration $(2, 10, 2)$ provided at the NIST tables is a matrix with 8 rows at [59]. The two new matrices (in the sense of Definition 2.1.1) are given below:

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Algorithm 4 Guess

procedure GUESS(k, N)
Require: $k \in \mathbb{N}$ ▷ k is the number of parameters
Require: $N \in \mathbb{N}$ ▷ N is the number of columns
$A \leftarrow k \times N$ matrix over $\mathbb{Q}[x_1, \ldots, x_{kN}, a, b]$ with entries x_i
$eqall \leftarrow \emptyset$
for $i = 1, 2, \ldots, k$ **do**
for $j = i + 1, \ldots, k$ **do**
for $\tau \in \{\begin{pmatrix}0 & 0\end{pmatrix}^\top, \begin{pmatrix}1 & 0\end{pmatrix}^\top, \begin{pmatrix}0 & 1\end{pmatrix}^\top, \begin{pmatrix}1 & 1\end{pmatrix}^\top\}$ **do**
$eqall \leftarrow eqall \cup \{coveq_\tau^{(i,j)}(A)\}$
end for
end for
end for

SetOfBinaryConditions ← Compute binary equations
$eqall \leftarrow eqall \cup$ SetOfBinaryConditions
Randomly replace a and b in $eqall$
Regard polynomials in $eqall$ as elements of a set s over $R = \mathbb{Q}[x_1, \ldots, x_{kN}]$
$I_R \leftarrow$ `ideal` $< R|s >$
$GB \leftarrow$ `GröbnerBasis`(I_R)
if $GB \neq \{1\}$ **then**
$V \leftarrow$ `Variety`(GB)
Print "Non-empty set of solutions (CAs) found."
return V
else
Print "No solutions found."
return $\emptyset$
end if
end procedure

$$\begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0
\end{pmatrix}$$

Continuing in this direction, we started with a covering array for 16 binary parameters of strength two and 8 rows available at [60]. This covering array is given in a specialized

format, where the described matrix contains entries which are undefined so as to indicate that the algorithm determined during its construction that these entries in the matrix are irrelevant from the standpoint of ensuring the pairwise coverage properties. In a preprocessing step, we replaced these entries with zeros and denote the resulting matrix as $\hat{M}$. Again, when applying the Parameter Extension procedure to the matrix $\hat{M}$, yielded twelve possible extensions while keeping 8 rows. The best covering array for the configuration $(2, 17, 2)$ provided at the NIST tables is a matrix with 10 rows available at [61].

A table listing the best known sizes of binary covering arrays of strength two is available at [24].

CHAPTER 6

Experimental Design Theory Applications

In this chapter, we move to the application of combinatorial design structures in applied computer science, specifically, to a branch of software testing called *combinatorial testing*. This rather novel application of these structures fits perfectly into the well-established understanding that not only do combinatorial design structures have lots of applications in real-world scenarios, but also that problems in real-world application domains have given rise to the definition of new combinatorial structures. For a treatment of this rich and fascinating interplay and exchange we refer to [66].

In Section 6.1, we introduce *combinatorial testing* and accentuate how *covering arrays* and their properties are leveraged to design efficient test sets for software. In Section 6.2, we briefly review some concepts from enumerative combinatorics and use them to present combinatorial structures, which can be interpreted for event sequence testing as part of combinatorial event sequence testing.

6.1 Combinatorial Testing

Combinatorial testing is a specialized branch of software testing using abstract models and providing combinatorial test case generation strategies. Input models for software are based *covering array* configurations, where the range of values which in the mathematical

formulation are assumed to be in the set[1] $[v]$ for some $v \in \mathbb{N}$ are replaced[2] by actual values specific to the SUT. Dedicated modelling methodologies have been reported in the literature [112, 7, 113, 52]. For a general treatment of and a list of software which have successfully been tested with *combinatorial testing* we refer to [79, 78].

Modern software operates in a complex environment making exhaustive testing attempts infeasible due to the combinatorial explosion of possible inputs[3].

Results from empirical studies investigating what degree of interaction occurs in real failures in real systems [74, 75, 76, 77, 123, 6, 53] have led to the formulation of the following hypothesis [78]:

> Interaction Rule: Most failures are induced by single factor faults or by the joint combinatorial effect (interaction) of two factors, with progressively fewer failures induced by interactions between three or more factors.

The implications for software testing are now immediate [78]:

> Failures appear to be caused by interactions of only a few variables, so tests that cover all such few-variable interactions can be very effective.

Covering arrays provide exactly this mentioned "interactions of only a few variables" and state-of-the-art *covering array* generation algorithms and tools have the capabilities to produce *covering arrays* that are significantly smaller than the full Cartesian product space and provide the means to generate such combinatorial test sets.

Given an input model of an SUT, then some parameter value combinations might not be valid or executable. As a result, constraints are imposed on what kind of parameter value combinations are valid. Such constraints can either be given in the form of logical expressions or as a list of forbidden tuples. The notion of constraints given in the form of a list of forbidden tuples can be algebraically semantically-equivalently captured with the notion of an interaction-membership function. The notion of constraints is indispensable for practical applications of *combinatorial testing* [130, 51, 129, 39, 17, 103].

[1] Often times MCAs are used instead of *covering arrays*.

[2] This replacement results in a relabeling of array entries.

[3] The word input is to be understood in a generic sense referring to either configuration options of software, inputs in the literal sense or any other modelled property for the software testing problem at hand.

For certain applications in practice, even more conditions are imposed upon the structure of the underlying *covering arrays* [11, 12, 110, 22] and a combinatorial fault-localization tool has been developed [45].

6.2 Enumerative Combinatorics for Combinatorial Sequence Testing

In this section, we switch to design structures where the defining notion is based on nonempty finite sequences over a fixed nonempty finite alphabet those elements are called events. The constructed combinatorial objects also exhibit characterizing coverage criteria, but of a different kind: coverage is understood in terms of appearances of all permutations of t-selections of events, but not necessarily adjacent to each other. A meta-model is created by following a weight-based selection strategy for the selection of events. The underlying formalism for the weight-based approach is given by partitions of positive integers. This way, the modelling methodology is very general and can be instantiated not only specific to each application, but can be tuned or updates within one application via an iterative process. The meta-model for event selection is built upon enumerative combinatorics. Once events have been selected, we impose combinatorial sequence coverage and the resulting artifacts will be collected in a set of sequences[4]

We briefly review essential definitions for enumerative combinatorics and partitions of positive integers in Section 6.2.1. In Section 6.2.3, we consider coverage properties of sequences, including the notion of a *sequence covering array*. A meta-model called *weighted t-way sequences* is presented in Section 6.2.4.

6.2.1 Enumerative Combinatorics

One of the main problems of enumerative combinatorics is that of enumeration, i.e., determining the number of combinatorial configurations described by a finite number of rules for all possible sizes. The astonishing connection between set theoretic operations on objects and operations on formal power series has put the notion of generating functions at the core of enumerative combinatorics.

[4]One of the capabilities of the meta-model is that the constructed sequences might not have uniform length, which makes a uniform array/matrix representation impossible.

