STUDIA INFORMATICA

Formerly: *Zeszyty Naukowe Politechniki Śląskiej, seria INFORMATYKA*

Volume 21, Number 2 (40)

Adam KAPRALSKI

MODELING ARBITRARY SETS OF COMBINTORIAL OBJECTS AND THEIR SEQUENTIAL AND PARALLEL GENERATION

WYDAWNICTWO POLITECHNIKI ŚLĄSKIEJ

Silesian University of Technology Press
Gliwice 2000

STUDIA INFORMATICA

Formerly: *Zeszyty Naukowe Politechniki Śląskiej, seria INFORMATYKA*

Volume 21, Number 2 (40)

Adam KAPRALSKI

MODELING ARBITRARY SETS OF COMBINTORIAL OBJECTS AND THEIR SEQUENTIAL AND PARALLEL GENERATION

WYDAWNICTWO POLITECHNIKI ŚLĄSKIEJ

Silesian University of Technology Press
Gliwice 2000

ABSTRACT. We give the representations of different subsets of compositions, decompositions, number and set partitions by means of choice functions of indexed families. The structure of the symmetric sets of choice functions is represented by investigated tables D related to Pascal's triangle and to the Stirling's numbers. Unranking and ranking representation models concerning sets of choice functions are developed. We study transformations of the models their equivalence, congruence and isomorphism basing on the tables D. We have shown superiority of the investigated models for representing the sets of combinatorial objects in comparing with the classical methodology. Then, the basic algorithms and their variants for different classes of models concerning the generation of choice functions are developed. The algorithms concerning rank use widely the tables D. The general methodology for parallel or distributed generation of the choice functions in SIMD or MIMD systems is used and developed.

STRESZCZENIE. Zaproponowano metodę reprezentowania arbitralnych pod- zbiorów kompozycji liczb, dekompozycji zbiorów, podziałów liczb i zbiorów poprzez odpowiadające zbiory funkcji wyboru rodzin indeksowych. Istotne znaczenie dla reprezentowania arbitralnych zbiorów obiektów kombinatorycznych mają rosnące funkcje wyboru, monotoniczne funkcje wyboru oraz bijekcje. Zaproponowano i rozwinięto modele zbiorów obiektów kombinatorycznych jako modele nierankingowe oraz modele rankingowe. Przedstawiono ogólną teorię struktury arbitralnych zbiorów funkcji wyboru. Struktura ta reprezentowana jest poprzez tablice D, których elementy pozostają w związku z trójkątem Pascal'a oraz liczbami Stirling'a. Tablice D są wykorzystywane ponadto w algorytmach tworzących podstawy systemu generowania obiektów kombinatorycznych. Przedstawione w dalszej części twierdzenie o rankingu precyzuje własności modeli optymalnych z punktu widzenia możliwie najbardziej zwartych zbiorów rankingów funkcji należących do modelowanych zbiorów. Uzyskanie możliwie najbardziej zwartego zbioru rankingów ma istotne znaczenie dla sekwencyjnego, rozproszonego i równoległego generowania zbiorów funkcji wyboru. Ogólna metodologia generacji zbiorów funkcji wyboru w systemach SIMD, MIMD jest przedstawiona i rozwinięta w kolejnych rozdziałach.

Key words: partition, composition, decomposition, indexed families, monotonic choice functions, increasing choice functions, choice bijections, modeling by rank, layer, structural numbers, Ranking Theorem, combinatorial object generation

CONTENTS

SPIS TREŚCI

Introduction

The main combinatorial computations concern the generation and processing subsets, subgraphs or submatrices or other similar structures or different their collections. These all structures are commonly called the combinatorial objects. This text is devoted to the representation methodology of arbitrary sets of the combinatorial objects and to their automatic generation in sequential or distributed or parallel systems.

Many authors have contributed to the generation of full sets of permutations, combinations, partitions, compositions and decompositions in the last two decades [1], [8], [23], [2], [3], [32],[49], [57], [44], [52], [13], [51], [37], [38], [55]. On the other hand the problem of generation of full sets of the basic combinatorial objects has been discussed for over 300 years and the details concerning the old history and newest achievements can be found in [23], [44], [42], [53].

The classical approach represents sets of combinatorial objects basically by means of full sets of permutations or combinations or partitions or compositions or decompositions. However, arbitrary sets (subsets) are much more needed for practical applications than the full sets of the basic combinatorial objects. The classical approach proposes only very few solutions of this problem. The main method is to represent and to generate sets of combinatorial objects by rank [1]. This method is not suitable for numerous practical applications especially if the rank range of the represented set is not compact. In fact, if more compact rank range, then usage of the rank is more suitable. The reasons are as follows:

(i) the algorithms concerning rank are more time consuming than the counterparts not related to the rank,

(*ii*) if more compact rank range, then less runs of a ranking algorithm required,

(*iii*) the sets of combinatorial objects represented by non-compact rank ranges are very hard for uniform distribution,

(*iv*) the method lacks of efficient modelling subsets of combinatorial objects by ranking.

In the last years several algebraic and linguistic approaches for modelling and generation of sets (subsets) of combinatorial objects have been proposed [4], [17], [19], [33], [49], [54], [41]. Among them the general methodology of representing sets of combinatorial objects by sets of choice functions proposed by the author in [19] seems to be the most attractive. Several papers contribute to this approach [49],[35], [39], [23], [20].

This text accepts and develops the methodology basing on the choice functions of indexed families. We investigate and study the unranking models those represent sets of choice functions. Our main concerns are sets of increasing choice functions, sets of monotonic choice functions and sets of choice bijections. We demonstrated usage of these models for representing the basic combinatorial objects, i.e., permutations, combinations, variations, partitions, decompositions and compositions. The given concepts can be seen as a development of the consequences that could be drawn from Hall's Theorem [14]. That theorem by many authors was considered as the basic result in combinatorics but developed here theory for the first time shows it so widely. Then, the tables D for representing the structure of sets of choice functions are studied. The tables D create new theoretical foundation for deeper understanding the structure of the whole combinatorics in relation to Stirling's numbers and to Pascal's triangle. These tables are the main components of the ranking models, we investigate also transformations of the models basing on the tables D. Ranking Theorem presents the main existential result for our philosophy of getting the most suitable models. As far as the representation of sets of combinatorial objects is concerned the choice function approach gives benefits that could not be obtained using the classical methodology, we demonstrate these benefits. Then, we develop the system of the basic procedures for the generation of choice functions. We investigate the general algorithms and their most efficient variants dedicated to specific classes of the models. We emphasize that the tables D are the essential components of the developed algorithms concerning the rank. Then, the sequential, parallel or distributed systems of the generation are given. In the last years there are investigated non-standard generation problems and tasks that concern non-exhaustive generation and generation of a specified subset of combinatorial objects [45], [46], [5], [10]. It can happen for these new tasks that no backtracking-less algorithms exists for their solving. The investigated notion "the Q property" enables us to explain the model properties leading to existence of backtracking-less algorithm. We demonstrate also usefulness of the presented here approach into solving the general tasks for the models that the Q property does not

hold. We have shown backtracking and looking forward algorithms showing simultaneously the structure of efforts undertaken in order to diminish as much as possible the number of backtracking and looking forward steps. We have also shown development of digital signature system basing on the given theory. The results concerning non exhaustive generation are for the first time so strongly unified supplying convenient tools for implementation and further development of optimization algorithms especially the genetic algorithms.

The most important advantages of representing the combinatorial objects by choice functions of indexed families and by using the proposed here approach can be pointed out as follows:

(*i*) methods of modelling are versatile and flexible in order to meet different and easily changeable requirements,

(*ii*) methods of generation of choice functions of indexed families are also very flexible, and suitable for sequential, parallel or distributed systems,

(*iii*) algorithms for the generation of the basic combinatorial objects by means of choice functions are the fastest known,

(*iv*) the known methodology is very convenient, both for development of the software generators and for development of the dedicated hardware generators as well [34], [36], [39],

(*v*) the whole theory that is developed for modelling and for the generation is versatile and it enables us to find good methods of the generation for very uncommon and irregular sets of combinatorial objects too, even if symmetry of the sets of choice functions or the Q property do not hold,

(*vi*) the given approach separates clearly modelling problems from the generation problems, so selection of proper algorithms is oriented towards a model instead being oriented towards an application that is commonly used till now.

Chapter 1 is devoted to recalling the known results concerning choice function modelling and to the essential properties of the basic models, from the point of view of their correctness and their convenience for modelling.

Chapter 2 develops usage of particular models for representing full sets of the basic combinatorial objects and their subsets.

Chapter 3 presents structural properties of the models and it investigates widely the tables D and it studies their properties.

Chapter 4 develops ranking concept using the tables D and it contains Ranking Theorem.

Chapter 5 presents examples of intuitive modelling given sets of combinatorial objects and it contains formal methods of transformations of the models representing a set of combinatorial objects.

Chapter 6 presents the basic algorithms and their variants concerning the generation of choice functions and the ranking and unranking methodology.

Chapter 7 is devoted to the methodology of the generation of sets of choice functions in sequential or distributed or parallel systems.

Chapter 8 presents miscellaneous generation methods and algorithms concerning non-exhaustive or random generation mainly for the models that the Q property does not hold. We have also given modeling and generation of combinatorial objects that is widely applicable in cryptography.

Acknowledgments. The main ideas objective for this text were investigated when the author stayed at the University of Aizu in Japan and led the project "Generation of combinatorial objects" for development new teaching subject and offered the classes on SCCP project for students under the same title. The results concerning the generation of combinatorial objects for cryptography purpose was supported by Polish Committee for Research, project no 811C - 026 - 98C/4258. The author wish to thank to the referees Professors Konrad Wala from Academy of Mining and Metallurgy in Cracow and Józef Drewniak from the Silesian University in Katowice. Thanks are also to dr. Zbigniew Kokosiński who participated in the SCCP project supporting the theory especially by development of hardware generators of combinatorial objects. The author wish to thank to his former students from the University of Aizu for their active participation at the classes of SCCP, that revealed the topics requesting more detailed explanations.

1

Preliminary

1.1 Basic notions and properties

By a **basic combinatorial object,** we mean permutation, variation[1], combination, set partition, number partition, number composition or set decomposition.

We recall now some definitions concerning the basic combinatorial objects.

A **composition** of a number n into m parts is an ordered set of numbers $n_1, n_2,..., n_m$, so that $n = n_1 + n_2 +...+ n_m$.

A **partition** of a number n into m parts is a composition for which additionally $n_{i+1} \geq n_i$, $1 \leq i \leq m-1$.

A **decomposition** of a given set $S = \{1, 2, ..., n\}$ is a set of blocks (subsets of S) $\{B_1, B_2,..., B_m\}$, so that $B_1 \cup B_2 \cup ... \cup B_m = S$, and for any two B_i, B_j, $B_i \cap B_j = \emptyset$, $i, j \in \{1, 2, ..., m\}$. It is allowed that $B_i = \emptyset$, however, we may also make the restriction $B_i \neq \emptyset$, so always one has to make sure whether empty blocks are allowed or not. Distinct enumerations of blocks means different decompositions, for instance, $B_1 = \{1, 2, 4\}$, $B_2 = \{3, 5\}$ and $B_1 = \{3, 5\}$, $B_2 = \{1, 2, 4\}$ are two distinct decompositions.

A decomposition whose blocks are not empty and are enumerated in accordance with an increasing order of their minimum elements is called a **partition of a set.**

---

[1] The term permutation instead variation is widely used in English

So, the set of partitions is a subset of the corresponding set of decompositions.

Now, we will recall the basic definitions and results concerning indexed families and choice functions. The basic theory for representation of combinatorial objects by choice functions of a given indexed family has been given in [19].

With denotation $< \mathcal{G}_i >, 1 \leq i \leq m$, we mean an **indexed family** of sets; where $\mathcal{G}_i \subseteq \mathcal{U}$, $\mathcal{U} = \{1, 2, ...n\}$, we write also "$< \mathcal{G}_i >$, for every $i \in I$" or shortly $< \mathcal{G}_i >$, $i \in I$ if $I \subseteq \{1, 2, ..., m\}$.

A mapping h: $i \longrightarrow q$, where $1 \leq i \leq m$, $q \in \mathcal{G}_i$ is called a **choice function** [56].

If a choice function h of a given indexed family has to represent a given combinatorial object, then $h(i) = q$ must satisfy a defined **requirement** W. We say also that q satisfies requirement W or, if this requirement is satisfied for every $i \in I$, then we say that function h satisfies the requirement W. Generally, the requirement W is given in the form $h(i)$ R E$[i, h(i-1), h(i-2),..., h(1)]$ where by R, we mean a relation and E$[i, h(i-1), h(i-2),..., h(1)]$ denotes any expression involving i and/or $h(i-1), h(i-2), ..., h(1)$. For the most common cases the requirement W posses a simpler and more convenient form, i.e., $h(i)$ R E$[h(i-1), h(i-2),..., h(1)]$. We denote by $\{h\}_{\mathcal{G}_i}$ the **set of all the choice functions** h of a given indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$. With denotation $h \in \{h\}_{\mathcal{G}_i}$, we mean that the given choice function h satisfies the given requirement W and it is a choice function of the family $< \mathcal{G}_i >, 1 \leq i \leq m$. This complicated denotation we use since there is need to distinguish sets of choice functions $\{h\}$ of not specified indexed family, sets of choice functions $\{h\}_{\mathcal{G}_i}$ of the indexed family $< \mathcal{G}_i >, 1 \leq i \leq m$, and sets of choice functions $\{h\}_{\mathcal{G}j}$ of an indexed family $< \mathcal{G}_j >$, $j \in J$ or different their subsets.

A set of choice functions $\{h\}_{\mathcal{G}_i}$ of a given indexed family that satisfy a fixed requirement W, we will call the full set of choice functions. A set of choice functions represents the corresponding set of combinatorial objects. The given full set of choice functions represents the full set of combinatorial objects. Since a set of combinatorial objects can be a subset of another greater set of combinatorial objects, therefore, full set of combinatorial objects and full set of the corresponding choice functions are relative notions dependent on a given model.

We model subsets of choice functions in three ways:

(*) by restricting indexed sets of the original indexed family,

(**) by imposing additional requirement W_1 that is to be satisfied by the choice functions of the original indexed family, then a choice function belongs to the requested subset if it satisfies conjunction of the requirements W and W_1,

(***) by specification of the ranks of the choice functions which create a given subset.

The indexed family is **reduced or not redundant** for given W and W_1 if for every $i \in I$ and for every $q \in \mathcal{G}_i$ there is a choice function $h \in \{h\}_{\mathcal{G}_i}$ that $h(i) = q$. We say correspondingly that an indexed set $\mathcal{G}_i$ is reduced or not redundant if the above is satisfied for $\mathcal{G}_i$.

The set of choice functions $\{h\}_{\mathcal{G}_i}$ of a given indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$, satisfies the **Q property** if for any incomplete choice function f: $1 \longrightarrow q_1, 2 \longrightarrow q_2, ..., z \longrightarrow q_z$, $q_z \in \mathcal{G}_z$, there exists $h \in \{h\}_{\mathcal{G}_i}$ so that $f(j) = h(j)$, where $1 \leq j \leq z$, $z < m$. Both f and h must satisfy the requirements W and W_1, if any. Briefly speaking satisfiability of the Q property means that for each partial choice function there is its extension to a full choice function of the given indexed family. If the model containing a given indexed family and the requirements W and W_1 satisfies the Q property, then the choice functions can be generated using backtracking–less algorithm, see [19].

Definition 1.1.1 *The indexed family $< \mathcal{B}_i >$, $1 \leq i \leq m$, is called the maximal indexed family for a given requirement W if it is not redundant and if $\{h\}_{\mathcal{B}_i} = \{h\}_{\mathcal{E}_i}$, where $\mathcal{E}_i = \{1, 2, ..., n\}$, $1 \leq i \leq m$.*

Suppose, the indexed family $< \mathcal{B}_i >$, $1 \leq i \leq m$ is the maximal indexed family for a fixed requirement W.

Definition 1.1.2 *We say that a given family $< \mathcal{G}_i >$, $1 \leq i \leq m$, is a deformed indexed family for a given requirement W if $\{h\}_{\mathcal{G}_i} \subset \{h\}_{\mathcal{B}_i}$, where choice functions $\{h\}_{\mathcal{B}_i}$ satisfy the same requirement W.*

If $< \mathcal{G}_i >$, $1 \leq i \leq m$ is a deformed indexed family, then there exists at least one set $\mathcal{G}_i$ that $\mathcal{G}_i \subset \mathcal{B}_i$.

Given is an indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$, $\mathcal{G}_i \subseteq \{1, 2, ..., n\}$, let $<\mathcal{G}_i^1, \mathcal{G}_i^2, ..., \mathcal{G}_i^{k_i}>$ be a partition of the set $\mathcal{G}_i$ into k_i non–void blocks.
Any indexed family of the form $< \mathcal{G}_i^{r_i} >$, $1 \leq i \leq m$, $1 \leq r_i \leq k_i$; we call the **layer** of the family $< \mathcal{G}_i >$, $1 \leq i \leq m$.

The following result is the foundation for development of distributed models of a set of combinatorial objects and for the generation of choice functions, in a distributed or parallel system.

Any choice function $h \in \{h\}_{\mathcal{G}_i}$ is the choice function exactly one of its layer $< \mathcal{G}_i^{r_i} >$, $1 \leq i \leq m$,[19].

1.2 Main unranking models

Definition 1.2.1 *The basic unranking models S_U are the following systems <indexed family, requirement W, set of choice functions> or shortly $<< \mathcal{G}_i >$, $1 \leq i \leq m$; W; $\{h\}_{\mathcal{G}_i} >$.*

Usually, we model full sets of distinct combinatorial objects as the sets of increasing choice functions or as the sets of choice bijections or as the sets of monotonic choice functions or as certain their subsets.

Subsets of the full sets of combinatorial objects are represented by the models $S_U = << \mathcal{G}_i' >, 1 \leq i \leq m; W; W_1; \{h\}_{\mathcal{G}_i'} >$. Then, the choice functions $h \in \{h\}_{\mathcal{G}_i'}$ have to satisfy conjunction of the requirements W and W_1. Therefore, we use also the following specification $S_U = << \mathcal{G}_i' >, 1 \leq i \leq m; W \wedge W_1; \{h\}_{\mathcal{G}_i'} >$. The indexed sets $\mathcal{G}_i'$ satisfy the requirement $\mathcal{G}_i' \subseteq \mathcal{G}_i$, $1 \leq i \leq m$. If $\mathcal{G}_i' \subset \mathcal{G}_i$ at least for one index i, then $\{h\}_{\mathcal{G}_i'} \subset \{h\}_{\mathcal{G}_i}$ even if the requirement W_1 is void.

The requirements W and W_1 have the structure of a Boolean expression that uses alternative and conjunction. On the other hand each their component can represent even single choice function. Therefore, every arbitrary set of choice functions can be represented by the model S_U.

The left part of this section is devoted to fundamental properties of the basic unranking models.

1.2.1 *Increasing choice functions*

Given is an indexed family $< \mathcal{A}_i >$, $1 \leq i \leq m$, where $\mathcal{A}_i = \{i, i+1, ..., n-m+i\}$. The requirement

$$W : h(i) > h(i-1) \tag{1.1}$$

is to be satisfied by each choice function $h \in \{h\}_{\mathcal{A}_i}$.

The system $<< \mathcal{A}_i >, 1 \leq i \leq m; W$ given in (1.1); $h \in \{h\}_{\mathcal{A}_i} >$ represents all the increasing choice functions of the indexed family $< \mathcal{A}_i >$, $1 \leq i \leq m$.

It is known [19] that the above model satisfies the Q property and that the increasing choice functions of a given model correspond one-to-one m–elements combinations chosen from n items. Moreover, the indexed family $< \mathcal{A}_i >$, $1 \leq i \leq m$, is the maximal indexed family.

For modelling subsets of the given set of increasing choice functions, we will use both restricting the indexed sets or imposing additional requirements W_1.

Assumptions.

Given is the model:

$$<< \mathcal{P}_i >, 1 \leq i \leq m; W \text{ given in } (1.1); h \in \{h\}_{\mathcal{P}_i} > \tag{1.2}$$

representing subsets of $\{h\}_{\mathcal{A}_i}$, where

(*i*) $\mathcal{P}_i \subseteq \mathcal{A}_i$, $1 \leq i \leq m$,

(*ii*) $\max(\mathcal{P}_i) > \max(\mathcal{P}_{i-1})$, $2 \leq i \leq m$,

(*iii*) $\min(\mathcal{P}_i) > \min(\mathcal{P}_{i-1})$, $2 \leq i \leq m$,

Proposition 1.2.1 *The model (1.2) satisfies the Q property and the indexed family* $< \mathcal{P}_i >$, $1 \leq i \leq m$ *is reduced, if and only if, the assumptions* $(i) \div (iii)$ *given above do hold.*

Proof. For proving the sufficient condition, we observe that if (ii) and (iii) hold, then the indexed family is the reduced indexed family, since for each $q \in \mathcal{P}_i$ we can have a partial mapping $h(1)$, $h(2)$, $...h(i-1)$, $h(i) = q$ that $h(j) = \min(\mathcal{P}_j)$, for every $j < i$.

Moreover, for every partial mapping $h(1)$, $h(2)$, $...h(i-1)$, we can assign $i \longrightarrow \max(\mathcal{P}_i)$. Then, $h(1)$, $h(2)$, ..., $h(i-1)$, $h(i) = \max(\mathcal{P}_i)$ is the increasing partial mapping, for every $2 \leq i \leq m$. Hence, we have assertion that the Q property holds.

We prove now the necessary condition.

If (ii) does not hold, then exists j that $\max(\mathcal{P}_{j-1}) > \max(\mathcal{P}_j)$, $j \in \{2, 3, ..., m\}$. Suppose, (ii) holds for every $i < j$. Then, exists an increasing partial mapping $h(1)$, $h(2)$, ..., $h(j-1)$ that $h(j-1) = \max(\mathcal{P}_{j-1})$. Then, no value $h(j) \in \mathcal{P}_j$ could be assigned, in order to have an increasing partial mapping $h(1)$, $h(2)$, ..., $h(j-1)$, $h(j)$. Consequently the Q property does not hold.

If (iii) does not hold at least for one value $j \in \{2, 3, ..., m\}$, then $\min(\mathcal{P}_j) < q$, $q \in \mathcal{P}_{j-1}$. Then, the value $\min(\mathcal{P}_j)$ could not be assigned to any increasing partial mapping $h(1)$, $h(2)$, $...h(j-1)$, so that the partial mapping $h(1)$, $h(2)$, $...h(j-1)$, $h(j) = \min(\mathcal{P}_j)$ would be increasing one. Hence, the indexed family $< \mathcal{P}_i >$, $1 \leq i \leq m$ would not be reduced.

It finishes the proof.

□

We consider sets of choice functions of the indexed family $< \mathcal{A}_i >$, $1 \leq i \leq m$ that satisfy the requirement $W \wedge W_1$, where $W : h(i) > h(i-1)$ and W_1 is an additional requirement.

Consider the following cases of W_1.

$$h(z) \neq q, \text{where } q \neq \max(\mathcal{A}_z), z \in \{1, 2, ..., m\}. \tag{1.3}$$

$$h(z) = \max(\mathcal{A}_z), z \in \{1, 2, ..., m\}. \tag{1.4}$$

$$h(z) = h(z-1) + 1, z \in \{1, 2, ..., m\}. \tag{1.5}$$

$$h(z) - h(z-1) \neq q, z \in \{1, 2, ..., m\}, q \neq 1. \tag{1.6}$$

For W_1 given in (1.5) or (1.6) the model $<< \mathcal{A}_i >, 1 \leq i \leq m; W$ given in (1.1); W_1; $h \in \{h\}_{\mathcal{A}_i} >$ satisfies the Q property.

One can observe that the model $<< \mathcal{A}_i >, 1 \leq i \leq m; W$ given in (1.1); W_1 is given in (1.3) or W_1 given in (1.4); $h \in \{h\}_{\mathcal{A}_i} >$ is equivalent to the model (1.2). Moreover, if W_1 is given in (1.4), then the Q property holds since the assumptions (i), (ii), (iii) hold, however if $\mathcal{P}_i = \max(\mathcal{A}_z)$, then the indexed family $< \mathcal{P}_i >$, $1 \leq i \leq m$ is not reduced. If W_1 is given in (1.3), then the Q property holds. Moreover, if $q \neq \min(\mathcal{A}_z)$, then the corresponding indexed family $< \mathcal{P}_i >$, $1 \leq i \leq m$ is also reduced.

Testing satisfiability of the Q property for models containing any indexed family $< \mathcal{P}_i >$, $1 \leq i \leq m$ and W_1 given in (1.3) or (1.5) or (1.6) requires individual considerations.

If an arbitrary requirement W_1 is given, then for satisfiability of the Q property and for making the model reduced, the indexed sets can require to be deformed.

Given is the requirement

$$W : h(i) > h(i-1) + a_i, \quad a_i \in \{0, 1, ...\}, \text{ where } 2 \leq i \leq m. \tag{1.7}$$

The indexed family $< \mathcal{R}_i >$, $1 \leq i \leq m$, is specified as follows $\mathcal{R}_i = \{ 1 + a_2 + a_3 + ... + a_i, 1 + a_2 + a_3 + ... + a_i + 1, ..., n - m - a_n - a_{n-1} - ... - a_i + i\}$. Then, the model

$$<< \mathcal{R}_i >, 1 \leq i \leq m; W \text{ given in (1.7)}; \{h\}_{\mathcal{R}_i} > \tag{1.8}$$

is reduced and the Q property holds. One can prove it observing that $\max(\mathcal{R}_i) = \max(\mathcal{R}_{i-1}) + a_i$ and $\min(\mathcal{R}_i) = \min(\mathcal{R}_{i-1}) + a_i$. Hence, using arguments similar to those given in Proposition 1.2.1, we can prove that the Q property holds and that the model is reduced.

Setting values $a_1 \div a_n$, we can model widely subsets of increasing choice functions. If $a_1 = a_2 = ... = a_n = r$, then the model

$$<< \mathcal{T}_i >, \ 1 \leq i \leq m; \ W : \ h(i) \geq h(i-1) + r; \ \{h\}_{\mathcal{T}_i} > \tag{1.9}$$

is an instance of model 1.8, where $\mathcal{T}_i = \{1 + r.(i-1), 1 + r.(i-1) + 1, ..., n - (r+1).(m-i)\}$. Model 1.9 is very convenient for handling, so we will also use it as a foundation for more general considerations given in Chapter 4 concerning model 1.8.

1.2.2 Monotonic choice functions

Monotonic choice functions of indexed families are also used for modelling sets of combinatorial objects. Usually, we use non-decreasing choice functions, however, non-increasing choice functions could also be used.

Given is the indexed family $< \mathcal{E}_i >$, $1 \leq i \leq m$, where $\mathcal{E}_i = \{1, 2, ..., n\}$.

$$W : h(i) \geq h(i-1) \tag{1.10}$$

The model

$$<< \mathcal{E}_i >, 1 \leq i \leq m; W \text{ given in (1.10)}; \{h\}_{\mathcal{E}_i} > \tag{1.11}$$

represents the set of non-decreasing choice functions of the family $< \mathcal{E}_i >$, $1 \leq i \leq m$.

The indexed family $< \mathcal{E}_i >$ is the maximal indexed family for the requirement W given in (1.10). It is known that the above model satisfies the Q property [19]. The set of increasing choice functions of the family $< \mathcal{A}_i >$, $1 \leq i \leq m$ is a subset of the set of non-decreasing choice functions of the indexed family $< \mathcal{E}_i >$, $1 \leq i \leq m$.

Let $< \mathcal{R}_i >$, $1 \leq i \leq m$, be the indexed family, where $\mathcal{R}_i$ is a subset of $\mathcal{E}_i$ that contains $\max(\mathcal{E}_i)$. Then, the model

$$<< \mathcal{R}_i >, 1 \leq i \leq m; W \text{ given in (1.10)}; \{h\}_{\mathcal{R}_i} > \tag{1.12}$$

corresponds to a subset of the set of non-decreasing choice functions. This model satisfies the Q property. Moreover, if $\min(\mathcal{R}_i) > \min(\mathcal{R}_{i-1})$, $2 \leq i \leq m$, then the model is reduced. The arguments are similar to those given in the previous section.

Consider now the following cases of the additional requirement W_1.

$$h(z) \neq q, \text{where } q \neq \max(\mathcal{E}_z), z \in \{1, 2, ..., m\}. \tag{1.13}$$

$$h(z) = \max(\mathcal{E}_z), z \in \{1, 2, ..., m\}. \tag{1.14}$$

$$h(z) = h(z-1), z \in \{1, 2, ..., m\}. \tag{1.15}$$

$$h(z) - h(z-1) \neq q, z \in \{1, 2, ..., m\}, q \neq 0. \tag{1.16}$$

If W_1 is given in (1.13) or (1.14) or (1.15) or (1.16), then the model $<< \mathcal{E}_i >$, $1 \leq i \leq m; W$ given in (1.10); W_1; $\{h\}_{\mathcal{E}_i} >$ satisfies the Q property.

If we replace the indexed family $< \mathcal{E}_i >$, $1 \leq i \leq m$, with the indexed family $< \mathcal{R}_i >$, $1 \leq i \leq m$, then for W_1 given in (1.14) the Q property is still satisfied. Testing satisfiability of the Q property for models containing

any indexed family $< \mathcal{R}_i >$, $1 \leq i \leq m$ and W_1 given in (1.13) or (1.15) or (1.16) requires individual considerations.

Given is the indexed family $< \mathcal{T}_i >$, $1 \leq i \leq m$, where $\mathcal{T}_i = \{r.(i-1)+1, r.(i-1)+2, ..., n-r.(m-i)\}$, $1 \leq i \leq m$; $r \in \{0, 1, 2, ..., \text{int}(n/m)\}$. The requirement W is specified as follows:

$$W : h(i) \geq h(i-1) + r. \tag{1.17}$$

Consider now the following general model:

$$<< \mathcal{T}_i >, 1 \leq i \leq m;\ W \text{ given in (1.17)};\ \{h\}_{\mathcal{T}_i} > \tag{1.18}$$

The choice functions $h \in \{h\}_{\mathcal{T}_i}$ satisfy $h(i) \geq h(i-1)+r$, $2 \leq i \leq m$. The set of choice functions $\{h\}_{\mathcal{T}_i}$ is a subset of the set $\{h\}_{\mathcal{E}_i}$ of non-decreasing choice functions according to the model (1.11). For proving satisfiability of the Q property by the model (1.18), we observe that setting $h(i) = n - r.(m-i)$ that is the maximum of $\mathcal{T}_i$, $1 \leq i \leq m$, we get $h(i) \geq h(i-1) + r$ for every $i \in \{1, 2, ..., m\}$. Hence, the model satisfies the Q property. On the other hand $r.(i-1) + 1 \geq r.((i-1)-1) + 1 + r$. Therefore, every $q \in \mathcal{T}_i$ can be assigned to i in order to create a choice function $h \in \{h\}_{\mathcal{T}_i}$. Hence, the model is reduced.

The model (1.18) presents certain generalization of increasing choice functions and non-decreasing choice functions. If $r = 0$, then the set of choice functions $\{h\}_{\mathcal{T}_i}$ is equivalent to the set of non-decreasing choice functions. If $r = 1$, then the corresponding set $\{h\}_{\mathcal{T}_i}$ is equivalent to the set of increasing choice functions.

1.2.3 Choice bijections

Given is the indexed family $< \mathcal{E}_i >$, $1 \leq i \leq m$, $\mathcal{E}_i = \{1, 2, ..., n\}$, $m \leq n$ as it was specified previously. The requirement

$$W : h(i) \neq h(i-1), h(i) \neq h(i-2), ..., h(i) \neq h(1). \tag{1.19}$$

The model

$$<< \mathcal{E}_i >, 1 \leq i \leq m; W \text{ given in (1.19)}; h \in \{h\}_{\mathcal{E}_i} > \tag{1.20}$$

represents a set of choice bijections of the indexed family $< \mathcal{E}_i >$, $1 \leq i \leq m$. If $n = m$, then the model represents the full set of permutations of n that is called the group of symmetry and denoted by $Sym(n)$.

Consider now restrictions of the set $\{h\}_{\mathcal{E}_i}$ replacing one or more indexed sets $\mathcal{E}_i$ by their subsets $\mathcal{S}_i$. Then, we have the restricted model

$$<< \mathcal{S}_i >, 1 \leq i \leq m; W \text{ given in (1.19)}; \{h\}_{\mathcal{S}_i} > . \qquad (1.21)$$

We are concerned with satisfiability of the Q property for such restricted models. Let $< \mathcal{S}_i >$, $1 \leq i \leq m$, $\mathcal{S}_i \subseteq \{1, 2, ..., n\}$, $m \leq n$, be the reduced family [19] for the choice bijections.

Proposition 1.2.2 *If for every* $i \in \{1, 2, ...m\}$ *at least one of the following two conditions hold*
(i) $| \mathcal{S}_i | \geq i$
(ii) $\mathcal{S}_i - \{\mathcal{S}_1 \cup \mathcal{S}_2 \cup ..., \mathcal{S}_{i-1}\} \neq \emptyset$,
then the model (1.21) satisfies the Q property.

Proof. We prove by observing that for any partial choice mapping h of $< \mathcal{S}_i >$, $1 \leq i \leq i-1$, we can always select a value $h(i) \in \mathcal{S}_i$ that (1.19) holds, if (i) or (ii) holds.

□

We will study now in more detail conditions for satisfiability of the Q property and conditions for the model (1.21) to be reduced.

Let $J \subset \{1, 2, ..., m\}$. The indexes contained in J are ordered increasingly, so we use also denotation $J = \{j_1, j_2, ..., j_r\}$.

Definition 1.2.2 *The indexed family* $< \mathcal{S}_j >$, $j \in J$, *is a closed subfamily of the family* $< \mathcal{S}_i >$, $1 \leq i \leq m$, *if* $| \mathcal{S}_{j_1} \cup \mathcal{S}_{j_2} \cup ... \cup \mathcal{S}_{j_r} | = | J |$.

Definition 1.2.3 *If the indexed family* $< \mathcal{S}_i >$, $1 \leq i \leq m$ *possess a closed subfamily and if for any indexed set* $\mathcal{S}_i$ *that* $i \notin J$ *we have* $(\mathcal{S}_{j_1} \cup \mathcal{S}_{j_2} \cup ... \cup \mathcal{S}_{j_r}) \cap \mathcal{S}_i \neq \emptyset$, *then the set* $\mathcal{S}_i$ *is called a successor if* $i > j_r$ *while* $\mathcal{S}_i$ *is called a predecessor if* $i < j_r$.

Proposition 1.2.3 *The indexed family* $< \mathcal{S}_i >$, $1 \leq i \leq m$, *is the reduced indexed family, if and only if, it does not possess the successor.*

Proof. If $\mathcal{S}_i$ is a successor, then there is an element q that it belongs to at least one indexed set $\mathcal{S}_{j_1}$ or $\mathcal{S}_{j_2}$ or ... or $\mathcal{S}_{j_r}$ and to the set $\mathcal{S}_i$. Since $| \mathcal{S}_{j_1} \cup \mathcal{S}_{j_2} \cup ... \cup \mathcal{S}_{j_r} | = | J |$, so all the elements of the set $\mathcal{S}_{j_1} \cup \mathcal{S}_{j_2} \cup ... \cup \mathcal{S}_{j_r}$ are assigned to the set of indexes J. Therefore, q could not be assigned to the index i. Hence, the set $\mathcal{S}_i$ is not reduced. That finishes the proof of the necessary condition.

We are concerned now with the proof of the sufficient condition. If the indexed family does not possess the successor $\mathcal{S}_i$, then it does not possess any closed subfamily that $j_r < m$. Therefore, for any index i there is not any subset of $\mathcal{S}_i$ being codomain of all partial mappings $h(1)$, $h(2)$, ..., $h(i-1)$. for the subfamily $< \mathcal{S}_k >$, $1 \leq k \leq i-1$ of the family $< \mathcal{S}_i >$, $1 \leq i \leq m-1$, i.e., $\mathcal{S}_k = \mathcal{S}_i$ for $k = i$. Therefore, each $q \in \mathcal{S}_i$ can be assigned

to i that $h(i) = q$ and $h \in \{h\}_{\mathcal{S}_i}$. Hence, there is no q and there is no $\mathcal{S}_i$ that $q \in \mathcal{S}_i$ and q is redundant in $\mathcal{S}_i$. Therefore, the set $\mathcal{S}_i$ is not redundant.

Consider now the case of existence of a closed subfamily that $j_r = m$. Then, for each $q \in \mathcal{S}_m$ there is a choice function $h \in \{h\}_{\mathcal{S}_i}$ that $h(m) = q$. That finishes the proof of the sufficient condition.

□

Proposition 1.2.4 *The indexed family* $< \mathcal{S}_i >$, $1 \leq i \leq m$, *satisfies the Q property if and only if it does not possess the predecessor.*

Proof. We will prove now the necessary condition.

If exists a predecessor $\mathcal{S}_i$, then exists $q \in \mathcal{S}_i$ and $q \in \mathcal{S}_{j_1} \cup \mathcal{S}_{j_2} \cup ... \cup \mathcal{S}_{j_r}$, where $< \mathcal{S}_{j_1}, \mathcal{S}_{j_2}, ..., \mathcal{S}_{j_r} >$ is the subfamily for the set of indexes $J = \{j_1, j_2, ..., j_r\}$. Since $h(i)$ can equal q, so all values of the set $\mathcal{S}_{j_1} \cup \mathcal{S}_{j_2} \cup ... \cup \mathcal{S}_{j_r}$ belong to the codomain of each partial choice mapping $h(1), h(2), ..., h(i) = q, ..., h(j_r - 1)$. Hence, no value could be assigned to the index j_r. Therefore, the Q property does not hold. We conclude that existence of a predecessor $\mathcal{S}_i$ effects that the Q property does not hold. Hence, the necessary condition for satisfiability of the Q property is lack of the predecessor $\mathcal{S}_i$.

We are concerned now with the sufficient condition.

If there is no predecessor $\mathcal{S}_i$, then no closed subfamily of the family $< \mathcal{S}_i >$, $1 \leq i \leq m$, exists. Then, for each index i and for each partial mapping $h(1), h(2), ..., h(i-1)$ there is a value $q \in \mathcal{S}_i$ that q does not belong to the codomain of the partial mapping. Hence, $h(i)$ can equal q. Therefore, the Q property holds. That finishes the proof.

□

The last two propositions lead to exponential algorithms for testing satisfiability of the Q property and for testing whether the indexed family is reduced. We can diminish time complexity of these algorithms by the following observations:

(i) if $\mathcal{S}_i = \mathcal{E}_i$, then $\mathcal{S}_i$ does not belong to any closed subfamily of the family $< \mathcal{S}_i >$, $1 \leq i \leq m$,

(ii) if $\mathcal{S}_i = \mathcal{E}_i$, then $\mathcal{S}_i$ is not any predecessor neither any successor.

Basing on the above observation, we see that it is needed to consider only collections of indexed sets $\mathcal{S}_i$ that $\mathcal{S}_i \subset \mathcal{E}_i$. It means that for numerous instances of the model (1.21) the test for satisfiability of the Q property and for the model be reduced can be done in polynomial time.[2]

We need to mention here that the representation of permutations by choice bijections was investigated by Mirsky [49].

---

[2]It is interesting to observe that Hall's theorem [49], [14], [15] on matching problem could be expressed using the concepts of reducibility and Q property.

1.3 Hierarchical models S_U

We are concerned now with hierarchical unranking representation models S_U. Hierarchical representation models are developed in order to replace a complex requirement W with a simpler one. We can make a hierarchical model also in order to make assertion that the Q property would be satisfied if it does not hold for a given model. Moreover, we develop hierarchical systems S_U in order to manage parallel or distributed generation of a set of choice functions.

Let the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$ be given. With denotation J, we mean a set of indices $J = \{j_1, j_2, ..., j_p, ..., j_r\}$, $J \subset \{1, 2, ..., m\}$. Correspondingly, we have the indexed subfamily $< \mathcal{G}_j >$, $j \in J$, of the family $< \mathcal{G}_i >$, $1 \leq i \leq m$, that $\mathcal{G}_{j_p} = \mathcal{G}_i$ if $j_p = i$. With denotation f, we mean a choice function of the family $< \mathcal{G}_j >$, $j \in J$. The set of all the choice functions f satisfying the requirement W^f is denoted with $\{f\}_{\mathcal{G}_j}$.

The hierarchical S_U is a collection of the following systems.

$$\begin{cases} \textit{System for } \mathcal{J} = \{J\} \textit{ generation}, \\ << \mathcal{G}_j >,\ j \in J; W^f;\ \{f\}_{\mathcal{G}_j} >, \textit{for each } J \in \mathcal{J}, \\ << \mathcal{G}_i >, 1 \leq i \leq m; W;\ \{h\}_{\mathcal{G}_i} >, \end{cases} \tag{1.22}$$

With denotation h^f we mean a choice function $h \in \{h\}_{\mathcal{G}_i}$ that is simultaneously an extension of f to h. Then, to each choice function $f \in \{f\}_{\mathcal{G}_j}$ corresponds a set of choice functions $\{h^f\}$ that $\{h^f\} \subseteq \{h\}_{\mathcal{G}_i}$. With denotation f_k we mean any choice function $f \in \{f\}_{\mathcal{G}_j}$, where $1 \leq k \leq z$, $z = |\{f\}_{\mathcal{G}_j}|$. So, we have sets of choice functions $\{h^{f_1}\}$, $\{h^{f_2}\}$, ..., $\{h^{f_z}\}$ that are extensions of the choice functions f_1, f_2, ..., f_z, accordingly. Observe, the second line contains a number of systems equal to $|\mathcal{J}|$. Nevertheless, it can happen that $|\mathcal{J}| = 1$.

Definition 1.3.1 *We say that the set of choice functions $\{f\}_{\mathcal{G}_j}$ creates a partition on the set $\{h\}_{\mathcal{G}_i}$ if $\{h^{f_1}\} \cup \{h^{f_2}\} \cup ... \cup \{h^{f_z}\} = \{h\}_{\mathcal{G}_i}$ and if $\{h^{f_s}\} \cap \{h^{f_t}\} = \emptyset$ for any two indices s, t that $1 \leq s \leq z$, $1 \leq t \leq z$.*

Creation of a partition on the set $\{h\}_{\mathcal{G}_i}$ is an important property of the hierarchical model S_U. The requirement W^f is specified with the following general form $W^f : f(j_p)\ \mathrm{R}\ \mathrm{E}[f(j_1, f(j_2, ..., f(j_{p-1}), j_1, j_2, ..., j_p], 1 \leq p \leq r$.

Definition 1.3.2 *We say that the requirement W^f is a restriction of the requirement W if for every $f \in \{f\}_{\mathcal{G}_j}$ there exists $h \in \{h\}_{\mathcal{G}_i}$ that $h(j_p) = f(j_p)$ and $h(j_{p-1}) = f(j_{p-1})$.*

For instance, if $W : h(i) \geq h(i-1)$, then the requirement $W^f : f(j_p) \geq f(j_{p-1}) + (j_p - j_{p-1} - 1)$ is a restriction of the requirement W. Making the requirement W^f as the restriction of the requirement W is relatively a

simple task. We take the requirement W and for each pair $(h(j_{p-1}), h(j_p))$ the relation R is to be specified by the form $h(j_p)$ R E$[h(j_1, h(j_2, ..., h(j_{p-1}),$ $j_1, j_2, ..., j_p\ ]$, $1 \le p \le r$. Replacing the symbol h with f, we get W^f that is the restriction of the requirement W. It is obvious that for the hierarchical system given in (1.22) W^f must be a restriction of the requirement W, otherwise we have no assertion that $\{h^{f_1}\} \cup \{h^{f_2}\} \cup ... \cup \{h^{f_s}\} = \{h\}_{\mathcal{G}_i}$ holds. On the other hand making W^f as a restriction of the requirement W is not sufficient for assertion that $\{h^{f_s}\} \cap \{h^{f_t}\} = \emptyset$ holds for any two indices s, t. Therefore, making W^f as a restriction of the requirement W is not sufficient for assertion "$\{f\}_{\mathcal{G}_j}$ creates a partition on the set $\{h\}_{\mathcal{G}_i}$".

Consider now the hierarchical model S_U as follows:

$$\begin{cases} J = \{1, 2, ..., r\},\ r < m \\ << \mathcal{G}_j >,\ j \in J; W^f(j);\ \{f\}_{\mathcal{G}_j} >, \\ << \mathcal{G}_i >, 1 \le i \le m; W; h \in \{h\}_{\mathcal{G}_i} >, \end{cases} \tag{1.23}$$

where $W^f(j)$ is the restriction of the requirement W. It is obvious that if the Q property holds for the system $<< \mathcal{G}_i >$, $1 \le i \le m$; W; $\{h\}_{\mathcal{G}_i} >$, then it holds also for the system $<< \mathcal{G}_j >$, $j \in J$; $W^f(j)$; $\{f\}_{\mathcal{G}_j} >$.

Proposition 1.3.1 *If for the system $S_U = << \mathcal{G}_i >, 1 \le i \le m; W(i); h \in \{h\}_{\mathcal{G}_i} >$ the Q property holds, then the set of choice functions $\{f\}_{\mathcal{G}_j}$ of the system S_U given in (1.23) creates a partition on the set $\{h\}_{\mathcal{G}_i}$.*

Proof. Each choice function $f \in \{f\}_{\mathcal{G}_j}$ is a partial mapping for $< \mathcal{G}_i >$, $1 \le i \le m$. Since the Q property holds, so each $f \in \{f\}_{\mathcal{G}_j}$ could be extended up to a full choice function $h \in \{h\}_{\mathcal{G}_i}$. Two choice functions f_p and f_q that $f_p = f_q$ are the distinct prefixes of two sets of choice functions $\{h^{f_p}\}$ and $\{h^{f_q}\}$. Hence, $\{h^{f_p}\} \cap \{h^{f_q}\} = \emptyset$ for any p, q.