6.2.1 Definition ([37, Definition I.1]). *A combinatorial class (abbrv., simply class) is a finite or denumerable set on which a size function is defined, satisfying the following conditions:*

1. *the size of an element is a non-negative integer;*
2. *the number of elements of any given size is finite.*

For a class $\mathcal{A}$, the size of an element $\alpha \in \mathcal{A}$ is denoted by $|\alpha|_{\mathcal{A}}$ (or simply $|\alpha|$ if the underlying class is clear from the context) and the set of objects in $\mathcal{A}$ of size n by $\mathcal{A}_n$. ◁

6.2.2 Definition ([37, Definition I.2.]). *The counting sequence of a combinatorial class is the sequence of integers $(A_n)_{n\geq 0}$, where $A_n = |\mathcal{A}_n|$ is the number of objects in class $\mathcal{A}$ that have size n.* ◁

6.2.3 Definition ([37, Definition I.3.]). *Two combinatorial classes $\mathcal{A}$ and $\mathcal{B}$ are said to be combinatorially isomorphic, denoted by $\mathcal{A} \cong \mathcal{B}$, iff their counting sequences are identical.* ◁

6.2.4 Definition ([37, Definition I.4.]). *The ordinary generating function (OGF) of a sequence (A_n) is the formal power series*

$$A(z) = \sum_{n=0}^{\infty} A_n z^n. \tag{6.1}$$

The ordinary generating function of a combinatorial class $\mathcal{A}$ is the generating function of the numbers $A_n = |\mathcal{A}_n|$. Equivalently, the OGF of class $\mathcal{A}$ admits the combinatorial form

$$A(z) = \sum_{\alpha \in \mathcal{A}} z^{|\alpha|}. \tag{6.2}$$

◁

6.2.5 Notation ([37, I.1. Symbolic enumeration methods]). *For given OGF $f(z) = \sum f_n z^n$, we denote the operation of extracting the coefficient of z^n in the formal power series $f(z)$ with $[z^n]f(z)$, i.e.,*

$$[z^n]\left(\sum_{n\geq 0} f_n z^n\right) = f_n. \tag{6.3}$$

◁

6.2.6 Definition ([37, Definition I.5.]). *Let Φ be an m-ary construction that associates to any collection of classes $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(m)}$ a new class*

$$\mathcal{A} = \Phi\left(\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(m)}\right). \tag{6.4}$$

The construction Φ is admissible iff the counting sequence (A_n) of $\mathcal{A}$ only depends on the counting sequences $(\mathcal{B}_n^{(1)}), \ldots, (\mathcal{B}_n^{(m)})$ of $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(m)}$. ◁

6.2.7 Remark ([37, I.1. Symbolic enumeration methods]). *For an admissible construction, there exists a well-defined operator Ψ acting on the corresponding ordinary generating function:*

$$A(z) = \Psi\left(B^{(1)}(z), \ldots, B^{(m)}(z)\right). \tag{6.5}$$

◁

6.2.8 Proposition ([37, I.2. Admissible constructions and specifications]). *The classes of combinatorial sum (disjoint union), Cartesian product, sequence construction (denoted by $\mathsf{SEQ}()$) and multiset construction (denoted by $\mathsf{MSET}()$) are basic constructions in the specification language for combinatorial structures.* ◁

6.2.9 Theorem ([37, Theorem I.1.]). *(Basic admissibility, unlabelled universe) Assume that $\mathcal{B}_0 = \emptyset$, then the constructions of Cartesian product, sequence and multiset are admissible. For the associated operators, the following holds:*

$$\mathcal{A} = \mathcal{B} \times \mathcal{C} \implies A(z) = B(z) \cdot C(z); \tag{6.6a}$$

$$\mathcal{A} = \mathsf{SEQ}(\mathcal{B}) \implies A(z) = \frac{1}{1 - B(z)}; \tag{6.6b}$$

$$\mathcal{A} = \mathsf{MSET}(\mathcal{B}) \implies A(z) = \prod_{n \geq 1} (1 - z^n)^{-B_n} = \exp\left(\sum_{k=1}^{\infty} \frac{1}{k} B(z^k)\right). \tag{6.6c}$$

◁

Proof. We only prove the case of the multiset operator in the only to this Thesis relevant case of a finite set $\mathcal{B}$ (with $\mathcal{B}_0 = \emptyset$ by assumption). The multiset class $\mathcal{A} = \mathsf{MSET}(\mathcal{B})$ is definable by

$$\mathsf{MSET}(\mathcal{B}) \cong \prod_{\beta \in \mathcal{B}} \mathsf{SEQ}(\{\beta\}). \tag{6.7}$$

Equation (6.7) holds, since any multiset can sorted and subsequently interpreted as a sequence of repeated elements of $\mathcal{B}$. This relation translates into generating functions

via the product and sequence rules and an application of the exp − log transformation, resulting in:

$$A(z) = \prod_{\beta \in \mathcal{B}} \left(1 - z^{|\beta|}\right)^{-1} = \prod_{n=1}^{\infty} (1 - z^n)^{-B_n} = \exp\left(\sum_{k=1}^{\infty} \frac{1}{k} B(z^k)\right). \tag{6.8}$$

□

6.2.10 Proposition ([37, I.3. Integer compositions and partitions]). *(Integers, as a combinatorial class) Let $\mathcal{I} = \mathbb{N}^{\times}$ denote the combinatorial class of all integers at least one (the summands), and let the size of each integer be its value. Then, the OGF of $\mathcal{I}$ is*

$$I(z) = \sum_{n \geq 1} z^n = \frac{z}{1 - z}, \tag{6.9}$$

since $I_n = 1$, for $n \geq 1$, corresponding to the fact that there is exactly one object in $\mathcal{I}$ for each size $n \geq 1$. ◁

6.2.11 Proposition ([37, I.3. Integer compositions and partitions]). *For partitions specified as multisets, the general translation mechanism of Theorem 6.2.9 leads to*

$$P = \mathsf{MSET}(\mathcal{I}) \implies P(z) = \exp\left(I(z) + \frac{1}{2} I(z^2) + \frac{1}{3} I(z^3) + \cdots\right), \tag{6.10}$$

and we obtain the expression

$$P(z) = \prod_{m=1}^{\infty} \frac{1}{1 - z^m}. \tag{6.11}$$

The corresponding counting sequence is EIS `A000041` *[101].* ◁

The case where the partition itself has to fulfill some properties (e.g., number of parts, bounds on summands, etc.) can also be dealt with via OGF.

6.2.2 Partitions of positive Integers

We state the definition of a partition of a positive integer and also reuse some notation from [3]:

6.2.12 Definition ([3]). *A* partition *of a positive integer n is a finite nonincreasing sequence of positive integers $\lambda_1, \lambda_2, \ldots, \lambda_r$ such that $\sum_{i=1}^{r} \lambda_i = n$. The λ_i are called the* parts *of the partition. The partition function $p(n)$ is the number of partitions of n. The function $p_{listall}(n)$ returns all possible partitions of n. As the order in which the parts*

of a partition appear is not relevant, we denote a partition as $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r) = \left(\lambda_{i_1}^{f_1}, \lambda_{i_2}^{f_2}, \ldots, \lambda_{i_\rho}^{f_\rho}\right)$. *The exponential notation uses pairwise different* $\lambda_{i_j}, 1 \leq j \leq \rho$, *making explicit the number of times a particular integer occurs as a part. The number of parts occurring in a partition including multiplicities is denoted as* $|\lambda| = r = \sum_{i=1}^{\rho} f_i$. ◁

6.2.13 Example. *The sequence* $(4, 2, 2, 1)$ *is a partition of the positive integer* 9. ◁

6.2.14 Remark ([3])**.** *It follows from Definition 6.2.12 that the order in which the parts of a partition appear is not relevant. As a result, instead of using the notation for partitions given in Definition 6.2.12 as* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$, *one can employ the following exponential notation* $\lambda = \left(\lambda_{i_1}^{f_1}, \lambda_{i_2}^{f_2}, \ldots, \lambda_{i_\rho}^{f_\rho}\right)$ *for pairwise different* λ_i. *Using the exponential notation, we have that* $n = \sum_{i=1}^{\rho} f_i \lambda_i$.