On the other hand each choice function h possess a prefix $f \in \{f\}_{\mathcal{G}_j}$. Therefore, we have assertion that $\{h^{f_1}\} \cup \{h^{f_2}\} \cup ... \cup \{h^{f_s}\} = \{h\}_{\mathcal{G}_i}$. That finishes the proof.

□

Observe, each choice function $f \in \{f\}_{\mathcal{G}_j}$ of the hierarchical representation model given in (1.23) can be identified with the following non−void layer of the indexed family $< \{f(1)\}, \{f(2)\}, ..., \{f(t)\}, < \mathcal{G}_i >>$, $t+1 \le i \le m$. Moreover, the representation model (1.23) can be identified with splitting indexed family into special layers. The hierarchical models given in (1.22) that $\{f\}_{\mathcal{G}_j}$ creates a partition on the set $\{h\}_{\mathcal{G}_i}$ can not be always identified with splitting the indexed family $< \mathcal{G}_i >, 1 \le i \le m$, into layers. That happens since for a pair (f_1, f_2) their domains can differ. Consequently, an index j that belongs to domain of f_1 may not belong to domain of f_2. So, we would have the indexed families $< ...\{f_1(j)\}... >, 1 \le i \le m$ for the function f_1 and $< ...\mathcal{G}_j... >, 1 \le i \le m$ for the function f_2. Because

$\{f_1(j)\} \subset \mathcal{G}_j$, hence the indexed families $< ...\{f_1(j)\}... >, 1 \leq i \leq m$ and $< ...\mathcal{G}_j... >, 1 \leq i \leq m$ can not be the products of splitting the indexed family $< \mathcal{G}_i >, 1 \leq i \leq m$ into layers. The given result can be generalized if we let $\mid \mathcal{J} \mid = 1$. Then, the assumption $J = \{1, 2, ..., r\}$ is not needed for proving that $\{f\}_{\mathcal{G}_j}$ creates a partition on the set $\{h\}_{\mathcal{G}_i}$. Generalization of the above given proposition seems to be rather obvious.

On the other hand there are the general models (1.22) that $\mid \mathcal{J} \mid > 1$ and the hierarchical system used can not be identified with partition of the indexed family into layers, while the sets of choice functions $\{f\}_{\mathcal{G}_j}$ create a partition on the set $\{h\}_{\mathcal{G}_i}$. We will consider later on such specific models.

1.4 Conclusions

We have investigated the basic unranking models concerning arbitrary sets of choice functions. These basic models concern increasing choice functions, monotonic choice functions and choice bijections. We have shown a number of criteria enabling us to judge whether the models are reduced and whether the Q property holds. The researched classes of models have important applications as to representations of basic sets of combinatorial objects.

2
Basic representations

In this chapter we are concerned with unranking representation models S_U. The goal is to give representation method for arbitrary sets of compositions or partitions of a number and decompositions or partitions of a set by the corresponding sets of choice functions of indexed families. The main philosophy is to represent the full sets of combinatorial objects. Then, we will discuss restrictions of the model in order to represent its subsets. The common method is to create requested representation model by restricting the indexed sets and/or by imposing additional requirements W_1. Moreover, representing a set of combinatorial object by a partial choice mapping will also be used. We recall the representation of sets of combinations and permutations given in [19].

2.1 Compositions of numbers

Given is the indexed family $< \mathcal{H}_i >$, $1 \leq i \leq m$, where $\mathcal{H}_i = \{i, i+1, ..., n-m+i\}$, $1 \leq i \leq m-1$ and $\mathcal{H}_m = \{n\}$. We are concerned with the following model

$$<< \mathcal{H}_i >, 1 \leq i \leq m;\ W \text{ given in (1.1)};\ \{h\}_{\mathcal{H}_i} > \qquad (2.1)$$

The requirement W given in (1.1) means that the set of choice functions $\{h\}_{\mathcal{H}_i}$ is the set of increasing choice functions. Observe, we have $\mathcal{H}_i = \mathcal{A}_i$ for $1 \leq i \leq m-1$. It is known that the model containing indexed family $< \mathcal{A}_i >$, $1 \leq i \leq m-1$ and the requirement $W : h(i) > h(i-1)$ satisfies

the Q property. Therefore, the Q property is satisfied for the subfamily $\{\mathcal{H}_i\}$, $1 \leq i \leq m-1$. Since $\max(\mathcal{H}_{m-1}) = n-1$ and $\mathcal{H}_m = \{n\}$, so $h(m) > h(m-1)$. Hence, the model (2.1) satisfies the Q property.

Let $\{n_1, n_2, ..., n_m\}$ be a composition of the number n into m elements that $n_j \neq 0$, $1 \leq j \leq m$ and $n_1 + n_2 + ... + n_m = n$.

There is correspondence between each composition of n and the choice function $h \in \{h\}_{\mathcal{H}_i}$ as follows:

$$n_i = h(i) - h(i-1), 1 \leq i \leq m;\ h(0) = 0. \tag{2.2}$$

Since $\sum_{i=1}^{m}(h(i) - h(i-1)) = n$, and two distinct compositions that differ at least by two components n_i and n_{i+1} correspond to the choice functions h_1 and h_2 that $h_1(i) \neq h_2(i)$, so the correspondence between the choice functions and the compositions is one–to–one. For a given set of numbers $\{n_1, n_2, ..., n_m\}$ the corresponding choice function can be evaluated recursively using the following top–down procedure.

for $i \longleftarrow m$ **down to** 2 **do**

$h(i-1) = h(i) - n_i$;

Example 2.1.1 *For $n = 7$ and $m = 4$, we can represent a composition of n by the corresponding choice function $h \in \{h\}_{\mathcal{H}_i}$.*

$$\mathcal{H}_1 = \{1,2,3,4\}, \mathcal{H}_2 = \{2,3,4,5\}, \mathcal{H}_3 = \{3,4,5,6\}, \mathcal{H}_4 = \{7\}$$

For instance the choice function h:1 $\longrightarrow$ 3, 2 $\longrightarrow$ 4, 3 $\longrightarrow$ 6, 4 $\longrightarrow$ 7 represents the following composition $n_1 = 3$, $n_2 = 1$, $n_3 = 2$, $n_4 = 1$, $n = n_1 + n_2 + n_3 + n_4$.

For the composition {2, 1, 2, 2} we have the following choice function h : $h(4) = 7$, $h(3) = 5$, $h(2) = 3$, $h(1) = 2$.

□

We can also represent certain generalization of the given compositions enabling $n_i = 0$. Then, we use for modelling the non-decreasing choice functions of the indexed family $< \mathcal{Z}_i >$, $1 \leq i \leq m$, where $\mathcal{Z}_i = \{0,1,2,...n\}$ for $1 \leq i \leq m-1$ and $\mathcal{Z}_m = \{n\}$.

The model

$$<< \mathcal{Z}_i >, 1 \leq i \leq m; W \text{ given in (1.10)}; \{h\}_{\mathcal{Z}_i} > \tag{2.3}$$

represents the set of compositions $\{n_1, n_2, ..., n_m\}$ that $n_i = 0$ is enabled.

The Q property for this model is satisfied since for every index i, $\max(\mathcal{Z}_i) = n$. Hence, we have assertion that $h(i) \geq h(i-1)$.

If the order $<^n$ on the set of compositions is defined as the lexical order of m numbers $\{n_1, n_2, ..., n_m\}$ and the order $<^h$ on the corresponding set

of choice functions is defined as the natural lexical order of $m-1$ numbers $\{h(1), h(2), ..., h(m-1)\}$, then these two orders are in accordance, i.e., if there are correspondences $\{h^1(1), h^1(2), ..., h^1(m-1)\} \rightarrow \{n_1^1, n_2^1, ..., n_m^1\}$ and $\{h^2(1), h^2(2), ..., h^2(m-1)\} \rightarrow \{n_1^2, n_2^2, ..., n_m^2\}$, then $\{h^1(1), h^1(2), ..., h^1(m-1)\} <^h \{h^2(1), h^2(2), ..., h^2(m-1)\} \Longleftrightarrow \{n_1^1, n_2^1, ..., n_m^1\} <^n \{n_1^2, n_2^2, ..., n_m^2\}$.

We now consider modelling subsets of compositions. Consider sets of compositions represented by (2.1). The requirements W_1 given in (1.3) or (1.4) or (1.5) or (1.6) can be used for restricting $\{h\}_{\mathcal{H}_i}$. Selection of the specified requirement W_1 corresponds to the following subsets of compositions.

If W_1 is selected according to (1.3), then the corresponding set of compositions satisfies additionally: the sum $n_1+n_2+...+n_z \neq q$, and consequently $n_{z+1} + n_{z+2} + ... + n_m \neq n-q$.

If W_1 is selected according to (1.4), then the corresponding set of number compositions satisfies additionally: the sum $n_1 + n_2 + ... + n_z = \max(\mathcal{H}_z)$ and consequently $n_{z+1} + n_{z+2} + ... + n_m = n - \max(\mathcal{H}_z)$.

If W_1 is selected according to (1.5), then the corresponding set of number compositions satisfies additionally: $n_z = 1$.

If W_1 is selected according to (1.6), then the corresponding set of number compositions satisfies additionally: $n_z \neq q$, where $q \neq 1$.

We know basing on the properties of the restricted models concerning increasing choice functions that the Q property holds.

We are concerned now with the sets of compositions represented by the model (2.3).

If W_1 is selected according to (1.15), then the corresponding set of compositions satisfies additionally: $n_z = 0$.

If W_1 is selected according to (1.16), then the corresponding set of compositions satisfies additionally: $n_z \neq q$, where $q \neq 0$.

If W_1 is selected according to (1.13) or (1.14), then the compositions are similar to those specified by the requirements given in (1.3) or (1.4), respectively.

For the corresponding restricted models the Q property holds.

Example 2.1.2 *Represent the set of the compositions that* $n_2 \neq 2$, *and* $n_3 = 0$.

If we accept "n_i can equal 0 for $i \neq 2$", then the requirement W_1 is conjunction of the instances of the requirements (1.16) and (1.15). We specify W_1 in (2.4).

$$h(2) - h(1) \neq 2,\ h(3) - h(2) = 1 \tag{2.4}$$

For instance, the following function h: $1 \longrightarrow 3$, $2 \longrightarrow 3$, $3 \longrightarrow 4$, $4 \longrightarrow 7$ satisfies additionally (2.4) and it corresponds to the following composition of number 7, $n_1 = 3$, $n_2 = 0$, $n_3 = 1$, $n_4 = 3$.

□

The model (1.8) can also be adopted for modelling sets of compositions. Given is the indexed family $< \mathcal{R}_i >$, $1 \leq i \leq m$, that $\mathcal{R}_i = \{ 1+a_2 +a_3 + ... + a_i, 1 + a_2 + a_3 + ... + a_i + 1 ,..., n - m - a_n - a_{n-1} - ... - a_i + i \}$ for $i \leq m-1$ and $\mathcal{R}_m = \{n\}$. Then, the model

$$<< \mathcal{R}_i >, 1 \leq i \leq m; W : h(i) > h(i-1) + a_i, \ a_i \in \{0, 1, ...\}; \ \{h\}_{\mathcal{R}_i} >$$

represents the set of compositions that $n_i \geq a_i$. Since, this model satisfies the Q property for $1 \leq i \leq m-1$ and $\max(\mathcal{R}_{m-1}) + a_m = n$, so the Q property holds also for $i = m$.

Consider now the following general model:

$<< \mathcal{T}_i >$, $1 \leq i \leq m-1$, $\mathcal{T}_i = \{r.(i-1)+1, r.(i-1)+2, ..., n - r.(m-i)\}$, $\mathcal{T}_m = \{n\}$; W given in (1.17); where $r \in \{0, 1, 2, ..., int(n/m)\}$.

Using the form $n_i = h(i) - h(i-1)$ each $h \in \{h\}_{\mathcal{T}_i}$ represents a composition of n into m integers such that $n_i \geq r$. We know (Section 1.2.1.) that for the subfamily $< \mathcal{T}_i >$, $1 \leq i \leq m-1$ the Q property holds. Since, the maximal value taken by $h(m-1) = n - r$ and $h(m) = n$, so we have assertion that $h(m) - h(m-1) \geq r$. Hence, the Q property holds.

If we do not care about satisfiability of the Q property, then we can model the requirement W_1 in a very flexible manner. For instance,

$$W_1 : \ h(q) - h(q-1) < h(q+1) - h(q) \tag{2.5}$$

Then, each choice function $h \in \{h\}_{\mathcal{Z}_i}$ of the model

$$<< \mathcal{Z}_i >, 1 \leq i \leq m; W \text{ given in (1.10)}; W_1 \text{ given in (2.5)}; \ \{h\}_{\mathcal{Z}_i} > \tag{2.6}$$

corresponds to a composition such that $n_q < n_{q+1}$, where $q \in \{2, 3, ..., m\}$. The Q property does not hold for that model.

2.2 Partitions of numbers

A composition of a number n that $n_{i+1} \geq n_i$, $1 \leq i \leq m-1$ is the partition of the number n. Therefore, a set of choice functions representing a given set of partitions is a subset of the set of choice functions representing compositions. We will develop the model of partitions basing on the model (2.1).

Let $n = m.a + r$, while $b = \begin{cases} r - (m-i), \text{ if it is positive} \\ 0 \text{ if } r - (m-i) \text{ is negative} \end{cases}$

Given is the indexed family $< \mathcal{M}_i >$, $1 \leq i \leq m$, where $\mathcal{M}_i = \{i, i+1, ..., a.i+b\}$ for $1 \leq i \leq m-1$ and $\mathcal{M}_m = \{n\}$. The following requirements are investigated.

$$h(i) - h(i-1) \geq h(i-1) - h(i-2) \tag{2.7}$$

$$h(i) \leq \frac{n - h(i-1).(m-i)}{m-i} \tag{2.8}$$

We have the model

$$<< \mathcal{M}_i >, 1 \leq i \leq m; W \text{ equals} (1.1) \wedge (2.7) \wedge (2.8); \{h\}_{\mathcal{M}_i} > . \tag{2.9}$$

With denotation W equals $(1.1) \wedge (2.7) \wedge (2.8)$, we mean that the requirement W is conjunction of (1.1) and (2.7) and (2.8).

Proposition 2.2.1 *Any choice function* $h \in \{h\}_{\mathcal{M}_i}$ *represents a partition, that* $n_i = h(i) - h(i-1)$, $1 \leq i \leq m-1$, *where* $h(0) = 0$. *The model is reduced and the Q property holds.*

Proof. The requirement (1.1) satisfied makes assertion that the corresponding sets of partitions is a subset of the set of compositions.

If the requirement (2.7) holds, then we have assertion $n_i \geq n_{i-1}$. Hence, the requirement (2.7) satisfied makes assertion that the components n_{i-1} and n_i are enumerated following the non-decreasing order of their values. The requirement (2.8) satisfied for i makes assertion that the requirement (2.7) can be satisfied for $j > i$. Hence, the requirement (2.8) is an auxiliary one enabling satisfiability of the Q property by a system that contains the requirement (2.7). We observe it noting that for given $h(i)$ and for given $h(i-1)$ the corresponding component of the partition is n_i. Then, $h(i) \leq \frac{n-h(i-1).(m-i)}{m-i}$ satisfied makes assertion that $\sum_{j=i+1}^{m}(n_j) \geq n_i.(m-i)$. If it holds, then we can create the corresponding partition that the components n_j can be not smaller than the component n_i.

The requirement $h(i) - h(i-1) \geq h(i-1) - h(i-2)$ satisfied, makes assertion that $n_i \geq n_{i+1}$.

For proving that the model is reduced, we observe that assigning $h(i) = \max(\mathcal{M}_i)$ we get the corresponding partition whose rank according to the lexical order is maximal, while assigning $h(i) = \min(\mathcal{M}_i)$, we get the partition that its rank is minimum. All values of $\mathcal{M}_i$ between $\min(\mathcal{M}_i)$ and $\max(\mathcal{M}_i)$ can also be assigned into i, in order to make a choice function h.

□

Example 2.2.1 *For $n = 9$ and $m = 3$ the task is to represent all the partitions.*

We use the model given in (2.9).

The indexed family is as follows: $M_1 = \{1, 2, 3\}$, $M_2 = \{2, 3, 4, 5, 6\}$, $M_3 = \{9\}$, since $a = 9/3 = 3$ and $b = 0$ for every i.

We have the following choice functions satisfying the requirement W and the corresponding partitions.

No	$h(1)$	$h(2)$	$h(3)$	n_1	n_2	n_3
1	1	2	9	1	1	7
2	1	3	9	1	2	6
3	1	4	9	1	3	5
4	1	5	9	1	4	4
5	2	4	9	2	2	5
6	2	5	9	2	3	4
7	3	6	9	3	3	3

The choice functions are listed following the lexical order and the corresponding partitions are listed accordingly.

□

Consider modelling subsets of the partitions.

We are concerned now with the representation model for the set of partitions that $n_i \leq z$, where $z > a$. Then, the choice functions have to satisfy the additional requirement W_1 given in (2.10).

$$W_1 : h'(i) - h'(i-1) \leq z \tag{2.10}$$

The reduced indexed family $< \mathcal{M}'_i >$, $1 \leq i \leq m$, has to satisfy the conditions:

(i) $\max(\mathcal{M}'_i) - \max(\mathcal{M}'_{i-1}) \leq z$,

(ii) $\min(\mathcal{M}'_i) - \min(\mathcal{M}'_{i-1}) \leq z$, $1 \leq i \leq m$.

Since $\max(\mathcal{M}'_i) = a.i + b$ and ($b = r$ or $b = 0$), so we have $\max(\mathcal{M}'_i) = a.i+r$ and $\max(\mathcal{M}'_{i-1}) = a.(i-1)$, for certain value i . Then, $z \geq a.i+r-(a.(i-1)) = r+a$. For other values of i we have to have $z \geq a.i-a(i-1) = a$

Then, the set of choice functions $h' \in \{h'\}_{\mathcal{M}'_i}$ of the model

$$<< \mathcal{M}'_i >, 1 \leq i \leq m; W \text{ equals} (1.1) \wedge (2.7) \wedge (2.8); W_1 = (2.10); \{h'\}_{\mathcal{M}'_i} > \tag{2.11}$$

represents a subset of all the partitions possessing the biggest component n_i not greater than z, where $z \geq a$ or $z \geq a + r$. Since the numbers $\{n_1, n_2, ..., n_m\}$ creating the partitions are ordered increasingly, so we have $n_m = h'(m) - h'(m-1) \leq z$, where $h' \in \{h'\}_{\mathcal{M}'_i}$.

Since $\max(\mathcal{M}'_i) - \max(\mathcal{M}'_i) \leq z$, so
(i) the Q property holds automatically,
(ii) the requirement W_1 given in (2.10) is satisfied automatically.
Therefore, the following model represents the set of partitions that $n_i \leq z$.

$$<< \mathcal{M}'_i >, 1 \leq i \leq m; W \text{ given in } (1.1) \wedge (2.7) \wedge (2.8); \{h'\}_{\mathcal{M}'_i} > . \quad (2.12)$$

2.3 Decompositions into at most m enumerated blocks

We are concerned now with representation of the decomposition of a set $A = \{1, 2, ..., n\}$ into at most m blocks, $m \leq n$. For some blocks no item can be assigned but we have to have at least one non−void block.

Let $< \mathcal{B}_i >$, $1 \leq i \leq n$, be the indexed family, where $\mathcal{B}_1 = \mathcal{B}_2 = ... = \mathcal{B}_n = \{1, 2, ..., m\}$.

Then, we have the representation model

$$<< \mathcal{B}_i >, 1 \leq i \leq n; \{h\}_{\mathcal{B}_i} > . \quad (2.13)$$

Any choice function $h \in \{h\}_{\mathcal{B}_i}$ of the family $< \mathcal{B}_i >$ represents a decomposition of the set A into at most m enumerated blocks, each value $q = h(i)$ has the following interpretation: the i−th item belongs to the q−th block. We have to emphasis that no requirement W is to be satisfied by functions $h \in \{h\}_{\mathcal{B}_i}$. Please note that a value $q \in \{1, 2, ..., m\}$ may not be assigned to any index $i \in \{1, 2, ..., n\}$. It means that no item belongs to the block B_i, so $|B_i| = \emptyset$. Obviously, the Q property holds for this model.

Example 2.3.1 *For instance the distribution* $B_1 = \{1, 2, 3\}$, $B_2 = \emptyset$, $B_3 = \{4\}$, $B_4 = \emptyset$, $B_5 = \{5, 6\}$ *of the set* $\{1, 2, 3, 4, 5, 6\}$ *is represented by the choice function:* $h(1) = 1$, $h(2) = 1$, $h(3) = 1$, $h(4) = 3$, $h(5) = 5$, $h(6) = 6$.

□

We can restrict the set of decompositions into m blocks by replacing any original set $\mathcal{B}_i$ or by modelling the requirement W_1. Since no requirement W has to be satisfied by the choice functions of the family $< \mathcal{B}_i >$, so replacing $\mathcal{B}_i$ by its any non−void subset does not affect the Q property.

Consider now the following requirements W_1 which can be imposed on the set of the choice functions of the family $< \mathcal{B}_i >$:

$$W_1 : h(j) = h(i) \text{ or } h(j) \neq h(i) \;\; j > i \tag{2.14}$$

$$W_1 : h(i) = q \text{ or } h(i) \neq q, \;\; q \in \{1, 2, ..., m\} \tag{2.15}$$

One can prove that for the models

$$<< \mathcal{B}_i >, 1 \leq i \leq n; W_1; \{h\}_{\mathcal{B}_i} > \tag{2.16}$$

the Q property is satisfied if W_1 is given in (2.14) or if W_1 is given in (2.15).

If we combine both a restriction of the set(s) $\mathcal{B}_i$ and a formulation of the requirement W_1, then satisfiability of the Q property is an open problem, since for intersection of these two conditions no value from $\mathcal{B}_i$ can be selected.

Example 2.3.2 *The task is to model the set of all the decompositions of the set* {1, 2, 3, 4, 5, 6} *into at most* 3 *enumerated blocks so that the items* $3-th$ *and* $4-th$ *do not belong to the same block and items* $5-th$ *and* $6-th$ *do not belong to the* $3-th$ *block.*

The requested set of the decompositions can be modeled by the choice functions $\{h\}_{\mathcal{B}_i}$ of the following indexed family $< \mathcal{B}_i >, 1 \leq i \leq 6$; $B_1 = \mathcal{B}_2 = B_3 = \mathcal{B}_4 = \{1, 2, 3\}$, $B_5 = \mathcal{B}_6 = \{1, 2\}$. The requirement W_1 to be satisfied by $\{h\}_{\mathcal{B}_i}$ is as follows: W_1:$h(4) \neq h(3)$ and $h(5) \neq h(6)$. Since $| \mathcal{B}_3 | > 1$, $| \mathcal{B}_4 | > 1$, $| \mathcal{B}_5 | > 1$, $| \mathcal{B}_6 | > 1$, so the Q property holds for the model.

□

2.4 Partitions of a set into at most m blocks

Let the family $< \mathcal{F}_i >$, such that $\mathcal{F}_i = \{1, 2, ..., \min(i, m)\}$, $1 \leq i \leq n$, be given. The requirement W is given in (2.17).

$$h(1) = 1, \text{ for every } i > 1, \; h(i) = j, \text{ only if there exists}$$

$$i_1 < i, \text{so that } h(i_1) = j - 1; j \leq m \tag{2.17}$$

$$<< \mathcal{F}_i >, 1 \leq i \leq n; W \text{ given in (2.17)}; h \in \{h\}_{\mathcal{F}_i} > \tag{2.18}$$

Proposition 2.4.1 *The model given in (2.18) represents the set of all partitions of the set* $A = \{1, 2, ..., m\}$ *into at most* m *subsets.*

Proof. The assignment $h(i) = j$ means that the $i-$th element belongs to the $j-$th block. Two choice functions h_1, h_2 such that $h_1(i) \neq h_2(i)$ for any $i \in \{1, 2, ..., m\}$ give distinct specifications of blocks for given items. In order to make assertion that each partition corresponds uniquely to exactly one choice function $h \in \{h\}_{\mathcal{F}_i}$, we have to prove that the obtained blocks of the partition are enumerated according to the increasing order of their minimum elements.
Given is $j_1 \in \{1, 2, ..., m\}$, let i_1 be the minimum element that $i_1 \in A$ and $h(i_1) = j_1$.
Since $h \in \{h\}_{\mathcal{F}_i}$ satisfies (2.17), so we have assertion that exists $i_2 < i_1$, so that $h(i_2) = j_1 - 1$.

If i_2 is the smallest index for which the above is satisfied, then exists $i_3 < i_2$, so that $h(i_3) = h(i_2) - 1$. Repeating this argument j_1 times, we prove that if $h(i) = j$, then exist arguments i_1, i_2, ..., i_{j-1} such that $i_1 < i_2 < ... < i_{j-1}$ and $h(i_1) = 1$, $h(i_2) = 2$, ..., $h(i_{j-1}) = j - 1$. Therefore, blocks of partitions are enumerated according to the increasing order of their minimum elements. Hence, there is one–to–one correspondence between partition of the set $\{1, 2, ..., m\}$ and a choice function $h \in \{h\}_{\mathcal{F}_i}$.

□

One can easily show that the Q property holds for the above model.

Example 2.4.1 *Show the indexed family* $< \mathcal{F}_i >$ *and a choice function representing a partition of the set* $S = \{1, 2, ..., 7\}$ *into at most* 3 *blocks.*

For this case we have: $\mathcal{F}_1 = \{1\}$, $\mathcal{F}_2 = \{1, 2\}$, $\mathcal{F}_3 = \mathcal{F}_4 = \mathcal{F}_5 = \mathcal{F}_6 = \mathcal{F}_7 = \{1, 2, 3\}$. For instance, the choice function h: $1 \longrightarrow 1$, $2 \longrightarrow 1$, $3 \longrightarrow 2$, $4 \longrightarrow 1$, $5 \longrightarrow 3$, $6 \longrightarrow 2$, $7 \longrightarrow 1$ corresponds to the partition that $B_1 = \{1, 2, 4, 7\}$, $B_2 = \{3, 6\}$, $B_3 = \{5\}$.

□

We can create restricted models by imposing (2.14) or (2.15) as the requirements W_1. Moreover, we can remove any number of upper elements from any indexed set $\mathcal{F}_i$ so that $\mathcal{F}_i$ remains non–void. Each such restricted model satisfies the Q property.

The correspondence between any restricted model and the represented set of partitions is enough obvious.

2.5 Decompositions into exactly m blocks

We are concerned now with the set of decompositions of a given set A into exactly m non–void blocks. If these decompositions were represented

by the set of choice functions of the indexed family $< \mathcal{B}_i >$, $1 \leq i \leq n$, then making assertion that exactly m different values would be taken by the choice function h is essential for this modelling. So, the simple representation models S_U would require investigation of rather composed W. The requirement W has to be especially composed if satisfiability of the Q property is demanded. So, modelling the set of decompositions by a simple S_U model is impractical.

In order to avoid the mentioned earlier difficulties, we use the hierarchical representation model.

Let $< \mathcal{C}_k >$, $1 \leq k \leq m$, where $\mathcal{C}_k = \{1, 2, ..., n\}$, be an indexed family. We create an auxiliary model as follows:

$$<< \mathcal{C}_k >, 1 \leq k \leq m;\ W = (1.19);\ \{p\}_{\mathcal{C}_k} > . \tag{2.19}$$

So, the set of choice functions $\{p\}_{\mathcal{C}_k}$ is the set of choice bijections. The model (2.19) satisfies the Q property because $n > m$. With denotation CD_p we mean codomain of the mapping p. Since each $p(k) \in \{1, 2, ..., n\}$, $1 \leq k \leq m$, so for every p we have $CD_p \subset \{1, 2, ..., n\}$. With each codomain CD_p, we have the set of indexes $J_p = CD_p$, that the cardinal $\mid J_p \mid = m$. With denotation $\{J_p\}$ we mean the collection of all the sets of indexes, each set J_p is obtained for every $p \in \{p\}_{\mathcal{C}_k}$. For each J_p we have an auxiliary indexed family $< \mathcal{B}_j >$, $j \in J_p$ that $B_j = B_i$ if $i = j$, $1 \leq i \leq n$. We create the choice function $f \in \{f\}_{\mathcal{B}_j}$ that $f = p^{-1}$, i.e., the choice function f is the inverse mapping for given p.

$$W \quad : \quad h(i) = q, \text{only if } q \notin J_p \text{ and if} \tag{2.20}$$

$$\text{there exists } s \quad \in \quad J_p, \text{ that } s < i \text{ and } f(s) = q \tag{2.21}$$

For the choice functions $\{h\}_{\mathcal{B}_i}$ of the family $< \mathcal{B}_i >$, $1 \leq i \leq n$, the requirement W given in (2.20) is to be satisfied.

Then, we have the following hierarchical models S_U:

$$< \quad \mathcal{J} = \{J_p\} \text{ is the collection of codomains of } \{p\}_{\mathcal{C}_k} >, \tag{2.22}$$

$$< \quad < \mathcal{B}_k >,\ k \in J_p;\ W^f \text{ given in (1.19)};\ \{f\}_{\mathcal{B}_k} >,\ J_p \in \mathcal{J},$$

$$< \quad < \mathcal{B}_i >,\ 1 \leq i \leq n;\ W \text{ given in (2.20)};\ \{h\}_{\mathcal{B}_i} > .$$

Each choice function $h \in \{h\}_{\mathcal{B}_i}$ is an extension of a choice function $f \in \{f\}_{\mathcal{B}_k}$, the Q property holds. The set of partial mappings f is not the set of choice functions of one indexed family, therefore we have to make assertion that the set $\{f\}_{\mathcal{B}_k}$ makes partition on $\{h\}_{\mathcal{B}_i}$.

Proposition 2.5.1 *If h^{f_s} is an extension of f_s and h^{f_t} is an extension of f_t and if $f_s \neq f_t$, then $h^{f_s} \neq h^{f_t}$.*

Proof. Because $f = p^{-1}$, so for any pair of choice functions (f_s, f_t) their domains differ. Since f is one–to–one mapping and for any pair (f_s, f_t) of mappings their domains are disjoint and codomains are common so exist j_1, j_2 such that $j_1 \neq j_2$ and $f_s(j_1) = f_t(j_2) = q$, $q \in \{1, 2, ...m\}$.

We have to prove now the following lemma:

if $f_s(j_1) = f_t(j_2) = q$ and $j_1 \neq j_2$, then $h^{f_s}(j_2) \neq h^{f_t}(j_1)$.

For $j_1 < j_2$, we have already $h^{f_t}(j_1) \neq q$ and $h^{f_s}(j_2) = q$, hence $h^{f_s}(j_2) \neq h^{f_t}(j_1)$.

For $j_1 > j_2$, we have $h^{f_s}(j_2) \neq q$ and $h^{f_t}(j_1) = q$, hence again $h^{f_s}(j_2) \neq h^{f_t}(j_1)$. It completes the proof.

□

The set $\{f\}_{\mathcal{B}_k}$ of the model $<< \mathcal{B}_k >$, $k \in J_p$; $W = (1.19)$; $\{f\}_{\mathcal{B}_k} >$ represents all the possible assignments of the minimal element into the blocks, so $\{ h^{f_1}\} \cup \{ h^{f_2}\} \cup ... \cup \{ h^{f_z}\} = \{h\}_{\mathcal{B}_i}$. Since, we have assertion that $\{h^{f_s}\} \cap \{h^{f_t}\} = \emptyset$, so the set $\{f\}_{\mathcal{B}_k}$ makes partition on the set $\{h\}_{\mathcal{B}_i}$.

We discuss now the representation issue for the above model. Each choice function $p \in \{p\}_{C_j}$ represents assignment of the blocks to their minimum elements, i.e., if $p(k) = q$, then the element k is the minimum element of the block B_q of a decomposition $<B_1, B_2, ..., B_m>$. The corresponding choice function f represents assignment of the minimum elements to the blocks, accordingly. The choice function h being an extension of a function f represents assignment of elements to the blocks of the decomposition. Satisfiability of the requirement W given in (2.20) makes assertion that minimum elements are assigned to each block according to the function f indeed.

Example 2.5.1 *The task is to represent a decomposition of the set* $\{1 \div 7\}$ *into* 3 *blocks.*

For this case the auxiliary family $< C_j >$ is as follows: $C_1 = C_2 = C_3 = \{1 \div 7\}$.

Any choice bijection p of the family $< C_j >$, $1 \leq j \leq 3$ represents an assignment of a block decomposition to the minimum element of this block. For instance, p:1 $\longrightarrow$ 1, 2 $\longrightarrow$ 5, 3 $\longrightarrow$ 4 assigns minimum elements 1, 5, 4 to the blocks B_1, B_2, B_3, respectively. Then, the partial mapping $f = p^{-1}$ is as follows 1 $\longrightarrow$ 1, 5 $\longrightarrow$ 2, 4 $\longrightarrow$ 3 and $CD_f = \{1, 4, 5\}$. Extending f up to a full choice function of the family $< \mathcal{B}_i >$, $1 \leq i \leq 7$, we have h: 1 $\longrightarrow$ 1, 2 $\longrightarrow$ 1, 3 $\longrightarrow$ 1, 4 $\longrightarrow$ 3, 5 $\longrightarrow$ 2, 6 $\longrightarrow$ 3, 7 $\longrightarrow$ 1. The given choice function h represents the following decomposition: $B_1 = \{1, 2, 3, 7\}$, $B_2 = \{5\}$, $B_3 = \{4, 6\}$.

□

Important property of the modelling by usage of these two indexed families are different numbers of extensions for each f.

Let $\mid \{h_f\}_{\mathcal{B}_i} \mid$ be the number of extensions of a mapping f that $f = p^{-1}$. With denotation $\mid n - p(1) \mid$ we mean the absolute value of the difference $n - p(1)$.
We have $\mid \{h_f\}_{\mathcal{B}_i} \mid = \mid n - p(1) \mid . \mid n - p(2) \mid ... \mid n - p(m) \mid$. If $CD_f = \{1, 2, ...m\}$, then $\mid \{h_f\}_{\mathcal{B}_i} \mid$ is maximum, while for $CD_f = \{n - m, n - m + 1, ..., n\}$ the number of all extensions $\mid \{h_f\}_{\mathcal{B}_i} \mid$ of f is minimum. Taking the formula $\mid \{h_f\}_{\mathcal{B}_i} \mid = \mid n - p(1) \mid . \mid n - p(2) \mid ... \mid n - p(l) \mid$ as the base, we see that the number of extensions of each $f = p^{-1}$ is rather decreasing with the rank of f. Since to a greater rank of p corresponds relatively smaller rank of f, so we say that the number of extensions is increasing with the rank of p, however this growing is not monotonic. This property has important consequences, when the generation issue is discussed.

Consider now modelling subsets of the full set of decompositions. The hierarchical model contains two independent indexed families $< \mathcal{C}_j >$, $1 \leq j \leq q$ and $< \mathcal{B}_i >$, $1 \leq i \leq n$ and their choice functions p and h, respectively. Deforming indexed sets of each indexed family or imposing additional requirements, we can restrict the set $\{p\}_{\mathcal{C}_j}$ and consequently the set of choice functions $\{h\}_{\mathcal{B}_i}$ would be restricted. We can also restrict only the set of choice functions $\{h\}_{\mathcal{B}_i}$ leaving the set of choice functions $\{p\}_{\mathcal{C}_j}$ unchanged.

Distinct subsets of the set of the choice bijections p of the indexed family $< \mathcal{C}_j >, 1 \leq j \leq m$ can be modeled using general rules concerning modelling subsets of the choice bijections.

Since each choice function p represents assignment of the minimum element to each block of a modeled decomposition, so restrictions imposed on p represent the restrictions imposed on the assignment of minimum elements to each particular block. We can check whether restricted models satisfy the Q property following the general rules for subsets of choice bijections.

Example 2.5.2 *Show the representation of the set of decompositions of the set* {1, 2, 3, 4, 5, 6} *into* 3 *non–void blocks, so that the* 3*–th item is not a minimum element of any block and the* 5*–th and* 6*–th elements do not belong to the block* B_3.

The first requirement can be modeled by making proper indexed sets of the family $< \mathcal{C}_j >$, $1 \leq j \leq m$, while the second requirement can be modeled by making proper indexed sets of the family $< \mathcal{B}_i >$, $1 \leq i \leq n$.

So, we have

$\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{C}_3 = \{1, 2, 4, 5, 6, \}$.

$\mathcal{B}_1 = \mathcal{B}_2 = \mathcal{B}_3 = \mathcal{B}_4 = \{1, 2, 3\}$, $\mathcal{B}_5 = \mathcal{B}_6 = \{1, 2\}$.

One can observe that for this example the Q property holds.

□

Consider now the following restrictions W_1 which can be imposed additionally on the choice functions h of the family $< \mathcal{B}_i >$ for a fixed values $j, k \in \{1, 2, ..., m\}$ and $q \in \{1, 2, ..., n\}$.

$$W_1 : h(j) = h(k), \text{ where } k < j \tag{2.23}$$

$$W_1 : h(j) \neq q \tag{2.24}$$

$$W_1 : h(j) \neq h(k), \text{ where } k < j \tag{2.25}$$

$$W_1 : h(j) = q \tag{2.26}$$

If we use (2.23) as W_1, then the Q property holds, while using (2.24), or (2.25) or (2.26) satisfiability of the Q property is an open question and depends on j and q. One can observe that adding requests to the specified requirements W the Q property can hold. We will not discuss that problem since its solution is oriented towards a given task. We have the following correspondence between the particular requirements W_1 and the represented sets of decompositions:

(2.23) an element j must belong to the same block as an element k, where $k< j$,

(2.24) an element j can not belong to the block q,

(2.25) an element j can not belong to the same block as an element k, where $k< j$,

(2.26) an element j must belong to the block q.

2.6 Partitions of a set into exactly m blocks

The full set of partitions of a set into exactly m block is a subset of the full set of the corresponding decompositions. The main request of the modelling is to make assertion that the blocks are enumerated following an increasing order of their minimum elements. The other requirements are the same as for modelling decompositions into exactly m blocks. We use a similar hierarchical representation model as for representing decompositions.

Let the indexed family $< \mathcal{Q}_j >$, $1 \leq j \leq m$, be given, where $\mathcal{Q}_1 = \{1\}$, $\mathcal{Q}_j = \{j, j+1, ..., m-j+1\}$, $2 \leq j \leq m$. Then, we have the representation model:

$$<< \mathcal{Q}_j >, 1 \leq j \leq m;\ W \text{ given in (1.1)};\ \{p\}_{\mathcal{Q}_j} > \tag{2.27}$$

For each choice function $p \in \{p\}_{\mathcal{Q}_j}$ we create the set of indexes $J_p = CD_p$. Then, we have the indexed family $< \mathcal{F}_i >$, $1 \leq i \leq m$, $\mathcal{F}_i = \{1, 2, ..., \min\{i, m\}\}$ and the requirement (2.28).

$$W : h(i) = q \text{ only if exists } r \in J_p \text{ that } r < i \tag{2.28}$$

The indexed family $< \mathcal{F}_k >$, $k \in J_p$ is a subfamily of the indexed family $< \mathcal{F}_i >$, $1 \leq i \leq m$. Each choice function p is an increasing choice function. Since for each $p \in \{p\}_{\mathcal{Q}_j}$ there is $f \in \{f\}_{\mathcal{F}_k}$ that $f = p^{-1}$, so f is an increasing choice function too. That happens because the reversal function to an increasing one is also an increasing function. The codomain of f is the set $\{1, 2, ..., m\}$. Then, the following model is given.

$$\begin{cases} < \mathcal{J} = \{J_p\} \text{ is the collection of codomains of } \{p\}_{\mathcal{Q}_k}, \text{see(2.27)} >, \\ << \mathcal{F}_k >, k \in J_p; W \text{ given in (1.1)}; \{f\}_{\mathcal{F}_k} >, (f = p^{-1}), J_p \in \mathcal{J}, \\ << \mathcal{F}_i >, 1 \leq i \leq n; W \text{ given in (2.28)}; \{h\}_{\mathcal{F}_i} > . \end{cases} \tag{2.29}$$

The Q property holds for the model, moreover, the set of choice functions $\{f\}_{\mathcal{F}_k}$ creates the full decomposition of the set $\{h\}_{\mathcal{F}_i}$. The proof is very similar to the given in the previous section for the decompositions.

Discussing the representation issue, we state that each choice function $p \in \{p\}_{\mathcal{Q}_k}$ represents assignment of each block to its minimum element, so that the increasing enumeration of blocks according to the increasing order of their minimum elements is preserved. Since $f = p^{-1}$ so its representation is enough obvious. Each choice function $h \in \{h\}_{\mathcal{F}_i}$ represents assignment of the elements to the blocks. Satisfiability of the requirement (2.28) makes assertion that assignment of the minimum element to each block is made according to the corresponding choice function p.

Example 2.6.1 *Given is the set* $\{1, 2, 3, 4, 5, 6\}$. *The aim is to construct a choice function h that would represent a partition of the set into* 3 *blocks. The basic representation is in the form of choice function h of the following indexed family:*

$$\mathcal{F}_1 = \{1\},\ \mathcal{F}_2 = \{1, 2\},\ \mathcal{F}_3 = \mathcal{F}_4 = \mathcal{F}_5 = \mathcal{F}_6 = \{1, 2, 3\}$$

We create the following auxiliary family $< \mathcal{Q}_j >$, $1 \leq i \leq 3$, $\mathcal{Q}_1 = \mathcal{Q}_2 = \mathcal{Q}_3 = \{1, 2, 3, 4, 5, 6\}$.

Any increasing choice function p of the family $\mathcal{Q}_j$ represents an assignment of blocks into minimum elements of these blocks by mapping indices of particular blocks into minimum elements of these blocks. For instance, the mapping $p : 1 \longrightarrow 1,\ 2 \longrightarrow 4,\ 3 \longrightarrow 5$ corresponds to $\min(B_1) = 1$, $\min(B_2) = 4$, $\min(B_3) = 5$. The inverse function $f = p^{-1}$:

$1 \longrightarrow 1, \quad 4 \longrightarrow 2, \quad 5 \longrightarrow 3$. In order to get any choice function h representing a partition of the set $\{1, 2, 3, 4, 5, 6\}$ into 3 blocks, we have to make an extension of function f to the whole functions $h \in \{h\}_{\mathcal{F}_i}$. Therefore, we have the following choice functions that are extensions of f.:

$h_1: \quad 1 \to 1, \quad 2 \to 1, \quad 3 \to 1, \quad 4 \to 2, \quad 5 \to 3, \quad 6 \to 1$
$h_2: \quad 1 \to 1, \quad 2 \to 1, \quad 3 \to 1, \quad 4 \to 2, \quad 5 \to 3, \quad 6 \to 2$
$h_3: \quad 1 \to 1, \quad 2 \to 1, \quad 3 \to 1, \quad 4 \to 2, \quad 5 \to 3, \quad 6 \to 3$

The following partitions are represented by h_1, h_2, h_3.

$$\Pi_1 =< B_1 = \{1,2,3,6\}, \quad B_2 = \{4\}, \quad B_3 = \{5\} >,$$
$$\Pi_2 =< B_1 = \{1,2,3\}, \quad B_2 = \{4,6\}, \quad B_3 = \{5\} >,$$
$$\Pi_3 =< B_1 = \{1,2,3\}, \quad B_2 = \{4\}, \quad B_3 = \{5,6\} >$$

□

For modelling subsets of the partitions by proper subsets of the choice functions, we can restrict the set $\{p\}_{\mathcal{Q}_j}$ creating a set $\{p'\}_{\mathcal{Q}'_j}$ and/or restrict the set $\{h\}_{\mathcal{F}_i}$ creating $\{h'\}_{\mathcal{F}'_i}$. In order to restrict the set $\{p\}_{\mathcal{Q}_j}$ the general methodology for restricting sets of increasing choice functions can be applied. In order to restrict the set $\{h\}_{\mathcal{F}_i}$ to the set $\{h'\}_{\mathcal{F}'_i}$, we can restrict the indexed sets $\mathcal{F}_i$ by forming $f'_i \subset \mathcal{F}_i$ or we can impose an additional requirement W_1. If each restricted set $\mathcal{F}'_i$ contains the minimum element of $\mathcal{F}_i$, then the Q property holds. Any requirement W_1 which enables us to assign the minimum element of $\mathcal{F}_i$ as the value of $h(i)$ can also be used for the models. Then, the Q property holds.

Example 2.6.2 *In order to represent the set of partitions of* $\{1 \div 6\}$ *into exactly* 3 *blocks, so that* 4 *is not the minimum element of any block and elements* 2 *and* 3 *do not belong to the same block, we use the following model.*

$\mathcal{Q}_1 = \{1\}$, $\mathcal{Q}_2 = \{2, 3, 5\}$, $\mathcal{Q}_3 = \{3, 5, 6\}$
$\mathcal{F}_1 = \{1\}$, $\mathcal{F}_2 = \{1, 2\}$, $\mathcal{F}_3 = \mathcal{F}_4 = \mathcal{F}_5 = \mathcal{F}_6 = \{1, 2, 3\}$

The requirement $W_1 : h(3) \neq h(2)$ *is imposed on the set of choice functions* $\{h\}_{\mathcal{F}_i}$.

□

2.7 Decompositions with fixed cardinal of each block

By a decomposition with fixed cardinal of each block, we mean the decomposition of the set A into exactly m blocks whose cardinals are given. Suppose, we have fixed the cardinals $k_1 =\mid B_1 \mid, k_2 =\mid B_2 \mid, ..., k_m =\mid B_m \mid$

of the blocks B_1, B_2, ..., B_m creating a requested decomposition of the set $\{1, 2, ..., n\}$ into exactly m blocks. Then, a decomposition is an assignment of elements of the set $\{1, 2, ..., n\}$ into the blocks. In order to make assertion that each assignment represents decomposition indeed and no two assignments represent the same decomposition, we make the following model.

With denotation s_t^z we mean a "seat" inside a block, where z corresponds to the block number and t corresponds to the seat number inside the $z-$th block. Then, the codomain of the requested assignment can be seen as the ordered $n-$tuple $< s_1^1, s_2^1, ..., s_{k_1}^1, s_1^2, s_2^2, ..., s_{k_2}^2, ..., s_1^m, s_2^m, ..., s_{k_m}^m >$, where $s_1^1, s_2^1, ..., s_{k_1}^1$ represents the "seats" of the first block and so on.