◁

6.2.3 Combinatorial Sequence Testing and Sequence Covering Arrays

The definition of *sequence covering arrays* was motivated by an applied testing scenario where a list of peripherals had to be connected to a computing device, and it was suspected that the order in which they are connected may give rise to errors when these peripherals then (try to) cooperate with each other. Since all peripherals had to be connected for full operational functionality (note that it only makes sense to connect them once), it follows naturally that in this case permutations were employed for the abstract modelling. In particular, this implies that in each test sequence, each event appears exactly once and that the length of each test sequence is constant, and equal to the total number of events [42, Section 2].

6.2.15 Definition ([80])**.** *A sequence covering array,* $SCA(N, S, t)$, *is a* $N \times s$ *matrix, where the entries are from a finite set* S *of* s *symbols, such that every* t*-way permutation of symbols from* S *occurs in at least one row and each row is a permutation of the* s *symbols. The* t *symbols in the permutation are not required to be adjacent. That is, for every* t*-way arrangement of symbols* $x_1, x_2, \cdots, x_t$, *the regular expression* $.*x_1.*x_2 \cdots .*x_t.*$ *matches at least one row in the array.* ◁

6.2.4 Weighted t-way Sequences

The concept of combinatorial *weighted t-way sequences* can be seen as an extension of the notion of *sequence covering arrays*. This more general definition is motivated both from

the theory and application sides. Combinatorial methods based on permutations have been applied to develop methodologies for event sequence testing in a branch of software testing called *combinatorial testing* [78], leading to the notion of *SCAs*. In general event models for the real world, neither the appearance of all events nor the total lengths of the considered sequences can be fixed a priori in general, making a permutation model not universally applicable. For example, if a sequence codifies occurring weather phenomena per day, the multiple appearance of an event (e.g., rain) interleaved with another event (e.g., sunshine) is a correct sequence. Also, in modelling team sport movement strategies, only a subset of all possible moves will appear in a single attack (sequence). Furthermore, in the modern pervasive computing landscape, the appearance, disappearance, and re-appearance of devices – not only in a classical client-server model, but also in the growing IoT landscape – is of utmost importance with regard to functionality and is also at the core of the computing services provided today regarding security and security testing [42, Section 1].

To address these issues, the notion of *weighted t-way sequences* was developed. The integration of a weight-based modelling (based on partitions of positive integers) increases the expressiveness of the generated sequences considerably, while on the application side gives the power to incorporate current knowledge into the weights and therefore to tune or optimize generated test sequences for the application at hand. Moreover, the weight-based approach immediately gives reasons why certain sequences will be considered and provides a genuine justification of resulting sequence lengths as well as possible multiple occurrences of the same event[42, Section 1].

6.2.16 Definition ([42, Definition 4.1])**.** *We denote the positive integers with* $\mathbb{N}$*. Let* $\mathcal{E}$ *be a nonempty finite set those elements are called* events*, and let* ω *be a positive integer-valued* weight function *defined on the set of events; i.e.* $\omega\colon \mathcal{E} \longrightarrow \mathbb{N}$*. Let* $\mathcal{S}$ *be the set of all nonempty finite sequences over the alphabet* $\mathcal{E}$*. Based on the weight function* ω *defined on the set of events* $\mathcal{E}$*, we define a weight function for elements in* $\mathcal{S}$*, which will, par abus de notation, also be denoted with* ω*. For* $\mu \in \mathbb{N}$ *and given nonempty finite sequence* $s = (s_1, s_2, \ldots, s_\mu) = (s_i)_{i=1,\ldots,\mu} \in \mathcal{S}$ *of length* μ*, we define the weight of* s *as*

$$\omega(s) = \sum_{i=1}^{\mu} w(s_i). \tag{6.12}$$

◁

For given $R \in \mathbb{N}$, let $\emptyset \neq \mathcal{B} \subseteq \{1, 2, \ldots, R\}$, and for $\beta \in \mathcal{B}$ consider the set $p_{listall}(\beta)$. Let $\lambda = \left(\lambda_{i_1}^{f_1}, \lambda_{i_2}^{f_2}, \ldots, \lambda_{i_\rho}^{f_\rho}\right) \in p_{listall}(\beta)$ and $\zeta \in \{1, \ldots, \rho\}$. The part λ_{i_ζ} now encodes a specific weight, and we consider all events which get assigned this weight; i.e., we consider the set[5]

$$\omega^{-1}[\{\lambda_{i_\zeta}\}] \subseteq \mathcal{E}. \tag{6.13}$$

Starting from the partition λ, we now build a Cartesian product of sets of events that correspond to the appearing parts in λ via their weight as shown in (6.13):

$$\mathcal{C}_\lambda = \underbrace{\omega^{-1}[\{\lambda_{i_1}\}] \times \cdots \times \omega^{-1}[\{\lambda_{i_1}\}]}_{f_1 \text{ times}} \times \cdots \times \underbrace{\omega^{-1}[\{\lambda_{i_\rho}\}] \times \cdots \times \omega^{-1}[\{\lambda_{i_\rho}\}]}_{f_\rho \text{ times}}. \tag{6.14}$$

For $t \in \mathbb{N}$ and $C \in \mathcal{C}_\lambda$, artificially consider all of the elements appearing in the coordinates of C as pairwise different and forming a set $T(C)$; then it is possible to generate[6] a *SCA* $S(T(C), t)$ of strength t for the alphabet $T(C)$ in the sense of Definition 6.2.15. Let $t \in \mathbb{N}$ be the desired t-way coverage, then the complete *weighted t-way sequences* test set is defined as[7]:

$$\mathcal{T} = \bigcup_{\beta \in \mathcal{B}} \bigcup_{\lambda \in p_{listall}(\beta)} \bigcup_{C \in \mathcal{C}_\lambda} S(T(C), t) \subseteq \mathcal{S}. \tag{6.15}$$

6.2.17 Remark ([42, Remark 4.2])**.** *Instead of considering* all *possible partitions of a positive integer, one might only consider a proper nonempty subset. For example, one could restrict the parts that are allowed to appear in the partitions to make sure that the set defined in relation* (6.13) *is nonempty; or allow only parts which are greater or smaller than a given threshold, or restrict the number of appearing summands. Any such condition can be interpreted as a condition on the events that are allowed to appear in the sequence (via their weight) or on the length of the sequence.* ◁

6.2.18 Proposition ([42, Proposition 4.3])**.** *We list below some observations about the generated test set* $\mathcal{T}$ *in relation* (6.15)*, which immediately follow from the construction process described above:*

[5] Let A, B be sets and $f\colon A \longrightarrow B$ a function. For $C \subseteq B$, we denote the *preimage* of the subset C of B under the function f by $f^{-1}[C] = \{a \in A\colon f(a) \in C\}$.

[6] In this Thesis, we regard the problem of practical generation of a *SCA* for given parameters $t \in \mathbb{N}$ and alphabet size $\mu \in \mathbb{N}$ as solved and are not concerned with its efficiency.

[7] In relation (6.15) the appearing *SCAs* are interpreted as sets of sequences derived from the rows of the *SCAs*, where the respective matrices are regarded as sets of rows.

- *The set $\mathcal{B}$ determines those cumulative weights of sequences, which will be considered in the test sequence generation process, and thus makes it possible to limit the testing to exactly those cumulative weights of interest.*
- *Individual test sequence lengths might be nonuniform across all partitions. This is due to the fact that various partitions may have a differing number of parts.*
- *Events may appear multiple times in test sequences, or not at all. Multiple appearances happen, when a partition contains a part multiple times. However, from the point of generated t-SCAs, these events are regarded as different.*
- *The determination of the weight function ω and the set $\mathcal{B}$ is application domain specific and can be adapted in an iterative testing setting.*
- *The construction given above also works if the value t for the t-way coverage is nonuniform and depends on β, λ and $\mathcal{C}_\lambda$.*

◁

6.2.19 Remark ([42, Remark 4.4])**.** *It is possible to extend this approach, mutatis mutandis, to a setting where the weight of an event $e \in \mathcal{E}$ depends on the considered partition and cumulative weight, and where the weight of a finite nonempty sequence $s \in \mathcal{S}$ is a positive-valued function of the elements of the sequence and not necessarily their sum. These extensions might make it easier to tune the weights in an application domain, however, the connection to partitions of positive integers is then lost.* ◁

The concept of *weighted t-way sequences* has been used to derive test cases for security testing of the TLS protocol [43].

CHAPTER 7

Conclusion

In this Thesis, we used algebraic techniques to reason about certain design structures, in particular *covering arrays*. This was possible due to a connection between the statement of appearance "at least once" and zeros of multivariate polynomials. An algebraic enforcement of coverage conditions means the expression of the existence of a certain tuple as a zero of a multivariate polynomial.

Based on this building block, algebraic descriptions of designs and design problems concerning their characterization were successfully treated.

A routine application of symbolic computation techniques, including the theory of Gröbner bases, resulted in the formulation of algorithmic approaches for the algebraic treatment of certain constructive design problems. Points in corresponding nonempty varieties were used to instantiate matrices, which when interpreted as designs structures exhibited all required coverage properties.

We discussed applications of experimental design theory to the field of software testing. The defining notion of coverage regarding tuples in *covering arrays* and permutations of events in *sequence covering arrays* can be leveraged with great success in these real-world domains. Furthermore, a weight-based modelling approach called *weighted t-way sequences* built upon integer partitions and *sequence covering arrays* was presented, which offers additional meta-modelling capabilities.

Finally, the presented results in this Thesis confirm and extend the versatility of combinatorial structures inside and outside of discrete mathematics.

Glossary and Notation

$M^\top$ denotes, for a matrix M, its transpose.

$\mathbb{N}$ denotes the natural numbers, including zero.

$\mathbb{N}^\times$ denotes the natural numbers excluding zero. This notation applies in this sense to any set.

$\bar{K}$ denotes, for a field K, the algebraic closure of K.

$|S|$ denotes, for a set S, its cardinality.

$[n]$ is for $n \in \mathbb{N}$ by definition equal to the set $\{0, 1, \ldots, n-1\}$.

$\mathcal{P}(S)$ denotes, for a set S, its power set.

$\mathcal{I}_k$ is for $k \in \mathbb{N}$ by definition equal to the set $\{(i,j) \mid 1 \leq i < j \leq k\} \subseteq [k] \times [k]$.

$radical(I)$ denotes, for an ideal I, its radical.

Bibliography

[1] Bestoun S. Ahmed and Kamal Z. Zamli. "A variable strength interaction test suites generation strategy using Particle Swarm Optimization". In: *Journal of Systems and Software* 84.12 (2011), pp. 2171–2185. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2011.06.004. URL: http://www.sciencedirect.com/science/article/pii/S0164121211001464.

[2] Bestoun S. Ahmed, Kamal Z. Zamli, and Chee Peng Lim. "Application of Particle Swarm Optimization to uniform and variable strength covering array construction". In: *Applied Soft Computing* 12.4 (2012), pp. 1330–1347. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2011.11.029. URL: http://www.sciencedirect.com/science/article/pii/S1568494611004716.

[3] George E Andrews. *The theory of partitions*. 2. Cambridge university press, 1998.

[4] Satoshi Aoki and Akimichi Takemura. "Design and analysis of fractional factorial experiments from the viewpoint of computational algebraic statistics". In: *Journal of Statistical Theory and Practice* 6.1 (2012), pp. 147–161.

[5] T. Becker and V. Weispfenning. *Gröbner bases. A Computational Approach to Commutative Algebra*. Vol. 141. Graduate Studies in Mathematics. New York: Springer-Verlag, 1993.

[6] K. Z. Bell and M. A. Vouk. "On effectiveness of pairwise methodology for testing network-centric software". In: *2005 International Conference on Information and Communication Technology*. Dec. 2005, pp. 221–235. DOI: 10.1109/ITICT.2005.1609626.

[7] M. N. Borazjany et al. "An Input Space Modeling Methodology for Combinatorial Testing". In: *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*. Mar. 2013, pp. 372–381. DOI: 10.1109/ICSTW.2013.48.

[8] Wieb Bosma, John Cannon, and Catherine Playoust. “The Magma algebra system. I. The user language”. In: *J. Symbolic Comput.* 24.3-4 (1997). Computational algebra and number theory (London, 1993), pp. 235–265.

[9] Renée C. Bryce and Charles J. Colbourn. “A density-based greedy algorithm for higher strength covering arrays”. In: *Software Testing, Verification and Reliability* 19.1 (2009), pp. 37–53. DOI: 10.1002/stvr.393. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/stvr.393. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.393.

[10] Renée C Bryce and Charles J Colbourn. “One-test-at-a-time heuristic search for interaction test suites”. In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM. 2007, pp. 1082–1089.

[11] Renée C. Bryce and Charles J. Colbourn. “Prioritized interaction testing for pair-wise coverage with seeding and constraints”. In: *Information and Software Technology* 48.10 (2006). Advances in Model-based Testing, pp. 960–970. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2006.03.004. URL: http://www.sciencedirect.com/science/article/pii/S0950584906000401.

[12] Renée C. Bryce and Charles J. Colbourn. “Test Prioritization for Pairwise Interaction Coverage”. In: *Proceedings of the 1st International Workshop on Advances in Model-based Testing*. A-MOST ’05. St. Louis, Missouri: ACM, 2005, pp. 1–7. ISBN: 1-59593-115-5. DOI: 10.1145/1082983.1083275. URL: http://doi.acm.org/10.1145/1082983.1083275.

[13] Renée C. Bryce and Charles J. Colbourn. “The density algorithm for pairwise interaction testing”. In: *Software Testing, Verification and Reliability* 17.3 (2007), pp. 159–182. DOI: 10.1002/stvr.365. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/stvr.365. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.365.

[14] Renée C. Bryce, Charles J. Colbourn, and Myra B. Cohen. “A Framework of Greedy Methods for Constructing Interaction Test Suites”. In: *Proceedings of the 27th International Conference on Software Engineering*. ICSE ’05. St. Louis, MO, USA: ACM, 2005, pp. 146–155. ISBN: 1-58113-963-2. DOI: 10.1145/1062455.1062495. URL: http://doi.acm.org/10.1145/1062455.1062495.

[15] B. Buchberger. “Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems”. German. In: *Aequationes mathematicae* 4.3 (1970), pp. 374–383.

[16] Bruno Buchberger. “Bruno Buchberger’s PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal”. In: *J. Symb. Comput.* 41 (3-4 Mar. 2006), pp. 475–511. ISSN: 0747-7171. DOI: http://dx.doi.org/10.1016/j.jsc.2005.09.007. URL: http://dx.doi.org/10.1016/j.jsc.2005.09.007.

[17] Andrea Calvagna and Angelo Gargantini. “A Logic-Based Approach to Combinatorial Testing with Constraints”. In: *Tests and Proofs*. Ed. by Bernhard Beckert and Reiner Hähnle. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 66–83. ISBN: 978-3-540-79124-9.

[18] M. A. Chateauneuf, Charles J. Colbourn, and D. L. Kreher. “Covering Arrays of Strength Three”. In: *Designs, Codes and Cryptography* 16.3 (May 1999), pp. 235–242. ISSN: 1573-7586. DOI: 10.1023/A:1008379710317. URL: https://doi.org/10.1023/A:1008379710317.