We develop now the choice function model for representing the full set of the decompositions.

Let $< \mathcal{D}_i >$, $1 \leq i \leq n$ be the indexed family, where $\mathcal{D}_i = \{1, 2, ..., n\} = \{s_1^1, s_2^1, ..., s_{k_1}^1, s_1^2, s_2^2, ..., s_{k_2}^2, ..., s_1^m, s_2^m, ..., s_{k_m}^m\}$. We have the following requirements W.

$$h : i \longrightarrow s_t^z \text{ only if there exists } i_1 < i \text{ such that } i_1 \longrightarrow s_{t-1}^z; 2 \leq t \leq k_z \tag{2.30}$$

$$h : i \longrightarrow s_t^z \text{ only if there is } i_1 < i \text{ that } h(i_1) = s_t^{z-1} \tag{2.31}$$

$$h : i \longrightarrow s_t^z \text{ only if for every } i_1 < i \text{ there is } h(i_1) \neq s_t^z \tag{2.32}$$

Then, we have the following representation model;

$$<< \mathcal{D}_i >, 1 \leq i \leq n;\ W \text{ equals} (2.30) \wedge (2.31);\ \{h\}_{\mathcal{D}_i} > . \tag{2.33}$$

Proposition 2.7.1 *Each $h \in \{h\}_{\mathcal{D}_i}$ corresponds uniquely to one decomposition of the set A into m blocks with fixed cardinal of each block.*

Proof. A decomposition must be represented by a bijection that maps the set $\{1, 2, .., n\}$ into the set $\{s_1^1, s_2^1, ..., s_{k_1}^1, s_1^2, s_2^2, ..., s_{k_2}^2, ..., s_1^m, s_2^m, ..., s_{k_m}^m\}$. In order to make assertion that no two bijections represent the same decomposition, we request that each partial assignment restricted to the elements $s_1^i, s_2^i, ..., s_{k_i}^i$, i.e., to all the "seats" of the same block is an increasing mapping. Since it is to be satisfied for every $i : 1 \leq i \leq m$, so the requested assignment is to be a bijection piecewise increasing.

The requirement (2.31) satisfied makes assertion that each $h \in \{h\}_{\mathcal{D}_i}$ is a one−to−one mapping.

The requirement (2.30) satisfied makes assertion that for each $h \in \{h\}_{\mathcal{D}_i}$, we have $h(i) = s_t^z$ and if $h(j) = s_r^z$, then $j<i$, $r < t$. Hence, the choice function h assigns increasingly the elements i to the numbers of seats of a block. Therefore, no two choice functions h can represent the same decomposition.

Hence, each choice function $h \in \{h\}_{\mathcal{D}_i}$ represents uniquely a decomposition of the set A into m blocks with fixed cardinal of each block.

□

The model (2.33) is not reduced, since the index $i = 1$ can only be assigned to the first seat of any block, the index $i = 2$ can only be assigned to the first or the second seat of any block and so on. Moreover, the index $i = n$ can only be assigned to the last seats of each block, the index $i = n-1$ can only be assigned to the last or before last seat of each block and so on. In general the $i-$th index can be assigned to the seat s_q^j that its order q in the $j-$th block is not smaller than a_j and is not greater than r_j, where $r_j = \min\{i, k_j\}$, $a_j = \max\{s_1^j, (k_j - (n-i))\}$.

The Q property holds for the model (2.33) since satisfiability of the requirement W that is conjunction of (2.30) and (2.31) makes assertion that only the seats s_q^j satisfying the above restriction could be assigned to the indexes i. Moreover, for every i we can assign an item $s_t^z \in \mathcal{D}_i$.

For creation of the reduced model we use the mentioned restriction for modelling the indexed sets of the family $< \mathcal{K}_i >$, $1 \leq i \leq n$. We have, $\mathcal{K}_i = \{s_{a_1}^1 \div s_{r_1}^1, s_{a_2}^2 \div s_{r_2}^2, ..., s_{a_m}^m \div s_{r_m}^m\}$. Then, the reduced model is as follows:

$$<< \mathcal{K}_i >, 1 \leq i \leq n;\ W \text{ equals } (2.30) \wedge (2.31);\ \{h\}_{\mathcal{K}_i} >, \tag{2.34}$$

Example 2.7.1 *Let* $A = \{1 \div 7\}$, $m = 3$ *and* $|B_1| = 3$, $|B_2| = 3$, $|B_3| = 1$. *We specify the model for representing the decompositions.*

Each choice function $h \in \{h\}_{\mathcal{K}_i}$ of the model 2.34 represents a decomposition of the set A. We have here $< \mathcal{K}_i >$, $1 \leq i \leq 7$, $\mathcal{K}_i = \{s_1^1, s_2^1, s_3^1, s_1^2, s_2^2, s_3^2, s_1^3\}$. For instance $h : 1 \longrightarrow s_1^2$, $2 \longrightarrow s_1^3$, $3 \longrightarrow s_1^1$, $4 \longrightarrow s_2^2$, $5 \longrightarrow s_3^2$, $6 \longrightarrow s_2^1$, $7 \longrightarrow s_3^1$ represents decomposition that $B_1 = \{3, 6, 7\}$, $B_2 = \{1, 4, 5\}$, $B_3 = \{2\}$.

□

Consider now the restricted models. Since each $h \in \{h\}_{\mathcal{K}_i}$ is a bijection piecewise increasing, therefore the restrictions of the indexed sets and extensions of the requirements W_1 must follow the general rules concerning modelling subsets of bijections.

We are concerned now with modelling subsets $\{h\}_{\mathcal{K}_i'}$ of the set $\{h\}_{\mathcal{K}_i}$. The elementary restriction is of the form:

$$s_t^z \notin \mathcal{K}_i',\ i \in I,\ I \subseteq \{1, 2, ..., n\}, z \in \{1, 2, ..., m\}, 1 \leq t \leq k_z \tag{2.35}$$

The elementary restriction (2.35) means that the $i-$th element can not be assigned to any s_t^z, $1 \leq t \leq k_z$. The corresponding model represents the

set of decompositions $\{h\}_{\mathcal{K}_i'}$ that the $i-$th element does not belong to the $z-$th block. If $k_z < n-i+1$, then the model S_U containing a restricted indexed family $< \mathcal{K}_i >$, $1 \leq i \leq n$, whose an indexed set $\mathcal{K}_i'$ is modeled according to (2.35) satisfies the Q property. One can prove it observing that there are still $(n-i+1)-k_z$ elements belonging to the indexed set $\mathcal{K}_i$ that can be assigned to i. If the restricted indexed family $< \mathcal{K}_i' >$, $1 \leq i \leq n$, contains more than one restricted set $\mathcal{K}_i'$, then the general requirements for restricting indexed family for the models representing choice bijections must hold.

We observe that the model (2.34) is very inconvenient for handling. Therefore, we will show now another modelling. Let the set of seats $\{s_1^1, s_2^1, ..., s_{k_1}^1, s_1^2, s_2^2, ..., s_{k_2}^2, ..., s_1^m, s_2^m, ..., s_{k_m}^m\}$ be well ordered and being identified with the set of indexes, so $I =< s_1^1, s_2^1, ..., s_{k_1}^1, s_1^2, s_2^2, ..., s_{k_2}^2, ..., s_1^m, s_2^m, ..., s_{k_m}^m >$, the index $i = k_1 + k_2 + ... + k_{z-1} + t$ corresponds to the seat s_i^z. Then, we split the set of indexes I into subsets $I_1, I_2, ..., I_m$ that $I =< I_1 + I_2 + ... + I_m >$, the operator $+$ means here concatenation. Then, each indexed set $\mathcal{S}_i \subseteq \{1, 2, ..., n\}$ represents items of the given set A. The exact specification of the indexed sets $\mathcal{S}_i$ is as follows:

$\mathcal{S}_j = \{q, q+1, ..., n-k_z+1\}$, $q = j - k_1 + k_2 + ... + k_{z-1}$, $k_1 + k_2 + ... + k_{z-1} \leq j \leq k_1 + k_2 + ... + k_z$, i.e., $j \in I_z$. Then, we have the following specification of the requirements:

$$h(j) > h(j-1), \text{ where } j \in I_z \text{ and } (j-1) \in I_z \tag{2.36}$$

$$h(j) \neq h(p), \text{where } j \in I_z, p \in I_{z-1} \text{ or } p \in I_{z-2} ... \text{or } p \in I_1 \tag{2.37}$$

The representation model is

$$<< \mathcal{S}_i >, \ 1 \leq i \leq n; \ W \ \text{ equals (2.36)} \wedge \text{(2.37)}; \ \{h\}_{\mathcal{S}_i} > \tag{2.38}$$

One can easily check that $\{h\}_{\mathcal{S}_i}$ represents the set of all the decompositions of the set A with fixed cardinal of each block. Nevertheless, the requirement (2.37) is inconvenient for testing. Therefore, we will transform now the model (2.38) into the more convenient one for handling.

Let CD^z denote the codomain of the partial mapping $h(1), h(2), ..., h(q), ..., h(k_z)$, where $q = j - (k_1 + k_2 + ... + k_{z-1})$, $j \in I_z$, $1 \leq z \leq m$. Then, we investigate the indexed family $< \mathcal{T}_i >$, $1 \leq i \leq n$, where $\mathcal{T}_j \subset \mathcal{U}_z = \{1, 2, ..., n\} - CD^1 - CD^2 - ... - CD^{z-1}$, $(k_1 + k_2 + ... + k_{z-1}) \leq j \leq (k_1 + k_2 + ... + k_z)$. The set $\mathcal{U}_z$ is increasingly ordered and can be specified as $\mathcal{U}_z = \{x_1, x_2, ..., x_p\}$, where $p = n - (k_1 + k_2 + ... + k_{z-1})$. Then, $\mathcal{T}_j = \{x_q, x_{q+1}, ..., x_{p-k_z+q}\}$, since we model increasing choice functions for indexes j belonging to I_z.

Observe, the construction of the indexed family $< \mathcal{T}_i >$, $1 \leq i \leq n$, makes assertion that the requirement (2.37) is automatically satisfied. Therefore, we have at last the model:

$$<< \mathcal{T}_i >,\ 1 \leq i \leq n;\ W \text{ given in (2.36)};\ \{h\}_{\mathcal{T}_i} > \qquad (2.39)$$

The model (2.39) is reduced and the Q property holds.

Example 2.7.2 *Let us represent the decomposition shown at the example (2.7.1) using the models (2.38) and (2.39).*

The model (2.38) is specified as follows:
$\mathcal{S}_1 = \{1,2,3,4,5\}$, $\mathcal{S}_2 = \{2,3,4,5,6\}$, $\mathcal{S}_3 = \{3,4,5,6,7\}$, $\mathcal{S}_4 = \{1,2,3,4,\ 5\}$, $\mathcal{S}_5 = \{2,3,4,5,6\}$, $\mathcal{S}_6 = \{3,4,5,6,7\}$, $\mathcal{S}_7 = \{1,2,3,4,5,6,7\}$,
For constructing the model (2.39) we start with the subfamily $< \mathcal{T}_i >$, $1 \leq i \leq 3$.
$\mathcal{T}_1 = \{1,2,3,4,5\}$, $\mathcal{T}_2 = \{2,3,4,5,6\}$, $\mathcal{T}_3 = \{3,4,5,6,7\}$. Then, we select a partial choice mapping say $h : 1 \longrightarrow 3,\ 2 \longrightarrow 6,\ 3 \longrightarrow 7$. So, the codomain $CD^1 = \{3,\ 6,\ 7\}$. Hence, $\mathcal{U}_2 = \{1,\ 2,\ ...,\ 7\} - \{3,\ 6,\ 7\} = \{1,\ 2,\ 4,\ 5\}$.
Then, $\mathcal{T}_4 = \{1,2\}$, $\mathcal{T}_5 = \{2,4\}$, $\mathcal{T}_6 = \{4,5\}$.
We take the choice mapping $h : 4 \longrightarrow 1,\ 5 \longrightarrow 4,\ 6 \longrightarrow 5$. Then, $CD^2 = \{1,\ 4,\ 5\}$. We have $\mathcal{U}_3 = \mathcal{U}_2 - CD^2 = \{2\}$. Hence, $\mathcal{T}_7 = \{2\}$.

□

Modeling subsets of the decompositions with fixed cardinal of each block is much easier by using the model (2.39). For modelling subsets of $\{h\}_{\mathcal{T}_i}$, we have to follow the general theory concerning modelling subsets of increasing choice functions and subsets of choice bijections.

2.8 Partitions with fixed cardinals of each block

If the set of partition is modeled basing on (2.34), then additional requirements (2.40) and (2.41) that represent the difference between decompositions and the corresponding partitions have to be satisfied.

$$\begin{cases} h : i \longrightarrow s_1^z \text{ only if there exists } i_1 < i \\ \text{such that } i_1 \longrightarrow s_1^{z-1}, \text{where } 2 \leq z \leq m \end{cases} \qquad (2.40)$$

$$h : s_1^1 \longrightarrow 1 \qquad (2.41)$$

The requirements 2.40 and 2.41 are hard for testing their satisfiability, that would make the model very inconvenient for handling. Therefore, we use the model (2.39) as the basis and add the requirement (2.42).

Let $K \subset I$ that $K = \{1, k_1+1, k_1+k_2+1, ..., k_1+k_2+...+k_{z-1}+1\}$. For each $k \in K$, we form the following requirement W.

$$\begin{aligned} h(k) &> h(k-1) \\ h(1) &= 1 \end{aligned} \tag{2.42}$$

The model

$$<< \mathcal{T}_i >,\ 1 \leq i \leq n;\ W \text{ equals } (2.36) \wedge (2.42);\ \{h\}_{\mathcal{T}_i} > \tag{2.43}$$

is not reduced since only element 1 from $\mathcal{T}_1$ can be assigned to the index 1. The reduced model we get investigating indexed family $< \mathcal{N}_i >$, $1 \leq i \leq n$ that $\mathcal{N}_1 = \{1\}$ and $\mathcal{N}_k = \min(\mathcal{U}_z)$ for each $k \in K$, $\mathcal{N}_i = \mathcal{T}_i$ for $i \notin K$. Then, we have the following reduced model that the Q property holds.

$$<< \mathcal{N}_i >,\ 1 \leq i \leq n;\ W \text{ given in } (2.36);\ \{h\}_{\mathcal{N}_i} > \tag{2.44}$$

The set $\{h\}_{\mathcal{N}_i}$ of the model (2.44) represents the set of the partitions with fixed cardinal of each block. Observe, the requirement (2.42) does not need to be tested since construction of the subfamily $< \mathcal{N}_k >$, for all $k \in K$ makes assertion that (2.42) is satisfied automatically.

The model (2.44) is very convenient for representing distinct subsets of the set of all the partitions. For modelling such restricted sets of partitions, we use the indexed family $< \mathcal{N}_i >$, $1 \leq i \leq n$, where $\mathcal{N}_i \subseteq \mathcal{K}_i$.

Example 2.8.1 *The goal is to transform the model given in (2.7.1), so that the partitions instead of decompositions would be represented.*

We have the following indexed subfamily $< \mathcal{N}_i >$, $1 \leq i \leq 3$, $\mathcal{N}_1 = \{1\}$, $\mathcal{N}_2 = \{2,3,4,5,6\}$, $\mathcal{N}_3 = \{3,4,5,6,7\}$.

Selecting the partial mapping $h : 1 \longrightarrow 1,\ 2 \longrightarrow 4,\ 3 \longrightarrow 7$, we have $CD^1 = \{1, 4, 7\}$. Hence, $\mathcal{U}_2 = \{2, 3, 5, 6\}$. Then, the indexed subfamily $< \mathcal{N}_i >$, $4 \leq i \leq 6$ is as follows:

$\mathcal{N}_4 = \{2\}$, $\mathcal{N}_5 = \{3,5\}$, $\mathcal{N}_6 = \{5,6\}$,

Selecting the partial mapping $h : 4 \longrightarrow 2,\ 5 \longrightarrow 3,\ 6 \longrightarrow 6$, we have $CD^2 = \{2, 3, 6\}$. Hence, $\mathcal{U}_3 = \{5\}$. Therefore, $\mathcal{N}_7 = \{5\}$. Obviously, $h : 7 \longrightarrow 5$.

Then, the choice function h represents the partition $B_1 = \{1, 4, 7\}$, $B_2 = \{2, 3, 6\}$, $B_3 = \{5\}$.

□

By uniform partition, we mean a partition of the set $\{1, 2, ..., n\}$ into exactly m blocks such that each one has n/m elements. The uniform partition can be treated as the special case of partitions with fixed cardinal of each block.

2.9 Conclusions

In this chapter we have shown that arbitrary sets of combinatorial objects can be represented by the corresponding sets of choice functions. For the representation we have used mainly increasing choice functions, monotonic choice functions and choice bijections. In order to restrict the sets of the represented objects, we used additional requirements W_1 to be satisfied by the corresponding choice functions. The sets of combinatorial objects can also be modeled by deforming the indexed sets of the corresponding indexed families. The restricted models are mostly reduced and the Q property holds. Selection and construction of the most convenient model is not trivial and requires many considerations as it was shown with construction of the most suitable model for representing sets of decompositions with fixed cardinal of each block. The next chapters extend possibilities of modelling a given set of combinatorial objects using different models S_U.

3
Structural numbers and tables D

3.1 The canonical forms

Given is $S_U =<< \mathcal{G}_i^{\max} >, 1 \leq i \leq m, \mathcal{G}_i^{\max} \subseteq \mathcal{U}; W; \{h\}_{\mathcal{G}_i^{\max}} >$ that is reduced and the Q property holds. The indexed family $< \mathcal{G}_i^{\max} >, 1 \leq i \leq m$ is the maximal indexed family for the requirement W, see Definition 1.1.1, on page 17. With denotation $<^{\mathcal{U}}$ we mean a given order on the set $\mathcal{U}$. The indexed sets $\mathcal{G}_i^{\max}$ are ordered with $<^{\mathcal{U}}$, respectively. The order of $q \in \mathcal{G}_i^{\max}$, denoted by $ord(q, \mathcal{G}_i^{\max})$, we call the rank of q in $\mathcal{G}_i^{\max}$ according to $<^{\mathcal{U}}$. The order of q on $\mathcal{U}$, we denote similarly by $ord(q, \mathcal{U})$. Observe, for a given q we have $ord(q, \mathcal{G}_i^{\max}) \neq ord(q, \mathcal{U})$, in general. For instance, if $\mathcal{U} = \{1, 2, ...n\}$, $\mathcal{G}_i^{\max} = \{i, i+1, ..., n-m+i\}$ and if $<^{\mathcal{U}}$ means the increasing ordering $\mathcal{U}$, then $ord(i, \mathcal{G}_i^{\max}) = 1$, while $ord(i, \mathcal{U}) = i$. Hence, $ord(q, \mathcal{G}_i^{\max}) \neq ord(q, \mathcal{U})$ indeed. We treat $ord(q, \mathcal{G}_i^{\max})$ as an operation that assigns x to the pair $(q, \mathcal{G}_i^{\max})$. Then, with $x(\mathcal{G}_i^{\max}) = q$, we denote the inverse operation that assigns $q \in \mathcal{G}_i^{\max}$ that $ord(q, \mathcal{G}_i^{\max}) = x$. Therefore, with denotation $1(\mathcal{G}_i^{\max})$, we mean the minimum element from the set $\mathcal{G}_i^{\max}$, while $2(\mathcal{G}_i^{\max})$ means the next to the minimum element of the set $\mathcal{G}_i^{\max}$ and so on. We emphasize that the operation $x(\mathcal{G}_i^{\max})$ has been investigated only for the maximal indexed families. The aim of the following part of this section is to extend the operation $x(\mathcal{G}_i^{\max})$ for arbitrary indexed families.

Let $< \mathcal{G}_i >, 1 \leq i \leq m$, be an indexed family for the given model S_U.

Definition 3.1.1 *The minimal non–deformed family* $< \mathcal{G}_i^{\min} >, 1 \leq i \leq m$ *for an arbitrary indexed family* $< \mathcal{G}_i >, 1 \leq i \leq m$, *is defined as follows:*
(*i*) $\mathcal{U} \longleftarrow \mathcal{G}_1 \cup \mathcal{G}_2 \cup ... \cup \mathcal{G}_m$.

(ii) $< \mathcal{G}_i^{\min} >$, $1 \leq i \leq m$, *is isomorphic to the maximal indexed family for given* $n = \mid \mathcal{U} \mid$, m *and* W,
(iii) $\mathcal{G}_i \subseteq \mathcal{G}_i^{\min}$.

Example 3.1.1 *Given is the representation system* S_U *that the indexed family and the requirement* W *are specified as follows:* $\mathcal{G}_1 = \{2, 3, 4, 5\}$, $\mathcal{G}_2 = \{3, 5, 6\}$, $\mathcal{G}_3 = \{4, 7\}$ *and* $W{:}h(i) > h(i-1)$. *Specify the corresponding maximal indexed family and the minimal non–deformed indexed family* $< \mathcal{G}_i^{\min} >$, $1 \leq i \leq m$.

We have $\mathcal{U} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 = \{2, 3, 4, 5, 6, 7\}$, so $n = \mid \mathcal{U} \mid = 6$.
Therefore, the maximal indexed family is as follows: $\mathcal{G}_1^{\max} = \{1, 2, 3, 4\}$, $\mathcal{G}_2^{\max} = \{2, 3, 4, 5\}$, $\mathcal{G}_3^{\max} = \{3, 4, 5, 6\}$.
While $\mathcal{G}_1^{\min} = \{2, 3, 4, 5\}$, $\mathcal{G}_2^{\min} = \{3, 4, 5, 6\}$, $\mathcal{G}_3^{\min} = \{4, 5, 6, 7\}$ is the minimal non–deformed indexed family.

□

Observe, the minimal non–deformed family $< \mathcal{G}_i^{\min} >$, $1 \leq i \leq m$, is simultaneously the maximal indexed family if $\max(\mathcal{U}) = \mid \mathcal{U} \mid = n$. Consequently, if for a given system S_U the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$ is the maximal indexed family $< \mathcal{G}_i^{\max} >$, $1 \leq i \leq m$, then it is simultaneously the minimal non–deformed family $< \mathcal{G}_i^{\min} >$, $1 \leq i \leq m$.

Evaluation of the minimal non–deformed indexed family for a given system S_U is performed in three steps:

---

1. $\mathcal{U} \longleftarrow \mathcal{G}_1 \cup \mathcal{G}_2 \cup \ldots \cup \mathcal{G}_m$.
2. For the given $n = \mid \mathcal{U} \mid$, m and W make the maximal indexed family $< \mathcal{G}_i^{\max} >$, $1 \leq i \leq m$.
3. Make the minimal non–deformed indexed family $< \mathcal{G}_i^{\min} >$, $1 \leq i \leq m$ that is isomorphic to $< \mathcal{G}_i^{\max} >$, $1 \leq i \leq m$, and $\mathcal{G}_i^{\min} \subseteq \mathcal{U}$.

Performance of the steps 2. and 3. is simple and can be explained basing on the above given example.