[19] X. Chen et al. “Variable Strength Interaction Testing with an Ant Colony System Approach”. In: *2009 16th Asia-Pacific Software Engineering Conference*. Dec. 2009, pp. 160–167. DOI: 10.1109/APSEC.2009.18.

[20] D. M. Cohen et al. “The AETG system: an approach to testing based on combinatorial design”. In: *IEEE Transactions on Software Engineering* 23.7 (July 1997), pp. 437–444. ISSN: 0098-5589. DOI: 10.1109/32.605761.

[21] M. B. Cohen, C. J. Colbourn, and A. C. H. Ling. “Augmenting simulated annealing to build interaction test suites”. In: *14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003.* Nov. 2003, pp. 394–405. DOI: 10.1109/ISSRE.2003.1251061.

[22] Myra B Cohen, Aduri Pavan, and NV Vinodchandran. “Budgeted testing through an algorithmic lens”. In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM. 2016, pp. 948–951.

[23] P.M. Cohn. *Algebra*. Algebra Bd. 1. Wiley, 1974. ISBN: 9780471164319. URL: https://books.google.at/books?id=RH0dAQAAMAAJ.

[24] Charles Colbourn. *Table for CAN(2,k,2) for k up to 20000.* http://www.public.asu.edu/~ccolbou/src/tabby/2-2-ca.html. [Online; accessed December 31, 2015].

[25] Charles J. Colbourn. "Augmentation of Covering Arrays of Strength Two". In: *Graphs and Combinatorics* 31.6 (Nov. 2015), pp. 2137–2147. ISSN: 1435-5914. DOI: 10.1007/s00373-014-1519-9. URL: https://doi.org/10.1007/s00373-014-1519-9.

[26] Charles J. Colbourn. "Combinatorial Aspects of Covering Arrays". In: *Le Matematiche* LIX.I-II (2004), pp. 125–172.

[27] Charles J. Colbourn. "Covering arrays, augmentation, and quilting arrays". In: *Discrete Mathematics, Algorithms and Applications* 06.03 (2014), p. 1450034. DOI: 10.1142/S1793830914500347. eprint: https://doi.org/10.1142/S1793830914500347. URL: https://doi.org/10.1142/S1793830914500347.

[28] Charles J. Colbourn and Jeffrey H. Dinitz. *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2006. ISBN: 1584885068.

[29] Charles J. Colbourn and Peyman Nayeri. "Randomized Post-optimization for t-Restrictions". In: *Information Theory, Combinatorics, and Search Theory: In Memory of Rudolf Ahlswede*. Ed. by Harout Aydinian, Ferdinando Cicalese, and Christian Deppe. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 597–608. ISBN: 978-3-642-36899-8. DOI: 10.1007/978-3-642-36899-8_30. URL: https://doi.org/10.1007/978-3-642-36899-8_30.

[30] David A Cox, John B Little, and Donal O'Shea. *Ideals, varieties, and algorithms : an introduction to computational algebraic geometry and commutative algebra.* eng. 2. ed. Undergraduate texts in mathematics. New York, NY [u.a.]: Springer, 1997. ISBN: 0387946802.

[31] Jacek Czerwonka. "Pairwise testing in the real world: Practical extensions to test-case scenarios". In: (2008).

[32] Peter Danziger et al. "Covering arrays avoiding forbidden edges". In: *Theoretical Computer Science* 410.52 (2009). Combinatorial Optimization and Applications, pp. 5403–5414. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2009.07.057. URL: http://www.sciencedirect.com/science/article/pii/S030439750900454X.

[33] Persi Diaconis, Bernd Sturmfels, et al. “Algebraic algorithms for sampling from conditional distributions”. In: *The Annals of statistics* 26.1 (1998), pp. 363–397.

[34] Christian Eder and Jean-Charles Faugère. “A survey on signature-based Gröbner basis computations”. Anglais. Apr. 2014. URL: http://hal.inria.fr/hal-00974810.

[35] Jean Charles Faugère. “A New Efficient Algorithm for Computing GrÖBner Bases Without Reduction to Zero (F5)”. In: *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*. ISSAC ’02. Lille, France: ACM, 2002, pp. 75–83. ISBN: 1-58113-484-3. DOI: 10.1145/780506.780516. URL: http://doi.acm.org/10.1145/780506.780516.

[36] Jean-Charles Faugere. “A new efficient algorithm for computing Gröbner bases (F 4)”. In: *Journal of pure and applied algebra* 139.1 (1999), pp. 61–88.

[37] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511801655.

[38] Michael Forbes et al. “Refining the in-parameter-order strategy for constructing covering arrays”. In: *Journal of Research of the National Institute of Standards and Technology* 113.5 (2008), p. 287.

[39] Angelo Gargantini et al. “Validation of Constraints Among Configuration Parameters Using Search-Based Combinatorial Interaction Testing”. In: *Search Based Software Engineering*. Ed. by Federica Sarro and Kalyanmoy Deb. Cham: Springer International Publishing, 2016, pp. 49–63. ISBN: 978-3-319-47106-8.

[43] B. Garn et al. “Weighted Combinatorial Sequence Testing for the TLS Protocol”. In: *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. to appear. 2019.

[44] Brady J. Garvin, Myra B. Cohen, and Matthew B. Dwyer. “Evaluating improvements to a meta-heuristic search for constrained interaction testing”. In: *Empirical Software Engineering* 16.1 (Feb. 2011), pp. 61–102. ISSN: 1573-7616. DOI: 10.1007/s10664-010-9135-7. URL: https://doi.org/10.1007/s10664-010-9135-7.

[45] L. S. Ghandehari et al. “BEN: A combinatorial testing-based fault localization tool”. In: *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. Apr. 2015, pp. 1–4. DOI: 10.1109/ICSTW.2015.7107446.

[46] S. A. Ghazi and M. A. Ahmed. “Pair-wise test coverage using genetic algorithms”. In: *The 2003 Congress on Evolutionary Computation, 2003. CEC ’03.* Vol. 2. Dec. 2003, 1420–1424 Vol.2. DOI: 10.1109/CEC.2003.1299837.

[47] Loreto Gonzalez-Hernandez, Nelson Rangel-Valdez, and Jose Torres-Jimenez. “Construction of Mixed Covering Arrays of Variable Strength Using a Tabu Search Approach”. In: *Combinatorial Optimization and Applications*. Ed. by Weili Wu and Ovidiu Daescu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 51–64. ISBN: 978-3-642-17458-2.

[48] Loreto Gonzalez-Hernandez and Jose Torres-Jimenez. “MiTS: A New Approach of Tabu Search for Constructing Mixed Covering Arrays”. In: *Advances in Soft Computing*. Ed. by Grigori Sidorov, Arturo Hernández Aguirre, and Carlos Alberto Reyes García. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 382–393. ISBN: 978-3-642-16773-7.

[49] Loreto Gonzalez-Hernandez, José Torres-Jiménez, and Nelson Rangel-Valdez. “An Exact Approach to Maximize the Number of Wild Cards in a Covering Array”. In: *Advances in Artificial Intelligence*. Ed. by Ildar Batyrshin and Grigori Sidorov. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 210–221. ISBN: 978-3-642-25324-9.

[50] Loreto Gonzalez-Hernandez et al. “A Post-optimization Strategy for Combinatorial Testing: Test Suite Reduction through the Identification of Wild Cards and Merge of Rows”. In: *Advances in Computational Intelligence*. Ed. by Ildar Batyrshin and Miguel González Mendoza. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 127–138. ISBN: 978-3-642-37798-3.

[51] M. Grindal, J. Offutt, and J. Mellin. “Managing Conflicts When Using Combination Strategies to Test Software”. In: *2007 Australian Software Engineering Conference (ASWEC’07)*. Apr. 2007, pp. 255–264. DOI: 10.1109/ASWEC.2007.27.