The operation $ord(q, \mathcal{G}_i)$ is specified only if $q \in \mathcal{G}_i$. We can determine whether operation $x(\mathcal{G}_i)$ is valid by testing whether $x(\mathcal{G}_i^{\min})$ belongs to $\mathcal{G}_i$ or not.

Definition 3.1.2 *We define the operations* $ord(q, \mathcal{G}_i)$ *and* $x(\mathcal{G}_i)$ *as follows:* $ord(q, \mathcal{G}_i) = ord(q, \mathcal{G}_i^{\min})$ *and* $x(\mathcal{G}_i) = x(\mathcal{G}_i^{\min})$ *only if* $x(\mathcal{G}_i^{\min}) \in \mathcal{G}_i$.

Example 3.1.2 *Given is the representation system* S_U *that the indexed family* $< \mathcal{G}_i >$, $1 \leq i \leq m$, *is specified as follows:* $\mathcal{G}_1 = \{2, 3, 4, 5\}$, $\mathcal{G}_2 = \{3, 5, 6\}$, $\mathcal{G}_3 = \{4, 7\}$. *Evaluate* $ord(7, \mathcal{G}_3)$ *and* $3(\mathcal{G}_2)$.

The corresponding minimal non–deformed indexed family evaluated in the previous example is as follows: $\mathcal{G}_1^{\min} = \{2, 3, 4, 5\}$, $\mathcal{G}_2^{\min} = \{3, 4, 5,$

6}, $\mathcal{G}_3^{\min} = \{4, 5, 6, 7\}$. Then, we have $7 \in \mathcal{G}_3$. Hence, $ord(7, \mathcal{G}_3) = ord(7, \mathcal{G}_3^{\min}) = 4$. Concerning evaluation $3(\mathcal{G}_2)$ we observe that if the result exists, then it equals $3(\mathcal{G}_2^{\min})$. Since $3(\mathcal{G}_2^{\min}) = 5$ and since $5 \in \mathcal{G}_2$, so $3(\mathcal{G}_2) = 5$.

□

We are concerned now with the basic forms of a choice function.

Definition 3.1.3 *The custom form of a choice function h, we call the string $< h(1), h(2), ..., h(m) >$.*

The custom form has been used till now, we investigate now another valid form of a choice function.

Definition 3.1.4 *By the canonical form of the function $h \in \{h\}_{\mathcal{G}_i}$, we mean the mapping $h : i \longrightarrow x_i$, so that $x_i(\mathcal{G}_i) = q_i$ and $g_i \in \mathcal{G}_i$, $1 \leq x_i \leq \mid \mathcal{G}_i \mid$.*

We identify the canonical form of each choice function h with the following $m-$tuple $< x_1, x_2, ..., x_m >$, so the function h is represented by its ordered codomain.

If all the choice functions $h \in \{h\}_{\mathcal{G}_i}$ are specified by the canonical form, then we say that the set $\{h\}_{\mathcal{G}_i}$ is given in the canonical form. Similarly, with $< x_1, x_2, ..., x_i >$, $1 \leq i \leq m-1$, we denote the canonical form of a partial mapping $h(1), h(2), ..., h(i)$.

We say that the requirement W has the canonical specification if it is of the form $x_i R E\{i, x_1, x_2, ..., x_{i-1}\}$, where $E\{i, x_1, x_2, ..., x_{i-1}\}$ means an expression made of $i, x_1, x_2, ..., x_{i-1}$. Usually, we have the following simple form $E\{x_1, x_2, ..., x_{i-1}\}$ that corresponds to the simplified custom form of W. For instance, the canonical form of the requirement W for the model $<< \mathcal{A}_i >$, $1 \leq i \leq m$; W given in (1.1); $h \in \{h\}_{\mathcal{A}_i} >$ is $x_i \geq x_{i-1}$. The canonical form of the set $\{h\}_{\mathcal{G}_i}$, we can get indirectly by making the transformation of the corresponding custom form of $\{h\}_{\mathcal{G}_i}$ or by the direct generation using the canonical specification of W.

Example 3.1.3 *Let $< 2, 5, 7 >$, $< 2, 3, 4 >$, $< 2, 6, 7 >$ be the custom form of the choice functions h_1, h_2, h_3, respectively for S_U specified in Example 3.1.2. The goal is to transform h_1, h_2 and h_3 into the canonical form.*

The canonical form of the choice function h_1 is as follows: $< 1, 3, 4 >$.
The function h_2 given in the canonical form is as follows: $< 1, 1, 1 >$.
The function h_3 given in the canonical form is as follows: $< 1, 4, 4 >$.

□

If we have the canonical form of a choice function and the corresponding indexed family is known, then we can transform the canonical form into the custom form as follows: $h(1) = x_1(\mathcal{G}_1), h(2) = x_2(\mathcal{G}_2), ..., h(m) = x_m(\mathcal{G}_m)$.

The given relation between minimal non- deformed indexed family and the canonical form of a choice function is subjective for our following considerations.

3.2 The tables D

With denotation $N[x_1(\mathcal{G}_1), x_2(\mathcal{G}_2), ..., x_i(\mathcal{G}_i)]$, we mean the number of extensions of a partial mapping $< h(1) = x_1(\mathcal{G}_1), h(2) = x_2(\mathcal{G}_2), ..., h(i) = x_i(\mathcal{G}_i) >$ to the choice functions $h(1), h(2), ..., h(m)$ that $h \in \{h\}_{\mathcal{G}_i}$.

Definition 3.2.1 *The number* $N[x_1(\mathcal{G}_1), x_2(\mathcal{G}_2), ..., x_i(\mathcal{G}_i)]$ *is called the structural number for* $\{h\}_{\mathcal{G}_i}$.[1].

For a given representation model the set of all the structural numbers represents the structure of the set $\{h\}_{\mathcal{G}_i}$.

Let $< x_1^1(\mathcal{G}_1), x_2^1(\mathcal{G}_2), ..., x_i^1(\mathcal{G}_i) >$ and $< x_1^2(\mathcal{G}_1), x_2^2(\mathcal{G}_2), ..., x_i^2(\mathcal{G}_i) >$ be the canonical forms of two partial mappings $h_1(1), h_1(2), ..., h_1(i)$ and $h_2(1), h_2(2), ..., h_2(i)$ that $h_1(i) = h_2(i)$. That is equivalent to $x_i^1 = x_i^2$. By $N[1(\mathcal{G}_1)]$ we denote the number of all those choice functions for which $h(1) = 1(\mathcal{G}_1)$. Similarly, with denotation $N[x(\mathcal{G}_i)]$ we mean the number of all those choice functions $h \in \{h\}_{\mathcal{G}_i}$ that $h(i) = x(\mathcal{G}_i)$ and $h(1), h(2), ..., h(i-1)$ are any values except they are fixed for all the concerned choice functions.

Let $< x_1^1, x_2^1, ..., x_{i-1}^1, x_i >$ and $< x_1^2, x_2^2, ..., x_{i-1}^2, x_i >$ be two choice partial mappings for a fixed value $x_i(\mathcal{G}_i)$.

Definition 3.2.2 *The structure of the set* $\{h\}_{\mathcal{G}_i}$ *is symmetric if* $N[x_1^1(\mathcal{G}_1), x_2^1(\mathcal{G}_2), ..., x_{i-1}^1(\mathcal{G}_i), x_i(\mathcal{G}_i)] = N[x_1^2(\mathcal{G}_1), x_2^2(\mathcal{G}_2), ..., x_{i-1}^2(\mathcal{G}_{i-1}), x_i(\mathcal{G}_i)]$ *for any possible* $x_1^1, x_2^1, ..., x_i^1, x_1^2, x_2^2, ..., x_i^2$ *and* i.

If the structure is symmetric, then we have $N[x_1^1(\mathcal{G}_1), x_2^1(\mathcal{G}_2), ..., x_{i-1}^1(\mathcal{G}_{i-1}), x_i^1(\mathcal{G}_i)] = N[x_i(\mathcal{G}_i)]$ for every possible i. For the symmetric $\{h\}_{\mathcal{G}_i}$, we have at most $u.m$ distinct structural numbers, where $u = \max_i \mid \mathcal{G}_i \mid$. Then, the structure of the set $\{h\}_{\mathcal{G}_i}$ will be represented by a table $D[u \times m]$ whose

[1]The Stirling's numbers are also covered as a case of the defined here structural numbers, however the given defnition is much more general.

entries D[i, j] are as follows.

$$\begin{array}{c} \mathrm{D}[1,1] = \mathrm{N}[1(\mathcal{G}_1)] \\ \mathrm{D}[2,1] = \mathrm{N}[2(\mathcal{G}_1)] \\ \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots \\ \mathrm{D}[z,1] = \mathrm{N}[1(\mathcal{G}_1)] \\ \mathrm{D}[1,2] = \mathrm{N}[1(\mathcal{G}_2)] \\ \ldots\ldots\ldots\ldots \\ \mathrm{D}[\mathrm{i},\ \mathrm{j}] = \mathrm{N}[\mathrm{i}(\mathcal{G}_\mathrm{j})] \\ \ldots\ldots\ldots\ldots \\ \mathrm{D}[\mathrm{u},\ \mathrm{m}] = \mathrm{N}[\mathrm{u}(\mathcal{G}_\mathrm{m})] \end{array} \tag{3.1}$$

The tables D have very important meaning for representing the structure of the sets $\{h\}_{\mathcal{G}_i}$ and the structure of distinct subsets of $\{h\}_{\mathcal{G}_i}$. The structure of the sets $\{h\}_{\mathcal{G}_i}$ represented by the tables D we use for testing equivalence of the models and we can develop mutual transformations of the models basing on these tables. The tables D are also used as the basic data structures for developing different *UNRANK*, *RANK*, *RANGE* algorithms, see Chapter 6. That makes a foundation for sequential, parallel and distributed generation of the sets of choice functions. Therefore, we are much concerned with the development of methodology for evaluation of tables D for different classes of choice functions. For certain valid classes of choice functions the tables D have interesting properties enabling their easy evaluation and usage.

3.2.1 The basic algorithms for table D evaluation

Given is the unranking model S_U that is reduced and the Q property holds. The set $\{h\}_{\mathcal{G}_i}$ is symmetric, hence the table D represents the structure of $\{h\}_{\mathcal{G}_i}$.

We observe, that a value $h(1) \in \mathcal{G}_1$ is a prefix of any choice function $h \in \{h\}_{\mathcal{G}_i}$. Hence, the set $\{h(1)\}$ of all the possible values $h(1)$ makes a partition on the set $\{h\}_{\mathcal{G}_i}$. Consequently, $\mid \{h\}_{\mathcal{G}_i} \mid = \mathrm{N}[1(\mathcal{G}_1)] + \mathrm{N}[2(\mathcal{G}_1)] + \ldots . + \mathrm{N}[u(\mathcal{G}_1)]$, where $u(.\mathcal{G}_1) = \max(\mathcal{G}_1)$.

Since, D[x, 1] = N[$x(\mathcal{G}_{i_1})$], so

$$\sum_{x=1}^{u} \mathrm{D}[x,1] = \mid \{h\}_{\mathcal{G}_i} \mid, \tag{3.2}$$

where $\mid \{h\}_{\mathcal{G}_i} \mid$ is the cardinal of the set $\{h\}_{\mathcal{G}_i}$, while $u \times m$ is the size of the table D.

Definition 3.2.3 *With denotation $\mathcal{G}_i^*$ we mean the subset of $\mathcal{G}_i$ for given values $< h(1), h(2), \ldots, h(i-1) >$. The set $\mathcal{G}_i^*$ contains all the elements*

that could be assigned to i, in order to get enabled partial mapping $< h(1), h(2), ..., h(i) >$.

Suppose, we have given now a partial mapping $< h(1), h(2), ..., h(i) >$.

Definition 3.2.4 *With denotation* $\overline{\mathcal{G}}_i^*$ *we mean the subset of* $\mathcal{G}_i^*$ *that is obtained for given partial mapping* $< h(1), h(2), ..., h(i) >$ *and it contains all the elements of* $\mathcal{G}_i^*$ *that are not smaller than* $h(i)$.

We can evaluate the sets $\mathcal{G}_i^*$ and $\overline{\mathcal{G}}_i^*$ by means of values x specified as follows: $x = ord(q, \mathcal{G}_i)$. Then, $\mathcal{G}_i^* = \{q_i : q_i = x(\mathcal{G}_i)$ can equal $h(i)$ for making the possible extension of fixed $< h(1), h(2), ..., h(i-1) >$ to $< h(1), h(2), ..., h(i-1), h(i) >\}$, while $\overline{\mathcal{G}}_i^* = \{q_i : q_i = x(\mathcal{G}_i)$ that $< h(1), h(2), ..., h(i-1), q_i >$ is enabled partial mapping, where $q_i > h(i)$ for the fixed partial mapping $< h(1), h(2), ..., h(i-1), h(i) >\}$. The sets $\mathcal{G}_i^*$ and $\overline{\mathcal{G}}_i^*$ are very convenient notions for explanation and description of numerous algorithms concerning the theory considered here, nevertheless, full presentation of the rutines for evaluation $\mathcal{G}_i^*$ and $\overline{\mathcal{G}}_i^*$ now would be a little bit cumbersome, so, we postpone it. Formal algorithms for evaluation $\mathcal{G}_i^*$ and $\overline{\mathcal{G}}_i^*$ are given in Chapter 6.

The partial mapping $< h(1), h(2), ..., h(i) >$ is a prefix of the partial mapping $< h(1), h(2), ..., h(i), h(i+1) >$ and it is a prefix of a number of choice functions $h \in \{h\}_{\mathcal{G}_i}$. Since the system S_U is symmetric, so the number of extensions $N[h(i)]$ of each partial mapping $< h(i), h(i), ..., h(i) >$, to the full choice functions equals $\sum_{q_{i+1} \in \mathcal{G}_{i+1}^*} N[q_{i+1}]$, where q_{i+1} denotes those values of $h(i+1)$ that are enabled for the fixed $h(i)$. Then, using the canonical form of the choice functions, we have $\sum_{q_i \in \mathcal{G}_i^*} N[q_i] = \sum_{\{x\}} N[x(\mathcal{G}_i)]$, where $x = ord(q_i, \mathcal{G}_i^*)$ and $\{x\}$ denotes the set of all the possible values x that $x(\mathcal{G}_i) \in \mathcal{G}_i^*$. We have $D[x, i] = N[h(i)]$ and $N[h(i)] = \sum_{q_{i+1} \in \mathcal{G}_{i+1}^*} N[q_{i+1}]$. Using the canonical form, we can rewrite the above result as follows: $N[h(i)] = \sum_{\{y\}} N[y(\mathcal{G}_{i+1})]$, where $y = ord(q_{i+1}, \mathcal{G}_{i+1}^*)$ and $\{y\}$ denotes the set of all the possible values y that $y(\mathcal{G}_{i+1}) \in \mathcal{G}_{i+1}^*$.

Then,

$$D[x, i] = \sum_{\{y\}} D[y, i+1], i < n, x \leq u, y \leq u \tag{3.3}$$

Hence, each entry $D[x, i]$ of a table D can be evaluated by adding a subset of entries of the next column of the table D, where $i < n$. Moreover, we have $D[x, m] = 1$ if there is a choice function $h \in \{h\}_{\mathcal{G}_i}$ that $h(m) = x(\mathcal{G}_m)$ or $D[x, m] = 0$ if such function does not exist.

The formula (3.3) we call the sequential pattern for the entries of the table D since we can evaluate the entries of the $i-th$ column after making evaluation of the entries of the $i+1-th$ column.

Basing on the above formula we present the general algorithm for evaluation of the table D for the given system S_U.

Algorithm SEQTABD (general algorithm for SEQuential evaluation of the TABle D)

Input: The model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; \{h\}_{\mathcal{G}_i} >$

Output: the table D

Method: We make the minimum non-deformed indexed family $< \mathcal{G}_i^{\min} >$, $1 \leq i \leq m$ for the given $< \mathcal{G}_i >$, $1 \leq i \leq m$. Then, we evaluate $u = \max_{i,x}\{ x : x = ord(q, \mathcal{G}_i^{\min}), q \in \mathcal{G}_i^{\min}\}$. Determining whether D[$x$, i] = 1 or D[x, i] = 0 depends on existence of $h \in \{h\}_{\mathcal{G}_i}$ that $h(i) = x$. In particularly, D(x, m) = 1 if exists $h \in \{h\}_{\mathcal{G}_i}$ that $h(m) = x$. Evaluation of all the other entries D[x, i], $1 \leq x \leq u$, $1 \leq i \leq m$ is performed in the step 3. basing on the observation given in 3.3.

1. **for** $i \longleftarrow 1$ **to** m **do**
1.1. **if** $u(\mathcal{G}_i^{\min}) \in \mathcal{G}_i$ **then** D[u, i] $\longleftarrow$ 1 **else** D[u, i] $\longleftarrow$ 0
2. **for** $x \longleftarrow 1$ **to** u **do**
2.2. **if** $x(\mathcal{G}_m^{\min}) \in \mathcal{G}_m$ **then** D[x, m] $\longleftarrow$ 1 **else** D[x, m] $\longleftarrow$ 0
3. **for** $x \longleftarrow u - 1$ **downto** 1 **do**
3.1. **for** $i \longleftarrow m - 1$ **downto** 1 **do**
3.1.1. D[x, i] $\longleftarrow$ 0
3.1.2. **if** $x(\mathcal{G}_i^{\min}) \in \mathcal{G}_i^*$ **then**
3.2.1.1. **for** $y \longleftarrow 1$ **to** u **do**
3.1.2.1.1. **if** $y(\mathcal{G}_{i+1}^{\min}) \in \mathcal{G}_{i+1}^*$ **then** D[x, i] $\longleftarrow$D[x, i]+D[y, $i+1$]

The correctness of the presented algorithm is justified by the formula (3.3). Asymptotic complexity of the algorithm is $O(u^2.m)$ under the assumption that the minimal non−deformed indexed family $< \mathcal{G}_i^{\min} >$, $1 \leq i \leq m$ could be found in time $O(u^2.m)$.

If the given indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$ is simultaneously the minimal non−deformed indexed family $< \mathcal{G}_i^{\min} >$, $1 \leq i \leq m$, then the entries of the corresponding table D can be evaluated independently one from each other using general formulas of the form given in (3.4),

$$\mathrm{D}[x, p] = \delta(n, m, p, x), \tag{3.4}$$

where $\delta(n, m, p, x)$ is a function that involves factorials or symbols $\binom{i}{k}$ and so on. Basing on the form (3.4), we have developed parallel algorithm for evaluation the entries of the tables D.

Algorithm PARTABD (general algorithm for PARallel evaluation of the TABle D)

Input: The model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; \{h\}_{\mathcal{G}_i} >$ The number of the processors used equals $u.m$. Each processor is dedicated to an entry of the table D. Hence the processors P are enumerated with (p, i).

Output: the table D.

Method: Depending on the given model S_U we have a proper case of the general formula (3.4).

1. **for** $p \longleftarrow u$ **downto** 1 **do in parallel**

1.1. **for** $i \longleftarrow m$ **downto** 1 **do in parallel**

1.1.1. evaluate D[p, i] using a proper formula (3.4).

Time complexity of the above algorithm equals the complexity of the formula (3.4) that is much lower than time complexity of the algorithm SEQTABD if the number of processors used equals $u.\ m$. Nevertheless, if the algorithm PARTABD where implemented using one processor, then its time complexity would be greater than in the case of the algorithm SEQTABD since for evaluation entries D[p, i] the algorithm SEQTABD requires at least one adding using earlier evaluated entries, while the formula (3.4) involves much more complicated operations.

The algorithms SEQTABD and PARTABD have very general meaning since they can be adopted for evaluation of the entries of the tables D for any given model S_U. For special classes of the models used these algorithms are useless since they are too general and do not gain from specific properties of those models. On one side those algorithms are not very convenient on the other side they are too much complex. The next part of this chapter is devoted to methods of evaluation of the tables D for specific models. We study also equality of tables D for different models in order to extend applications of the specific algorithms and to extend the methods of modelling specific sets of the combinatorial objects as well.

3.3 Congruence and isomorphism

Given are the unranking models $S_U^1 =<< \mathcal{G}_i >, i \in I; W; \{h\}_{\mathcal{G}_i} >$ and $S_U^2 =< \{\mathcal{G}^a{}_j\}; W^a; \{h^a\}_{\mathcal{G}_j^a} >$, where I, J are the ordered sets of indexes, $\mathcal{G}_i \subset \mathcal{U}$, $\mathcal{G}^a{}_j \subset Y$. The models are reduced. We have $| I |=| J |$, while the cardinals $| \mathcal{U} |$ and $| Y |$ can differ.

Let $h \in \{h\}_{\mathcal{G}_i}$ and $h^a \in \{h^a\}_{\mathcal{G}_j^a}$, while $< x_1, x_2, ..., x_i >$ and $< x_1^a, x_2^a, ..., x_i^a >$ being the canonical forms of h and h^a, respectively.

Definition 3.3.1 *We say that the canonical forms of choice functions* $< x_1, x_2, ..., x_i >, i \in I$ *and* $< x_1^a, x_2^a, ..., x_j^a >, j \in J$ *are congruent if* $x_1 = x_1^a, x_2 = x_2^a, ..., x_i = x_j^a$.

If $| \{h\}_{\mathcal{G}_i} |=| \{h^a\}_{\mathcal{G}_j^a} |$ and if for each $h \in \{h\}_{\mathcal{G}_i}$ there is corresponding $h^a \in \{h^a\}_{\mathcal{G}_j^a}$, then the canonical forms of the sets of choice functions $\{h\}_{\mathcal{G}_i}$ and $\{h^a\}_{\mathcal{G}_j^a}$ equal. Similarly, we say that the models S_U^1 and S_U^2 are congruent if the canonical forms of $\{h\}_{\mathcal{G}_i}$ and $\{h^a\}_{\mathcal{G}_j^a}$ equal.

We have immediately the following observation.

- If the sets $\{h\}_{\mathcal{G}_i}$ and $\{h'\}_{\mathcal{G}'_j}$ are symmetric, and if the models S^1_U and S^2_U are congruent, then the same table D represents the structure of $\{h\}_{\mathcal{G}_i}$ and $\{h'\}_{\mathcal{G}'_j}$.

Let φ be a one–to–one mapping that $\varphi : I \longrightarrow J$ and $i_1 >^I i_2$ implies $\varphi(i_1) >^J \varphi(i_2)$, while ψ being a one–to–one mapping that $\psi : \mathcal{U} \longrightarrow Y$ and $u_1 >^{\mathcal{U}} u_2$ implies $\psi(u_1) >^Y \psi(u_2)$ for all the pairs (i_1, i_2) and (u_1, u_2), i.e., the mappings φ and ψ are in accordance with the given orders $<^I$, $<^J$, $<^{\mathcal{U}}$, $<^Y$.

Definition 3.3.2 *We say that the models S^1_U and S^2_U are isomorphic if for every $h \in \{h\}_{\mathcal{G}_i}$ exists $h^a \in \{h^a\}_{\mathcal{G}^a_j}$, that $\psi(h(i)) = h^a(\varphi(i))$ for every $i \in I$ and if $|\ \{h\}_{\mathcal{G}_i}\ | = |\ \{h^a\}_{\mathcal{G}^a_j}\ |$.*

Theorem 3.3.1 *If the models S^1_U and S^2_U are isomorphic, then they are congruent.*

Proof. (i) If S^1_U and S^2_U are isomorphic, then the mapping φ and the order $<^I$ are in accordance with $<^J$.