[52] Mats Grindal and Jeff Offutt. “Input Parameter Modeling for Combination Strategies”. In: *Proceedings of the 25th Conference on IASTED International Multi-Conference: Software Engineering*. SE’07. Innsbruck, Austria: ACTA Press, 2007, pp. 255–260. URL: http://dl.acm.org/citation.cfm?id=1332044.1332085.

[53] Mats Grindal, Jeff Offutt, and Sten F. Andler. “Combination testing strategies: a survey”. In: *Software Testing, Verification and Reliability* 15.3 (2005), pp. 167–199. DOI: 10.1002/stvr.319. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/stvr.319. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.319.

[54] Alan Hartman and Leonid Raskin. “Problems and algorithms for covering arrays”. In: *Discrete Mathematics* 284.1 (2004). Special Issue in Honour of Curt Lindner on His 65th Birthday, pp. 149–156. ISSN: 0012-365X. DOI: https://doi.org/10.1016/j.disc.2003.11.029. URL: http://www.sciencedirect.com/science/article/pii/S0012365X0400130X.

[55] Alan Hartman and Leonid Raskin. “Problems and algorithms for covering arrays”. In: *Discrete Mathematics* 284.1 (2004), pp. 149–156.

[56] A.S. Hedayat, N.J.A. Sloane, and J. Stufken. *Orthogonal Arrays: Theory and Applications*. Springer Series in Statistics. Springer New York, 2012. ISBN: 9781461214786. URL: https://books.google.at/books?id=lQfpBwAAQBAJ.

[57] Brahim Hnich et al. “Constraint Models for the Covering Test Problem”. In: *Constraints* 11.2 (July 2006), pp. 199–219. ISSN: 1572-9354. DOI: 10.1007/s10601-006-7094-9. URL: https://doi.org/10.1007/s10601-006-7094-9.

[58] Gérard Huet. “Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems: Abstract Properties and Applications to Term Rewriting Systems”. In: *J. ACM* 27.4 (Oct. 1980), pp. 797–821. ISSN: 0004-5411. DOI: 10.1145/322217.322230. URL: http://doi.acm.org/10.1145/322217.322230.

[59] IPOG-F. *CA(2,10,2)*. http://math.nist.gov/coveringarrays/ipof/cas/t=2/v=2/ca.2.2^10.txt.zip. [Online; accessed December 31, 2015].

[60] IPOG-F. *CA(2,16,2)*. http://math.nist.gov/coveringarrays/ipof/cas/t=2/v=2/ca.2.2^16.txt.zip. [Online; accessed December 31, 2015].

[61] IPOG-F. *CA(2,17,2)*. http://math.nist.gov/coveringarrays/ipof/cas/t=2/v=2/ca.2.2^17.txt.zip. [Online; accessed 31-December-2015].

[62] IPOG-F. *CA(2,9,2)*. http://math.nist.gov/coveringarrays/ipof/cas/t=2/v=2/ca.2.2^9.txt.zip. [Online; accessed December 31, 2015].

[63] IPOG-F. *CA(3,4,2)*. https://math.nist.gov/coveringarrays/ipof/cas/t=3/v=2/ca.3.2^4.txt.zip. [Online; accessed February 11, 2019].

[64] IPOG-F. *CA(3,7,2)*. https://math.nist.gov/coveringarrays/ipof/cas/t=3/v=2/ca.3.2^7.txt.zip. [Online; accessed February 11, 2019].

[65] NIST ITL. *Covering Array Tables*. http://math.nist.gov/coveringarrays/. [Online; accessed December 31, 2015].

[66] C J. Colbourn, J H. Dinitz, and D R. Stinson. "Applications of Combinatorial Designs to Communications, Cryptography, and Networking". In: *Lond Math Soc Lecture Note Ser* 267 (Feb. 1999). DOI: 10.1017/CBO9780511721335.004.

[67] Jacek Czerwonka. *Available Tools*. 2019. URL: http://www.pairwise.org/tools.asp (visited on 02/10/2019).

[68] L. Kampel, B. Garn, and D. E. Simos. "Combinatorial Methods for Modelling Composed Software Systems". In: *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. Mar. 2017, pp. 229–238. DOI: 10.1109/ICSTW.2017.43.

[69] Ludwig Kampel, Bernhard Garn, and Dimitris E. Simos. "Covering Arrays via Set Covers". In: *Electronic Notes in Discrete Mathematics* 65 (2018). 7th International Conference on Algebraic Informatics (CAI 2017): Design Theory Track, pp. 11–16. ISSN: 1571-0653. DOI: https://doi.org/10.1016/j.endm.2018.02.014. URL: http://www.sciencedirect.com/science/article/pii/S1571065318300416.

[70] S. K. Khalsa and Y. Labiche. "An Orchestrated Survey of Available Algorithms and Tools for Combinatorial Testing". In: *2014 IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)*. Vol. 00. Nov. 2014, pp. 323–334. DOI: 10.1109/ISSRE.2014.15. URL: doi.ieeecomputersociety.org/10.1109/ISSRE.2014.15.

[71] Youngil Kim, Dae-Heung Jang, and Christine M. Anderson-Cook. "Selecting the Best Wild Card Entries in a Covering Array". In: *Quality and Reliability Engineering International* 33.7 (2017), pp. 1615–1627. DOI: 10.1002/qre.2129. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qre.2129. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.2129.

[72] Takashi Kitamura et al. “Optimal Test Suite Generation for Modified Condition Decision Coverage Using SAT Solving”. In: *Computer Safety, Reliability, and Security*. Ed. by Barbara Gallina, Amund Skavhaug, and Friedemann Bitsch. Cham: Springer International Publishing, 2018, pp. 123–138. ISBN: 978-3-319-99130-6.

[73] Kristoffer Kleine, Ilias Kotsireas, and Dimitris E. Simos. “Evaluation of Tie-Breaking and Parameter Ordering for the IPO Family of Algorithms Used in Covering Array Generation”. In: *Combinatorial Algorithms*. Ed. by Costas Iliopoulos, Hon Wai Leong, and Wing-Kin Sung. Cham: Springer International Publishing, 2018, pp. 189–200. ISBN: 978-3-319-94667-2.

[74] D. R. Kuhn, R. N. Kacker, and Y. Lei. “Estimating t-Way Fault Profile Evolution During Testing”. In: *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 2. June 2016, pp. 596–597. DOI: 10.1109/COMPSAC.2016.110.

[75] D. R. Kuhn and V. Okum. “Pseudo-Exhaustive Testing for Software”. In: *2006 30th Annual IEEE/NASA Software Engineering Workshop*. Apr. 2006, pp. 153–158. DOI: 10.1109/SEW.2006.26.

[76] D. R. Kuhn and M. J. Reilly. “An investigation of the applicability of design of experiments to software testing”. In: *27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2002. Proceedings.* Dec. 2002, pp. 91–95. DOI: 10.1109/SEW.2002.1199454.

[77] D. R. Kuhn, D. R. Wallace, and A. M. Gallo. “Software fault interactions and implications for software testing”. In: *IEEE Transactions on Software Engineering* 30.6 (June 2004), pp. 418–421. ISSN: 0098-5589. DOI: 10.1109/TSE.2004.24.

[78] D Richard Kuhn, Raghu N Kacker, and Yu Lei. *Introduction to combinatorial testing*. CRC press, 2013.

[79] D. Richard Kuhn, Raghu N. Kacker, and Yu Lei. *SP 800-142. Practical Combinatorial Testing*. Tech. rep. Gaithersburg, MD, United States, 2010.

[80] D Richard Kuhn et al. “Combinatorial methods for event sequence testing”. In: *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*. IEEE. 2012, pp. 601–609.

[81] D.R. Kuhn and M.J. Reilly. “An investigation of the applicability of design of experiments to software testing”. In: *Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE*. Dec. 2002, pp. 91–95. DOI: 10.1109/SEW.2002.1199454.

[82] Victor Kuliamin and Alexander Petukhov. “Covering Arrays Generation Methods Survey”. In: *Leveraging Applications of Formal Methods, Verification, and Validation*. Ed. by Tiziana Margaria and Bernhard Steffen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 382–396. ISBN: 978-3-642-16561-0.

[83] Jim Lawrence et al. “A survey of binary covering arrays”. In: *the electronic journal of combinatorics* 18.1 (2011), p. 84.

[84] Yu Lei and K-C Tai. “In-parameter-order: A test generation strategy for pairwise testing”. In: *High-Assurance Systems Engineering Symposium, 1998. Proceedings. Third IEEE International*. IEEE. 1998, pp. 254–261.

[85] Yu Lei et al. “IPOG: A general strategy for t-way software testing”. In: *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the*. IEEE. 2007, pp. 549–556.

[86] Yu Lei et al. “IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing”. In: *Software Testing, Verification and Reliability* 18.3 (2008), pp. 125–148. DOI: 10.1002/stvr.381. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/stvr.381. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.381.

[87] X. Li et al. “Refining a Randomized Post-optimization Method for Covering Arrays”. In: *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*. Mar. 2014, pp. 143–152. DOI: 10.1109/ICSTW.2014.16.

[88] Jason R. Lobb et al. “Cover starters for covering arrays of strength two”. In: *Discrete Mathematics* 312.5 (2012), pp. 943–956. ISSN: 0012-365X. DOI: https://doi.org/10.1016/j.disc.2011.10.026. URL: http://www.sciencedirect.com/science/article/pii/S0012365X11004833.

[89] Feifei Ma and Jian Zhang. “Finding Orthogonal Arrays Using Satisfiability Checkers and Symmetry Breaking Constraints”. In: *PRICAI 2008: Trends in Artificial Intelligence*. Ed. by Tu-Bao Ho and Zhi-Hua Zhou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 247–259. ISBN: 978-3-540-89197-0.

[90] J. D. McCaffrey. “An Empirical Study of Pairwise Test Set Generation Using a Genetic Algorithm”. In: *Information Technology: New Generations, Third International Conference on(ITNG)*. Vol. 00. Apr. 2010, pp. 992–997. DOI: 10.1109/ITNG.2010.93. URL: doi.ieeecomputersociety.org/10.1109/ITNG.2010.93.

[91] J. D. McCaffrey. “Generation of Pairwise Test Sets Using a Genetic Algorithm”. In: *2009 33rd Annual IEEE International Computer Software and Applications Conference*. Vol. 1. July 2009, pp. 626–631. DOI: 10.1109/COMPSAC.2009.91.

[92] J. D. McCaffrey. “Generation of pairwise test sets using a simulated bee colony algorithm”. In: *2009 IEEE International Conference on Information Reuse Integration*. Aug. 2009, pp. 115–119. DOI: 10.1109/IRI.2009.5211598.

[93] Karen Meagher and Brett Stevens. “Covering arrays on graphs”. In: *Journal of Combinatorial Theory, Series B* 95.1 (2005), pp. 134–151. ISSN: 0095-8956. DOI: https://doi.org/10.1016/j.jctb.2005.03.005. URL: http://www.sciencedirect.com/science/article/pii/S0095895605000419.

[94] Karen Meagher and Brett Stevens. “Group construction of covering arrays”. In: *Journal of Combinatorial Designs* 13.1 (2005), pp. 70–77. DOI: 10.1002/jcd.20035. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcd.20035. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcd.20035.

[95] Teo Mora and Lorenzo Robbiano. “The Gröbner fan of an ideal”. In: *Journal of Symbolic Computation* 6.2 (1988), pp. 183–208. ISSN: 0747-7171. DOI: https://doi.org/10.1016/S0747-7171(88)80042-7. URL: http://www.sciencedirect.com/science/article/pii/S0747717188800427.

[96] Lucia Moura, Sebastian Raaphorst, and Brett Stevens. “The Lovász Local Lemma and Variable Strength Covering Arrays”. In: *Electronic Notes in Discrete Mathematics* 65 (2018). 7th International Conference on Algebraic Informatics (CAI 2017): Design Theory Track, pp. 43–49. ISSN: 1571-0653. DOI: https://doi.org/10.1016/j.endm.2018.02.019. URL: http://www.sciencedirect.com/science/article/pii/S1571065318300465.

[97] Peyman Nayeri, Charles J. Colbourn, and Goran Konjevod. “Randomized post-optimization of covering arrays”. In: *European Journal of Combinatorics* 34.1 (2013). Combinatorics and Stringology, pp. 91–103. ISSN: 0195-6698. DOI: https:

//doi.org/10.1016/j.ejc.2012.07.017. URL: http://www.sciencedirect.com/science/article/pii/S0195669812001345.

[98] Changhai Nie and Hareton Leung. "A survey of combinatorial testing". In: *ACM Computing Surveys (CSUR)* 43.2 (2011), p. 11.

[99] C. Nie et al. "Search Based Combinatorial Testing". In: *2012 19th Asia-Pacific Software Engineering Conference*. Vol. 1. Dec. 2012, pp. 778–783. DOI: 10.1109/APSEC.2012.16.

[100] Kari J. Nurmela. "Upper bounds for covering arrays by tabu search". In: *Discrete Applied Mathematics* 138.1 (2004). Optimal Discrete Structures and Algorithms, pp. 143–152. ISSN: 0166-218X. DOI: https://doi.org/10.1016/S0166-218X(03)00291-9. URL: http://www.sciencedirect.com/science/article/pii/S0166218X03002919.

[101] OEIS Foundation Inc. (2019). *The On-Line Encyclopedia of Integer Sequences*. https://oeis.org/A000041. [Online; accessed February 9, 2019].

[102] Jose Carlos Perez-Torres and Jose Torres-Jimenez. "A graph-based postoptimization approach for covering arrays". In: *Quality and Reliability Engineering International* 33.8 (2017), pp. 2171–2180. DOI: 10.1002/qre.2176. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qre.2176. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.2176.

[103] J. Petke. "Constraints: The Future of Combinatorial Interaction Testing". In: *2015 IEEE/ACM 8th International Workshop on Search-Based Software Testing*. May 2015, pp. 17–18. DOI: 10.1109/SBST.2015.11.

[104] Giovanni Pistone, Eva Riccomagno, and Henry P Wynn. *Algebraic statistics: Computational commutative algebra in statistics*. CRC Press, 2000.

[105] Giovanni Pistone and Henry P Wynn. "Generalised confounding with Gröbner bases". In: *Biometrika* 83.3 (1996), pp. 653–666.

[106] Lorenzo Robbiano. "On the theory of graded structures". In: *Journal of Symbolic Computation* 2.2 (1986), pp. 139–170. ISSN: 0747-7171. DOI: https://doi.org/10.1016/S0747-7171(86)80019-0. URL: http://www.sciencedirect.com/science/article/pii/S0747717186800190.

[107] Lorenzo Robbiano. "Term orderings on the polynomial ring". In: *EUROCAL '85*. Ed. by Bob F. Caviness. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 513–517. ISBN: 978-3-540-39685-7.

[108] Arturo Rodriguez-Cristerna and Jose Torres-Jimenez. “A Simulated Annealing with Variable Neighborhood Search Approach to Construct Mixed Covering Arrays”. In: *Electronic Notes in Discrete Mathematics* 39 (2012). EURO Mini Conference, pp. 249–256. ISSN: 1571-0653. DOI: https://doi.org/10.1016/j.endm.2012.10.033. URL: http://www.sciencedirect.com/science/article/pii/S1571065312000340.