(ii) If S^1_U and S^2_U are isomorphic, then the mapping ψ and the order $<^{\mathcal{U}}$ are in accordance with the order $<^Y$.

The statement (i) implies: if $h^a \in \{h^a\}_{\mathcal{G}^a_j}$ is an image of $h \in \{h\}_{\mathcal{G}_i}$ that $\psi(h(i)) = h^a(\varphi(i))$ for every $i \in I$ and if $h(i) = x(\mathcal{G}_i)$, then $\psi(h(i)) = x(\mathcal{G}^a_{\varphi(i)})$. The statement (ii) implies that the rank of i according to $<^I$ equals the rank of $\varphi(i)$ according to $<^J$. Hence, the canonical forms of h and h^a are the same.

Since $|\ \{h\}_{\mathcal{G}_i}\ | = |\ \{h^a\}_{\mathcal{G}^a_j}\ |$ holds, so we have assertion that for each $h \in \{h\}_{\mathcal{G}_i}$ there is $h^a \in \{h^a\}_{\mathcal{G}^a_j}$ that their canonical forms are congruent. Therefore, the canonical forms of $\{h\}_{\mathcal{G}_i}$ and $\{h^a\}_{\mathcal{G}^a_j}$ are the same. So, the models S^1_U and S^2_U are congruent. That finishes the proof.

□

Corollary 3.3.1 *If the models S^1_U and S^2_U are isomorphic and if the set $\{h\}_{\mathcal{G}_i}$ is symmetric, then there is a table D that represents the structure of $\{h\}_{\mathcal{G}_i}$ and the structure of $\{h^a\}_{\mathcal{G}^a_j}$, simultaneously.*

Proof. Symmetry of $\{h\}_{\mathcal{G}_i}$ makes assertion that the table D exists for the model S^1_U. Isomorphism of S^1_U and S^2_U makes assertion that if $\{h\}_{\mathcal{G}_i}$ is symmetric, then $\{h^a\}_{\mathcal{G}^a_j}$ is also symmetric and the table D represents the structure of S^2_U.

□

We emphasize that congruence and isomorphism defined are different notions. Isomorphism makes congruence, while congruence does not make isomorphism.

Example 3.3.1 *Given are the following models* $S_U^1 =<< \mathcal{G}_i >, 1 \leq i \leq 4$, $\mathcal{G}_i = \{i, i+1\}$; $W : h(i) > h(j)$; $\{h\}_{\mathcal{G}_i} >$ *and* $S_U^2 =<< \mathcal{G}^a{}_j >, 1 \leq j \leq 4$, $\mathcal{G}^a{}_i = \{1, 2\}$; $W^a : h(j) \geq h(j-1)$; $\{h^a\}_{\mathcal{G}_j^a} >$.

The models are congruent and non−isomorphic.

We show it as follows:

We have $\mathcal{U} = \{1, 2, 3, 4, 5\}$ and $Y = \{1, 2\}$ and the requirements W and W^a differ. Therefore, S_U^1 and S_U^2 can not be isomorphic.

The custom forms of the choice functions $\{h\}_{\mathcal{G}_i}$ and $\{h^a\}_{\mathcal{G}_j^a}$ are as follows:

$\{h\}_{\mathcal{G}_i} = \{< 1, 2, 3, 4 >, < 1, 2, 3, 5 >, < 1, 2, 4, 5 >, < 1, 3, 4, 5 >, < 2, 3, 4, 5 >\}$;

$\{h^a\}_{\mathcal{G}_j^a} = \{< 1, 1, 1, 1 >, < 1, 1, 1, 2 >, < 1, 1, 2, 2 >, < 1, 2, 2, 2 >, < 2, 2, 2, 2 >\}$.

Then, we have the following canonical form of $\{h\}_{\mathcal{G}_i}$: $\{< 1, 1, 1, 1 >, < 1, 1, 1, 2 >, < 1, 1, 2, 2 >, < 1, 2, 2, 2 >, < 2, 2, 2, 2 >\}$. The canonical form of $\{h^a\}_{\mathcal{G}_i^a}$. $= \{< 1, 1, 1, 1 >, < 1, 1, 1, 2 >, < 1, 1, 2, 2 >, < 1, 2, 2, 2 >, < 2, 2, 2, 2 >\}$. Since, the canonical forms equal, so the models S_U^1 and S_U^2 are congruent. Moreover, the sets of the choice functions are symmetric, so we have one common table D for the models S_U^1 and S_U^2 as follows:

$$\left| \begin{array}{cccc} 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{array} \right|.$$

□

Let three representation models S_U^1, S_U^2 and S_U^3 be given.

Proposition 3.3.1 *If* S_U^1 *and* S_U^2 *are congruent and if* S_U^2 *and* S_U^3 *are isomorphic, then* S_U^1 *and* S_U^3 *are congruent.*

Proof. Since S_U^2 and S_U^3 are isomorphic, so S_U^2 and S_U^3 are congruent. Hence, S_U^1 and S_U^3 are congruent.

□

3.4 The regularity of the tables D

Given are the representation models $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m$, $\mathcal{G}_i = \{1, 2, ..., n\}$; W; $\{h\}_{\mathcal{G}_i} >$ and $S^a{}_U =< < \mathcal{G}_i^a >, 1 \leq i \leq m+s$, $\mathcal{G}_i^a = \{1, 2, ..., n+q\}$; W; $\{h^a\}_{\mathcal{G}_i^a} >$. The indexed families $< \mathcal{G}_i >, 1 \leq i \leq m$, and $< \mathcal{G}_i^a >, 1 \leq i \leq m+s$, are the maximal indexed families for the common requirement W. Suppose, the table $D[n \times m]$ represents the structure of the set of choice functions $\{h\}_{\mathcal{G}_i}$. Then, we create the corresponding table $D^a[(n+q) \times (m+s)]$.

Definition 3.4.1 *We say that the table D is regular if* $D^a[j+q, i+s] = D[j, i]$ *for any integer s and q.*

Briefly speaking, if D is regular then it is the subtable of D^a. It was shown that for the set of increasing choice functions the table D = DCM and the table DCM is regular [19]. The regularity of the table D is a valid property of the models since it enables us to construct the table D regardless on parameters n and m. The regular tables D for big enough n and m can be stored and their proper subtables can be used for a particular model.

The regularity of the table D is not a trivial property and it does not hold for all types of choice functions possessing symmetric structure.

Example 3.4.1 *Let the unranking representation model* $<< \mathcal{P}_i >, 1 \leq i \leq m, \mathcal{P}_i = \{1, 2, ..., n\}; \{h\}_{\mathcal{P}_i} >$ *be given, while the requirement W being void. One can easily observe that the set of choice functions* $\{h\}_{\mathcal{P}_i}$ *possess the symmetric structure since* $\mathrm{N}[x(\mathcal{G}_i)] = \mathrm{N}[y(\mathcal{G}_j)]$ *for any pair of indexes* i, j *that* $1 \leq i \leq m$, $1 \leq j \leq m$, *and for any pair of values* x, y, $1 \leq x \leq n$, $1 \leq y \leq n$. *Hence, there is a table D for the model. Nevertheless, the table D is not regular. One can note it observing the following two instances of the general model.*

For $n = m = 3$ we have the following table D:

$$\mathrm{D} = \begin{vmatrix} 6 & 3 & 1 \\ 6 & 3 & 1 \\ 6 & 3 & 1 \end{vmatrix}$$

For $n = m = 4$ we have the following table D:

$$\mathrm{D} = \begin{vmatrix} 64 & 16 & 4 & 1 \\ 64 & 16 & 4 & 1 \\ 64 & 16 & 4 & 1 \\ 64 & 16 & 4 & 1 \end{vmatrix}$$

The table D for $n = m = 3$ is not a subtable of the table D for $n = m = 4$. Hence, the table D that represents the structure of $\{h\}_{\mathcal{P}_i}$ is not regular.

□

3.5 The tables DCM for increasing functions

The table DCM [19] is the table D for the model $S_U = << \mathcal{A}_i >, 1 \leq i \leq m; W$ given in (1.1); $\{h\}_{\mathcal{A}_i} >$ representing all the increasing choice functions of domain $\{1, 2, ..., m\}$ and codomain $\{1, 2, ..., n\}$. So, the indexed family $< \mathcal{A}_i >, 1 \leq i \leq m$; is the maximal indexed family for the given n and m.The size of the table DCM is $(n - m + 1) \times m$ since

$\max_i(|\ \mathcal{A}_i\ |) = n - m + 1$. The entries DCM$[n - m + 1, j] = 1$, DCM$[i, m] = 1$, $1 \leq i \leq n - m + 1$, $1 \leq j \leq m$. The table DCM can be evaluated using the algorithm SEQTABDCM.

Algorithm SEQTABDCM (algorithm for SEQuential evaluation of the TABle DCM)

Input: n, m for the model S_U.

Output: the table DCM

Method:

1. **for** $i \longleftarrow 1$ **to** m **do**

1.1. DCM$[n - m + 1, i] \longleftarrow 1$

2. **for** $p \longleftarrow 1$ **to** $n - m + 1$ **do**

2.2. DCM$[p, m] \longleftarrow 1$

3. **for** $p \longleftarrow n - m$ **downto** 1 **do**

3.1. **for** $i \longleftarrow m - 1$ **downto** 1 **do**

3.1.1. DCM$[p, i] \longleftarrow$DCM$[p + 1, i]$+DCM$[p, i + 1]$

Assuming that the step 3.1.1. can be performed in time $O(1)$, then the asymptotic complexity of the algorithm is $O(n.m)$, since the size of the table DCM equals $m.(n - m + 1)$.

The algorithm SEQTABDCM is an instance of the algorithm SEQTABD. The table DCM can also be evaluated using the parallel algorithm involving coefficients $\binom{i}{k}$ of Newton's binomial.

Algorithm PARTABDCM (algorithm for PARallel evaluation of the TABle DCM)

Input: n, m for the model S_U. The number of the processors used equals $(n - m + 1).m$. Each processor is dedicated to an entry of the table DCM. Hence, the processors P are enumerated with (p, i).

Output: the table DCM.

Method:

1. **for** $p \longleftarrow n - m + 1$ **downto** 1 **do in parallel**

1.1. **for** $i \longleftarrow m$ **downto** 1 **do in parallel**

1.1.1. DCM$(p, i) \leftarrow \binom{(n-m+1)-p}{m-i+1}$

The time complexity of the algorithm equals the time complexity of the step 1.1.1. The maximal number of the processors used equals $(n-m+1).m$

It has been observed that the entries of the table DCM are the entries of the Pascal's triangle [19].

Example 3.5.1 *For the model* S'_U *and parameters* $m = 4$, $n = 10$ *there is the following table DCM.*

$$\text{DCM}= \left| \begin{array}{cccc} 84 & 28 & 7 & 1 \\ 56 & 21 & 6 & 1 \\ 35 & 15 & 5 & 1 \\ 20 & 10 & 4 & 1 \\ 10 & 6 & 3 & 1 \\ 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{array} \right| .$$

□

3.5.1 The DDCM tables

Given is the model $S_U = << \mathcal{P}_i >, 1 \leq i \leq m; W$ in (1.1); $h \in \{h\}_{\mathcal{P}_i} >$ defined in (1.2), where $\mathcal{P}_i \subseteq \mathcal{A}_i$, $1 \leq i \leq m$, $\max(\mathcal{P}_i) > \max(\mathcal{P}_{i-1})$, $2 \leq i \leq m$. So, for a given indexed family $< \mathcal{P}_i >, 1 \leq i \leq m$; the corresponding indexed family $< \mathcal{P}_i^{\min} >, 1 \leq i \leq m$ is the minimal non−deformed indexed family.

The set of choice functions $\{h\}_{\mathcal{P}_i}$ is symmetric since for the canonical form of a given partial mapping $h(1)$, $h(2)$, ..., $h(t)$, $t < m$, we have $\mathrm{N}[x_t(\mathcal{P}_t)] = \mathrm{N}[x_1(\mathcal{P}_1), x_2(\mathcal{P}_2),..., x_t(\mathcal{P}_t)]$, where $x_1(\mathcal{P}_1) = h(1)$, $x_2(\mathcal{P}_2) = h(2)$, ..., $x_t(\mathcal{P}_t) = h(t)$. Moreover, $\mathrm{N}[x_m(\mathcal{P}_m)] = 1$ and $\mathrm{N}[x(\mathcal{P}_m^{\min})] = 0$ if $x(\mathcal{P}_m^{\min}) \notin \mathcal{P}_m$. The table D for the above model is denoted by DDCM (deformed DCM). The algorithm SEQTABDDCM enables us to evaluate the table DDCM for each particular model S_U.

Algorithm SEQTABDDCM (algorithm for SEQuential evaluation of the TABle DDCM]

Input: the model $S_U =<< \mathcal{P}_i >, 1 \leq i \leq m; W$ in (1.1]; $h \in \{h\}_{\mathcal{P}_i} >$ $u = n - m + 1$, the corresponding minimal non−deformed indexed family $< \mathcal{P}_i^{\min} >, 1 \leq i \leq m$.

Output: the table DDCM that can possess 0 entries.

Method:

1. **for** $i \longleftarrow 1$ **to** m **do**

1.1. **if** $u(\mathcal{P}_i^{\min}) \in \mathcal{P}_i$ **then** DDCM$[n - m + 1, i] \longleftarrow 1$ **else** DDCM$[n - m + 1, i] \longleftarrow 0$

2. **for** $p \longleftarrow 1$ **to** $n - m + 1$ **do**

2.2. **if** $p(\mathcal{P}_m^{\min}) \in \mathcal{P}_m$ **then** DDCM$[p, m] \longleftarrow 1$ **else** DDCM$[p, m] \longleftarrow 0$

3. **for** $p \longleftarrow n - m$ **downto** 1 **do**

3.1. **for** $i \longleftarrow m - 1$ **downto** 1 **do**

3.1.1. **if** $p(\mathcal{P}_i^{\min}) \in \mathcal{P}_i^*$ **then** DDCM$[p, i] \leftarrow \sum_{k=p}^{n-m+1}$DDCM$[k, i + 1]$ **else** DDCM$[p, i] \leftarrow 0$.

The algorithm SEQTABDDCM is an instance of the general sequential algorithm for evaluation of the tables D. The test $p(\mathcal{A}_i) \in \mathcal{P}_i$ and the similar ones can be performed in time $O(1)$. Asymptotic complexity is

$O(u^2.m)$, i.e., the same as for the algorithm SEQTABD. Nevertheless, time complexity of the step 3.1.1. is lower than time complexity of the corresponding step 3.1.2. of the algorithm SEQTABD since the sum $\sum_{k=p}^{n-m+1}$ contains less elements than u. Therefore, time complexity of the algorithm SEQTABDDCM is lower than of the algorithm SEQTABD.

Example 3.5.2 *Given is the following indexed family* $< \mathcal{P}_i >$, $1 \leq i \leq 4$, $\mathcal{P}_1 = \{2, 3\}$, $\mathcal{P}_2 = \{3, 4\}$, $\mathcal{P}_3 = \{4, 5, 6\}$, $\mathcal{P}_4 = \{5, 7\}$ *for the model specified in (1.2). We have to produce the corresponding table DDCM.*

The indexed family $< \mathcal{P}_i >$, $1 \leq i \leq 4$ is a deformed indexed family. The corresponding minimal non–deformed family is $< \mathcal{P}_i^{\min} >$, $1 \leq i \leq 4$, where $\mathcal{P}_1^{\min} = \{2, 3, 4\}$, $\mathcal{P}_2^{\min} = \{3, 4, 5\}$, $\mathcal{P}_3^{\min} = \{4, 5, 6\}$, $\mathcal{P}_4^{\min} = \{5, 6, 7\}$. So, $n = 7$ and $m = 4$. We have $u = \max_i(| \mathcal{P}_i^{\min} |) = 3$. The size of the table DDCM is 3×4. Using the algorithm table DDCM, we have:

$$\text{DDCM}= \left| \begin{array}{cccc} 6 & 4 & 2 & 1 \\ 2 & 2 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array} \right|$$

□

3.5.2 The tables DCM for non–void W_1

We are concerned now with the model S_U=<< $\mathcal{A}_i$ > , $1 \leq i \leq m$; W given in (1.1); W_1; $h \in \{h\}_{\mathcal{A}_i}$ >, where W_1 can be specified by the canonical form. If W_1 is given in (1.3) or W_1 is given in (1.4), then the model S_U and the model (1.2) are congruent. Hence, we have the DDCM table discussed in the previous section.

Suppose, now W_1 is given in (1.5), i.e., for every $z \in Z$ we have to have $h(z) = h(z-1) + 1$, where Z is a set that $Z \subset \{1, 2, ...m\}$. We observe that the corresponding set of choice functions $\{h\}_{\mathcal{A}_i}$ is symmetric. The corresponding table D is denoted by W(1.5)DCM (requirement W_1 given in (1.5) for DCM). The following algorithm SEQTABW(1.5)DCM can be used for evaluation of the table W(1.5)DCM.

Algorithm SEQTABW(1.5)DCM (SEQuential evaluation of TABle W(1.5)DCM)
Input: the set Z
Output: the table W(1.5)DCM
Method:
1. **for** $i \longleftarrow 1$ **to** m **do**

1.1. W(1.5)DCM$[n - m + 1, i] \longleftarrow 1$

2. **for** $i \longleftarrow 1$ **to** $n - m + 1$ **do**

2.1. W(1.5)DCM$[i, m] \longleftarrow 1$

3. **for** $i \longleftarrow m-1$ **downto** 1 **do**

3.1. **for** $j \longleftarrow n-m$ **downto** 1 **do**

3.1.1. **if** $(i+1) \notin Z$ **then** W(1.5)DCM$[j, i] \longleftarrow$W(1.5)DCM$[j, i+1]$ +W(1.5)DCM$[j+1, i]$ **else** W(1.5)DCM$[j, i] \longleftarrow$W(1.5)DCM$[j, i+1]$

In order to observe the correctness of the algorithm, we note the following relations:

* if $(i+1) \in Z$, then $N[j(\mathcal{A}_i)] = N[(j(\mathcal{A}_{i+1})]$.

* * if $(i+1) \notin Z$, then $N[j(\mathcal{A}_i)] = \sum_{k=j}^{n-m+1} N[k(\mathcal{A}_{i+1})]$,

*** $\sum_{k=j}^{n-m+1} N[k(\mathcal{A}_{i+1})] = N[(j+1)(\mathcal{A}_i)] + N[j(\mathcal{A}_{i+1})]$,

for the canonical form of the choice functions $h \in \{h\}_{\mathcal{A}_i}$.

Substituting $N[j(\mathcal{A}_i)]$ =W(1.5)DCM(j, i) and so on, we see that the algorithm is correct indeed. Complexity of the algorithm SEQTABW(1.5)DCM is determined by its size, i.e., it is $m.(n-m)$. The entries of the table W(1.5)DCM can also be evaluated by using the following algorithm PARTABW(1.5)DCM.

Algorithm PARTABW(1.5)DCM (PARallel evaluation of TABle W(1.5)DCM)

Input: the set Z

Output: the table W(1.5)DCM

Method:

1. **for** $i \longleftarrow m$ **downto** 1 **do in parallel**

1.1. $e_i \longleftarrow |\{1, 2, ..., i\} - Z|$

1.2. **for** $j \longleftarrow n-m+1$ **downto** 1 **do in parallel**

1.2.1. W(1.5)DCM$[j, i] \longleftarrow \binom{n-m+1-j}{m-i-e_i}$

Correctness of the above algorithm one can prove noting :

(i) the columns of the table W(1.5)DCM equal to some columns of the corresponding DCM table,

(ii) for each $z \in Z$ that $z < i$ the $i-th$ column of the table W(1.5)DCM equals the $i-th$ column of this table.

Example 3.5.3 *For* $n = 9$, $m = 5$, $Z = \{3, 5\}$ *the corresponding table W(1.5)DCM is as follows:*

$$\text{W(1.5)DCM}= \left|\begin{array}{ccccc} 15 & 15 & 5 & 5 & 1 \\ 10 & 10 & 4 & 4 & 1 \\ 6 & 6 & 3 & 3 & 1 \\ 3 & 3 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array}\right| .$$

□

The other classes of the models concerning increasing choice functions will be considered in the next sections by showing congruence of those models and the models representing sets of the non-decreasing choice functions.

3.6 The tables DCM for monotonic functions

The goal is to show that the tables D used for the sets of increasing choice functions represent also the structure of the sets of non-decreasing choice functions.

The tables D for monotonic choice functions are the same as the tables D for increasing choice functions. We will study now the structure of sets of the monotonic choice functions in more detail.

Let the unranking model $S_U^1 =<< \mathcal{R}_i >, 1 \leq i \leq m; W^1$ specified in (1.10); W_1^1; $h \in \{h\}_{\mathcal{R}_i} >$ be given, where $\mathcal{R}_i \subseteq \mathcal{E}_i = \{1, 2, ..., n-m+1\}$ and the requirement W_1^1 can be expressed in the canonical form. The model S_U^1 is reduced and the Q property holds. We know that the model S_U^1 represents a subset of non-decreasing choice functions.

The model $S_U^2 =<< \mathcal{P}_i >, 1 \leq i \leq m; W^2$ given in $(1.1); W_1^2; h \in \{h\}_{\mathcal{P}_i} >$ represents a subset of increasing choice functions, where $\mathcal{P}_i \subseteq \mathcal{A}_i = \{i, i+1, ..., n-m+i\}$, the requirement W_1^2 can be given in the canonical form. The model S_U^2 is reduced and the Q property holds.

Theorem 3.6.1 *For every model S_U^1 there exists a model S_U^2, such that S_U^1 and S_U^2 are congruent.*

Proof. Consider now the model $\overline{S}_U^1 =<< \mathcal{E}_i >, 1 \leq i \leq m; W^1$ given in (1.10); $h \in \{h\}_{\mathcal{E}_i} >$ that represents the full set of non-decreasing choice functions. The model $\overline{S}_U^2 =<< \mathcal{A}_i > , 1 \leq i \leq m; W^2 = (1.1); h \in \{h\}_{\mathcal{A}_i} >$ represents the full set of the increasing choice functions.

(i) We will prove now that the models $\overline{S}_U^1$ and $\overline{S}_U^2$ are congruent.

Let $< x_1(\mathcal{E}_1), x_2(\mathcal{E}_2), ..., x_m(\mathcal{E}_m) >$ be the canonical form of a choice function $h \in \{h\}_{\mathcal{E}_i}$. Since $\mid \mathcal{E}_1 \mid = n-m+1$, so the range for changing the parameters x_i is as follows: $1 \leq x_i \leq n-m+1$. The requirement W^1 specified in the canonical form is as follows: $x_i \geq x_{i-1}$, $2 \leq i \leq m$.