[109] K. Sarkar and C. Colbourn. “Upper Bounds on the Size of Covering Arrays”. In: *SIAM Journal on Discrete Mathematics* 31.2 (2017), pp. 1277–1293. DOI: 10.1137/16M1067767. eprint: https://doi.org/10.1137/16M1067767. URL: https://doi.org/10.1137/16M1067767.

[110] Kaushik Sarkar et al. “Partial Covering Arrays: Algorithms and Asymptotics”. In: *Combinatorial Algorithms*. Ed. by Veli Mäkinen, Simon J. Puglisi, and Leena Salmela. Cham: Springer International Publishing, 2016, pp. 437–448. ISBN: 978-3-319-44543-4.

[111] Kaushik Sarkar et al. “Partial Covering Arrays: Algorithms and Asymptotics”. In: *Theory of Computing Systems* 62.6 (Aug. 2018), pp. 1470–1489. ISSN: 1433-0490. DOI: 10.1007/s00224-017-9782-9. URL: https://doi.org/10.1007/s00224-017-9782-9.

[112] I. Segall, R. Tzoref-Brill, and A. Zlotnick. “Common Patterns in Combinatorial Models”. In: *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. Apr. 2012, pp. 624–629. DOI: 10.1109/ICST.2012.150.

[113] I. Segall, A. Zlotnick, and R. Tzoref-Brill. “Simplified Modeling of Combinatorial Test Spaces”. In: *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation(ICST)*. Vol. 00. Apr. 2012, pp. 573–579. DOI: 10.1109/ICST.2012.143. URL: doi.ieeecomputersociety.org/10.1109/ICST.2012.143.

[114] Toshiaki Shiba, Tatsuhiro Tsuchiya, and Tohru Kikuno. “Using artificial life techniques to generate test cases for combinatorial testing”. In: IEEE. 2004, pp. 72–77.

[115] N. J. A. Sloane. “Covering arrays and intersecting codes”. In: *Journal of Combinatorial Designs* 1.1 (1993), pp. 51–63. DOI: 10.1002/jcd.3180010106. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcd.

3180010106. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcd.3180010106.

[116] Brett Stevens, Lucia Moura, and Eric Mendelsohn. "Lower Bounds for Transversal Covers". In: *Designs, Codes and Cryptography* 15.3 (Dec. 1998), pp. 279–299. ISSN: 1573-7586. DOI: 10.1023/A:1008329410829. URL: https://doi.org/10.1023/A:1008329410829.

[117] J. Torres-Jimenez and I. Izquierdo-Marquez. "Survey of Covering Arrays". In: *2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. Sept. 2013, pp. 20–27. DOI: 10.1109/SYNASC.2013.10.

[118] Jose Torres-Jimenez and Arturo Rodriguez-Cristerna. "Metaheuristic post-optimization of the NIST repository of covering arrays". In: *CAAI Transactions on Intelligence Technology* 2.1 (2017), pp. 31–38. ISSN: 2468-2322. DOI: https://doi.org/10.1016/j.trit.2016.12.006. URL: http://www.sciencedirect.com/science/article/pii/S2468232216300312.

[119] Jose Torres-Jimenez and Eduardo Rodriguez-Tello. "New bounds for binary covering arrays using simulated annealing". In: *Information Sciences* 185.1 (2012), pp. 137–152. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2011.09.020. URL: http://www.sciencedirect.com/science/article/pii/S0020025511004774.

[120] Wolfgang Trinks. "Über B. Buchbergers verfahren, systeme algebraischer gleichungen zu lösen". In: *Journal of Number Theory* 10.4 (1978), pp. 475–488. ISSN: 0022-314X. DOI: https://doi.org/10.1016/0022-314X(78)90019-7. URL: http://www.sciencedirect.com/science/article/pii/0022314X78900197.

[121] Georgios Tzanakis et al. "Constructing new covering arrays from LFSR sequences over finite fields". In: *Discrete Mathematics* 339.3 (2016), pp. 1158–1171. ISSN: 0012-365X. DOI: https://doi.org/10.1016/j.disc.2015.10.040. URL: http://www.sciencedirect.com/science/article/pii/S0012365X15003945.

[122] Robert A. Walker and Charles J. Colbourn. "Tabu search for covering arrays using permutation vectors". In: *Journal of Statistical Planning and Inference* 139.1 (2009). Special Issue on Metaheuristics, Combinatorial Optimization and Design of Experiments, pp. 69–80. ISSN: 0378-3758. DOI: https://doi.org/10.1016/j.

jspi.2008.05.020. URL: http://www.sciencedirect.com/science/article/pii/S0378375808002310.

[123] DOLORES R. WALLACE and D. RICHARD KUHN. "FAILURE MODES IN MEDICAL DEVICE SOFTWARE: AN ANALYSIS OF 15 YEARS OF RECALL DATA". In: *International Journal of Reliability, Quality and Safety Engineering* 08.04 (2001), pp. 351–371. DOI: 10.1142/S021853930100058X. eprint: https://doi.org/10.1142/S021853930100058X. URL: https://doi.org/10.1142/S021853930100058X.

[124] Volker Weispfennig. "Constructing Universal Groebner Bases". In: *Proceedings of the 5th International Conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*. AAECC-5. London, UK, UK: Springer-Verlag, 1989, pp. 408–417. ISBN: 3-540-51082-6. URL: http://dl.acm.org/citation.cfm?id=646024.676396.

[125] Alan W. Williams and Robert L. Probert. "Formulation of the Interaction Test Coverage Problem as an Integer Program". In: *Testing of Communicating Systems XIV: Application to Internet Technologies and Services*. Ed. by Ina Schieferdecker, Hartmut König, and Adam Wolisz. Boston, MA: Springer US, 2002, pp. 283–298. ISBN: 978-0-387-35497-2. DOI: 10.1007/978-0-387-35497-2_21. URL: https://doi.org/10.1007/978-0-387-35497-2_21.

[126] F. Winkler. *Polynomial Algorithms in Computer Algebra*. 1996.

[127] A. Yamada et al. "Greedy combinatorial test case generation using unsatisfiable cores". In: *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Sept. 2016, pp. 614–624.

[128] A. Yamada et al. "Optimization of Combinatorial Testing by Incremental SAT Solving". In: *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. Apr. 2015, pp. 1–10. DOI: 10.1109/ICST.2015.7102599.

[129] L. Yu et al. "An Efficient Algorithm for Constraint Handling in Combinatorial Test Generation". In: *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*. Mar. 2013, pp. 242–251. DOI: 10.1109/ICST.2013.35.

[130] L. Yu et al. “Constraint handling in combinatorial test generation using forbidden tuples”. In: *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. Apr. 2015, pp. 1–9. DOI: 10.1109/ICSTW.2015.7107441.

[131] Ruyue Yuan, Zoe Koch, and Anant Godbole. “Covering array bounds using analytical techniques”. In: *arXiv preprint arXiv:1405.2844* (2014).

[132] Jian Zhang. “Automatic Symmetry Breaking Method Combined with SAT”. In: *Proceedings of the 2001 ACM Symposium on Applied Computing*. SAC ’01. Las Vegas, Nevada, USA: ACM, 2001, pp. 17–21. ISBN: 1-58113-287-5. DOI: 10.1145/372202.372206. URL: http://doi.acm.org/10.1145/372202.372206.

[133] Jian Zhang, Zhiqiang Zhang, and Feifei Ma. *Automatic generation of combinatorial test data*. Springer, 2014.

[134] Y. Zhao et al. “Cascade: A Test Generation Tool for Combinatorial Testing”. In: *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*. Mar. 2013, pp. 267–270. DOI: 10.1109/ICSTW.2013.37.