Similarly, $< x_1(\mathcal{A}_1), x_2(\mathcal{A}_2), ..., x_m(\mathcal{A}_m) >$ is the choice function $h \in \{h\}_{\mathcal{A}_i}$. We have $\mid \mathcal{A}_1 \mid = n-m+1$, so we have also $1 \leq x_i \leq n-m+1$. The requirements W^2 and W^1 specified using the canonical form equal.

Therefore, for each choice function $< x_1(\mathcal{E}_1), x_2(\mathcal{E}_2), ..., x_m(\mathcal{E}_m) >$ there is equal choice function $< x_1(\mathcal{A}_1), x_2(\mathcal{A}_2), ..., x_m(\mathcal{A}_m) >$. Moreover, $\mid \{h\}_{\mathcal{E}_i} \mid = \mid \{h\}_{\mathcal{A}_i} \mid$. Hence, the systems S_U^1 and S_U^2. are congruent. Therefore, the lemma (i) is proved.

(*ii*) We prove now the congruence of the models S_U^1 and S_U^2 for the general case. Restriction of $\{h\}_{\mathcal{E}_i}$ by deformation of sets $\mathcal{E}_i$ is equivalent to restriction of $\{h\}_{\mathcal{A}_i}$ by the corresponding deformation of $\mathcal{A}_i$.

Since for any additional requirement W_1^1 we can construct the additional requirement W_1^2 that their canonical forms equal, so for any restriction of the set $\{h\}_{\mathcal{E}_i}$ we can made the corresponding restriction of the set $\{h\}_{\mathcal{A}_i}$. Then, the structural numbers for $\{h\}_{\mathcal{E}_i}$ and $\{h\}_{\mathcal{A}_i}$ equal. Therefore, if there is a table D that represents a restricted model S_U^2, then there is a restricted model S_U^1 represented by the table D.

Therefore, we have proved (*ii*).

Hence, for every model S_U^2 there exists a model S_U^1, that S_U^1 and S_U^2 are congruent.

□

Therefore, any table D that represents the structure of a symmetric subset of monotonic choice functions equals the table D that represents a corresponding subset of increasing choice functions.

We are concerned now with a collection of the models $S_U^2 =<< \mathcal{T}_i >$, $1 \leq i \leq m$, $\mathcal{T}_i = \{r.(i-1)+1, r.(i-1)+2, ..., n-r.(m-i)\}$; W given in (1.17); $\{h\}_{\mathcal{T}_i} >$, where $r \in \{0, 1, 2, ..., w\}$, $w =$integer(n/m). It was shown that the Q property holds for the model S_U^2.

Putting $r = 0$, the set $\{h\}_{\mathcal{T}_i}$ given in the canonical form is the same as the full set of all non-decreasing choice functions for fixed n and m. If $r = 1$, then the corresponding set $\{h\}_{\mathcal{T}_i}$ is the set of increasing choice functions. For $r > 1$ we get a corresponding subset of non-decreasing choice functions. For $r > 1$ the canonical form of the set $\{h^1\}_{\mathcal{T}_i}$ is a subset of the canonical form of the corresponding set of increasing choice functions.

Let two models S_U^1 and S_U^2 be given that $S_U^1 =<< \mathcal{T}_i^1 >$, $1 \leq i \leq m$, $\mathcal{T}_i^1 = \{r_1.(i-1)+1, r_1.(i-1)+2, ..., n_1 - r_1.(m-i)\}$; W given in (1.17); $\{h^1\}_{\mathcal{T}_i} >$, $S_U^2 =<< \mathcal{T}_i^2 >$, $1 \leq i \leq m$, $\mathcal{T}_i^2 = \{r_2.(i-1)+1, r_2.(i-1)+2, ..., n_2 - r_2.(m-i)\}$; W given in (1.17); $\{h^2\}_{\mathcal{T}_i} >$.

Theorem 3.6.2 *The systems S_U^1 and S_U^2 are congruent if and only if $n_2 = n_1 - (m-1)(r_1 - r_2)$.*

Proof. We have $\mid \mathcal{T}_i^1 \mid = n_1 - r_1.(m-1)$ and $\mid \mathcal{T}_i^2 \mid = n_2 - r_2.(m-1)$, $1 \leq i \leq m$. The systems are congruent only if $\mid \mathcal{T}_i^1 \mid = \mid \mathcal{T}_i^2 \mid$. That happens if $n_1 - r_1.(m-1) = n_2 - r_2.(m-1)$. Therefore, we have shown that the systems S_U^1 and S_U^2 are congruent only if $n_2 = n_1 - (m-1)(r_1 - r_2)$.

We prove now the sufficient condition.

We have, $x_i(\mathcal{T}_i^1) = r_1.(i-1) + (x_i - 1)$.

Let $< x_1(\mathcal{T}_1^1), x_2(\mathcal{T}_2^1), ..., x_i(\mathcal{T}_i^1) >$ be a corresponding partial mapping. The number of extensions of it to a partial mapping $< x_1(\mathcal{T}_1^1), x_2(\mathcal{T}_2^1), ..., x_i(\mathcal{T}_i^1), x_{i+1}(\mathcal{T}_{i+1}^1) >$ equals $[n_1 - r_1.(m-1)] - x_i + 1$.

Similarly, the number of extensions of the corresponding partial mapping $x_1(\mathcal{T}_1^2)$, $x_2(\mathcal{T}_2^2)$, ..., $x_i(\mathcal{T}_i^2) >$ to the partial mapping $< x_1(\mathcal{T}_1^1)$, $x_2(\mathcal{T}_2^1)$, ..., $x_i(\mathcal{T}_i^1)$, $x_{i+1}(\mathcal{T}_{i+1}^1) >$ equals $[n_2 - r_2.(m-1)] - x_i + 1$.

If $n_2 = n_1 - (m-1)(r_1 - r_2)$, then $[n_2 - r_2.(m-1)] - x_i + 1 = n_1 - r_1(m-1) - x_i + 1$, so it is the same as for the model S_U^2. Since we can continue it changing i up to m, so $\mathrm{N}[x_i(\mathcal{T}_i^1)] = \mathrm{N}[x_i(\mathcal{T}_i^2)]$. Hence, the set of structural numbers for the model S_U^1 equals the set of structural numbers for the model S_U^2. Therefore, the models S_U^1 and S_U^2 are congruent. That finishes the proof.

□

Corollary 3.6.1 *The table D that represents the structure of any model* $S_U^2 = << \mathcal{T}_i >, 1 \leq i \leq m, \mathcal{T}_i = \{r.(i-1)+1, r.(i-1)+2, ..., n - r.(m-i)\}$; $W(i)$ *given in* (1.17); $\{h\}_{\mathcal{T}_i} >$ *is the DCM table.*

Proof. The set $\{h\}_{\mathcal{T}_i}$ for $r = 1$ is the full set of increasing choice functions. The structure of $\{h\}_{\mathcal{T}_i}$ is represented by the table DCM$[(n - m + 1) \times m]$. From the above theorem we conclude that any model S_U^2 and the model $<< \mathcal{T}_i >, 1 \leq i \leq m, \mathcal{T}_i = \{r.(i-1)+1, r.(i-1)+2, ..., n - r.(m-i)\}$; W given in (1.17); $\{h\}_{\mathcal{T}_i} >$ are congruent, where $r = 1$. Therefore, the structure of $\{h\}_{\mathcal{T}_i}$ is represented by the corresponding table DCM.

□

We have to emphasize that the models S_U^1 and S_U^2 are non–isomorphic. One can prove it observing that $\mathcal{T}_i^1 \subset \{1, 2, ..., n_1\} = \mathcal{U}$, while $\mathcal{T}_i^2 \subset \{1, 2, ..., n_2\} = Y$. Since $n_1 \neq n_2$, so $\mid \mathcal{U} \mid \neq \mid Y \mid$. Then, there is not one–to–one mapping $\psi : \mathcal{U} \longrightarrow Y$. Therefore, the models S_U^1 and S_U^2 are non–isomorphic, indeed.

Example 3.6.1 *There are tables DCM that represent structures of ranked subsets of non-decreasing choice functions for the fixed values* $n = 12$, $m = 4$ *and for* $r = 0$, $r = 1$, $r = 2$ *and* $r = 3$.

DCM $r = 0$

DCM $r = 1$

DCM $r = 2$

DCM $r = 3$

370	81	13	1
289	68	12	1
221	56	11	1
165	45	9	1
120	36	8	1
84	28	7	1
56	21	6	1
35	15	5	1
20	10	4	1
10	6	3	1
4	3	2	1
1	1	1	1

□

So, the collection of models $S_U^2 =<< \mathcal{T}_i >, 1 \leq i \leq m, \mathcal{T}_i = \{r.(i-1)+1, r.(i-1)+2, ..., n - r.(m-i)\}$; W given in (1.17); $\{h\}_{\mathcal{T}_i} >$ possess a hierarchical structure that is defined by subtables of the table DCM.

3.6.1 Tables D for monotonic choice functions with variable W

We are going to extend the result given in the previous section. Given is the system $S_U^1 =< < \mathcal{R}_i >, 1 \leq i \leq m$; W given in (3.5); $\{h^1\}_{\mathcal{R}_i} >$, where $\mathcal{R}_i = \{ 1+a_2 +a_3 +....+a_i, 1+a_2 + a_3 +....+a_i +1, ..., n_1-a_m-a_{m-1}-... -a_{i+1}\}$, $n_1 > a_1 + a_2 + ... + a_m$.

$$h(i) \geq h(i-1) + a_i, \quad a_i \in \{0, 1, ...\}, \text{ where } 2 \leq i \leq m \qquad (3.5)$$

Setting values a_i for each i we can model widely sets of non-decreasing choice functions $\{h\}_{\mathcal{R}_i}$. Since $(1 + a_2 + a_3 + ... + a_i) \geq (1 + a_2 + a_3 + ... + a_{i-1})$ and $(n_1 - a_{m-1} - a_{m-1} - ... - a_{i+1}) \geq (n_1 - a_{m-1} - a_{m-1} - ... - a_{i+1} - a_i)$, so the model is reduced and the Q property holds.

Let the representation model $S_U^2 =<< \mathcal{T}_i >, 1 \leq i \leq m, \mathcal{T}_i = \{r_2.(i-1)+1, r_2.(i-1)+2, ..., n_2 - r_2.(m-i)\}$; W given in (1.17); $\{h^2\}_{\mathcal{T}_i} >$ be given.

Proposition 3.6.1 *The models S_U^1 and S_U^2 are congruent if $n_2 - r_2.(m-1) = n_1 - a_{m-1} - a_{m-1} - ... - a_2$.*

Proof. We have

$\| \mathcal{R}_i \| = (n_1 - a_m - a_{m-1} - ... - a_{i+1}) - (1 + a_2 + a_3 + ... + a_i) = n_1 - a_m - a_{m-1} - ... - a_{i+1} - a_i - a_{i-1} - .... - a_2 - 1$;

$\| \mathcal{I}_i \| = n_2 - r_2.(m-i) - r_2(i-1) + 1 = n_2 - r_2.(m-1) - 1$;

We observe, the cardinals $\| \mathcal{R}_i \|$ and $\| \mathcal{I}_i \|$ are fixed for all indexed sets $\mathcal{R}_i$ and $\mathcal{I}_i$, $1 \leq i \leq m$.

If the models S_U^1 and S_U^2 are congruent, then $\| \mathcal{R}_i \| = \| \mathcal{I}_i \|$. So, $n_1 - a_m - a_{m-1} - ... - a_{i+1} - a_i - a_{i-1} - .... - a_2 - 1 = n_1 - a_m - a_{m-1} - ... - a_{i+1} - a_i - a_{i-1} - .... - a_2 - 1$ is the necessary condition for the congruence of S_U^1 and S_U^2. After obvious evaluations, we have $n_2 - r_2.(m-1) = n_1 - a_m - a_{m-1} - ... - a_2$ is the necessary condition for the congruence of the models.

Since $< \mathcal{R}_i >$, $1 \leq i \leq m$ and $< \mathcal{I}_i >$, $1 \leq i \leq m$ are the minimal non-deformed indexed families and since $\| \mathcal{R}_i \| = \| \mathcal{I}_i \|$, so for every x that $x(\mathcal{R}_i) \in \mathcal{R}_i$, we have $x(\mathcal{I}_i) \in \mathcal{I}_i$.

Let f_1 be a partial choice mapping $< x_1(\mathcal{R}_1), x_2(\mathcal{R}_2), ..., x_i(\mathcal{R}_i) >$, while f_2 being the partial choice mapping $< x_1(\mathcal{I}_1), x_2(\mathcal{I}_2), ..., x_i(\mathcal{I}_i) >$. Basing on the requirement 3.5, the number of extensions of f_1 to $< x_1(\mathcal{R}_1), x_2(\mathcal{R}_2), ..., x_i(\mathcal{R}_i), x_{i+1}(\mathcal{R}_i) >$ equals $\| \mathcal{R}_i \| - x_i$, while basing on the requirement 1.17 the number of extensions of f_2 to $< x_1(\mathcal{I}_1), x_2(\mathcal{I}_2), ..., x_i(\mathcal{I}_i), x_{i+1}(\mathcal{I}_i) >$ equals $\| \mathcal{I}_i \| - x_i$. By induction on i we prove that the number of extensions of f_1 to full choice functions equals the number of extensions of f_2 to corresponding full choice functions. Hence, the entries of the tables $\mathbb{D}$ for the model S_U^1 are respectively equal to the entries of the table $\mathbb{D}$ for the model S_U^2.

Therefore, the systems S_U^1 and S_U^2 are congruent if and only if $n_2 - r_2.(m-1) = n_1 - a_m - a_{m-1} - ... - a_2$.

□

We immediately get the following conclusion.

The structure of the set $\{h\}_{\mathcal{R}_i}$ is represented by the table $\mathbb{DCM}[u, m]$, where $u = n_1 - a_m - a_{m-1} - ... - a_2$.

Example 3.6.2 *Given is the model:* $S_U = << \mathcal{R}_1 = \{1,2,3\}, \mathcal{R}_2 = \{3,4,5\}, \mathcal{R}_3 = \{4,5,6\}, \mathcal{R}_4 = \{7,8,9\}, \mathcal{R}_5 = \{8,9,10\} >, 1 \leq i \leq 5$;

$W : h(i) \geq h(i-1) + a_i$, $2 \leq i \leq 5$, $a_2 = 2$, $a_3 = 1$, $a_4 = 3$, $a_5 = 1$; $\{h\}_{\mathcal{R}_i} >$

The table $\mathbb{D}$ for this case is as follows:

$$\mathbb{DCM}[3 \times 5] = \begin{Vmatrix} 15 & 10 & 6 & 3 & 1 \\ 5 & 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{Vmatrix}.$$

□

3.6.2 *The tables D for deformed families*

Given is a model $S_U^1 =< < \mathcal{M}_i >, 1 \leq i \leq m; W$ given in (3.5); $\{h\}_{\mathcal{M}_i} >$, where $\mathcal{M}_i \subseteq \mathcal{R}_i$, $\mathcal{R}_i = \{ 1 + a_2 + a_3 + ... + a_i, 1 + a_2 + a_3 + ... + a_i + 1, ..., n_1 - a_m - a_{m-1} - ... - a_{i+1}\}$, $n_1 > a_1 + a_2 + ... + a_m$, $\max(\mathcal{M}_i) \geq \max(\mathcal{M}_{i-1}) + a_i$, $\min(\mathcal{M}_i) \geq \min(\mathcal{M}_{i-1}) + a_i$, $2 \leq i \leq m$. One can show that the model is reduced and the Q property holds.

We are concerned with evaluation of the table D for the model S_U^1. The models $< < \mathcal{R}_i >, 1 \leq i \leq m; W$ given in (3.5); $\{h\}_{\mathcal{R}_i} >$ and $<< \mathcal{A}_i > , 1 \leq i \leq m; W$ given in (1.1); $\{h\}_{\mathcal{A}_i} >$ are congruent. Using this relation one can show that the models S_U^1 and $S_U^2 =<< \mathcal{P}_i >$, $1 \leq i \leq m; W$ given in (1.1); $\{h\}_{\mathcal{P}_i} >$ are congruent, where $\mathcal{P}_i \subseteq \mathcal{A}_i$, $\max(\mathcal{P}_i) \geq \max(\mathcal{P}_{i-1}) + a_i$, $\min(\mathcal{P}_i) \geq \min(\mathcal{P}_{i-1}) + a_i$, $2 \leq i \leq m$.

It was shown that the table DDCM represents the structure of the set $\{h\}_{\mathcal{P}_i}$. Hence, the table DDCM represents the structure of the set $\{h\}_{\mathcal{M}_i}$, too.

Example 3.6.3 *Given is the representation model* $S_U =< < \mathcal{M}_i > , 1 \leq i \leq m; W$ given in (3.5); $\{h^1\}_{\mathcal{M}_i} >$, *where* $m = 5$, $\mathcal{M}_1 = \{1, 2, 3\}$, $\mathcal{M}_2 = \{3, 4, 5\}$, $\mathcal{M}_3 = \{6, 7, 8, 9\}$, $\mathcal{M}_4 = \{7, 8, 9, 10, 11\}$, $\mathcal{M}_5 = \{10, 11, 12, 13\}$, $a_2 = 2$, $a_3 = 3$, $a_4 = 1$, $a_5 = 2$. *The task is to build the corresponding table D.*

The models S_U and $<< \mathcal{P}_i >, 1 \leq i \leq m; W$ given in (1.1); $h \in \{h\}_{\mathcal{P}_i} >$ are congruent, where $\mathcal{P}_i \subseteq \mathcal{A}_i = \{i, i+1, ..., n-m+1\}$, $m = 5$, $n = 13$, $\mathcal{P}_1 = \{1, 2, 3\}$, $\mathcal{P}_2 = \{2, 3, 4\}$, $\mathcal{P}_3 = \{3, 4, 5, 6\}$, $\mathcal{P}_4 = \{4, 5, 6, 7, 8\}$, $\mathcal{P}_5 = \{6, 7, 8, 9\}$. The structure of the set $\{h\}_{\mathcal{P}_i}$ is represented by the following DDCM table.

$$\text{DDCM}[5 \times 5] = \left| \begin{array}{ccccc} 51 & 33 & 14 & 4 & 0 \\ 28 & 19 & 10 & 4 & 1 \\ 9 & 9 & 6 & 3 & 1 \\ 0 & 0 & 3 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right|$$

Hence, the structure of $\{h^1\}_{\mathcal{M}_i}$ is represented by the above DDCM table.

□

Similarly, we evaluate the tables D for the other models concerning monotonic choice functions of deformed families.

3.6.3 *Tables D for non-increasing choice functions*

We used the term monotonic choice functions mainly in order to mean sets of non-decreasing choice functions. Nevertheless, we could also use sets of non-increasing choice functions. We will consider now the relations

between sets of non-decreasing choice functions and sets of non-increasing choice functions.

Given is the system $S_U^1 =<< \mathcal{L}_i >, 1 \leq i \leq m, \mathcal{L}_i = \{r.(m-i)+1, r.(m-i)+2, ..., n-r.(i-1)\}$; W given in (3.6); $\{h\}_{\mathcal{L}_i} >$. The set $\{h\}_{\mathcal{L}_i}$ is the set of non-increasing choice functions.

$$h(i) \leq h(i-1)+r \tag{3.6}$$

The set $\{h\}_{\mathcal{T}_i}$ is the set of non-decreasing choice functions for the system $S_U^2 =<< \mathcal{T}_i >, 1 \leq i \leq m, \mathcal{T}_i = \{r.(i-1)+1, r.(i-1)+2, ..., n-r.(m-i)\}$; W given in (1.17); $\{h\}_{\mathcal{T}_i} >$.

Proposition 3.6.2 *The systems S_U^1 and S_U^2 for fixed values n, r and m are isomorphic.*

Proof. We have $\mathcal{L}_i \subseteq \mathcal{U} = \{1, 2, ..., n\}$, $\mathcal{T}_i \subseteq Y = \{1, 2, ..., n\}$, so $\mathcal{U} = Y$.

Let $\psi : \mathcal{U} \longrightarrow \mathcal{U}$ that $\psi(x) = n - x + 1$, for every $x \in \mathcal{U}$, while φ being identity mapping.

We observe that $\psi(\mathcal{T}_i) = \mathcal{L}_i$.

We prove now that for every $h_1 \in \{h\}_{\mathcal{L}_i}$, we have $h_2 \in \{h\}_{\mathcal{T}_i}$ that $\psi(h_1) = h_2$.

Observe, if $h_1 =< h_1(1), h_1(2), ..., h_1(m) >$, then we have to have $h_1(i) < h_1(i-1) - r$, $2 \leq i \leq m$. Since $\psi(h_1(i)) = h_2(i)$ and $\psi(h_1(i)) = n-h_1(i)+1$, so $(h_2(i-1) = n-h_1(i-1)+1$. Therefore, $h_2(i) > h_2(i-1)+r$, $2 \leq i \leq m$. Hence, the choice function $h_2 = \psi(h_1)$ belongs to $\{h\}_{\mathcal{T}_i}$. That finishes the proof.

□

Corollary 3.6.2 *Every representation of a set of combinatorial objects by non-decreasing choice functions can also be transformed into the corresponding model concerning non-increasing choice functions.*

Proof. Since for every model concerning a set of non-decreasing choice functions there is an isomorphic model concerning a set of non-increasing choice functions, so a set of combinatorial objects modeled by a set of non-decreasing choice functions can also be modeled by a set of non-increasing choice functions.

□

3.7 The tables D for choice bijections

Given is the model $S_U =<< \mathcal{E}_i >, 1 \leq i \leq m, \mathcal{E}_i = \{1, 2, ..., n\}$; $W = (1.19); \{h\}_{\mathcal{E}_i} >$, so the set $\{h\}_{\mathcal{E}_i}$ is the set of choice bijections.

Theorem 3.7.1 *The table D corresponding to* $\{h\}_{\mathcal{E}_i}$ *is regular.*

Proof. Let $S'_U = << \mathcal{E}'_i >>$, $1 \leq i \leq m+s$, $\mathcal{E}'_i = \{1, 2, ..., n+q\}$; $W = (1.19)$; $\{h'\}_{\mathcal{E}'_i} >$ be a related representation model.

We consider a partial mapping $< h(1), h(2), ..., h(k) >$ for the model S_U. The number of extensions of this partial mapping to a choice function $h \in \{h\}_{\mathcal{E}_i}$ equals $(n-k).(n-k+1)....(n-m+1)$.

Correspondingly, $< h'(1), h'(2), ..., h'(k+s) >$ is a partial mapping for the model S'_U. The number of extensions of this partial mapping to a choice function $h' \in \{h'\}_{\mathcal{E}'_i}$ equals $((n+q)-(k+s)).((n+q)-(k+s)+1).$ $...((n+q)-(m+s)+1)$.

We have $(n-k).(n-k+1)....(n-m+1) = ((n+q)-(k+s)).((n+q)$ $-(k+s)+1).$ $...((n+q)-(m+s)+1)$ if and only if $q=s$. Then, D$[j,$ $i]$ =D$'[j+s, i+s]$. So, the table D that represents the structure of the set $\{h\}_{\mathcal{E}_i}$ is regular. That finishes the proof.

□

If $n=m$, then $\{h\}_{\mathcal{E}_i}$ represents the set of permutations. The corresponding table D we denote by DP.

The entries DP$[k,i]$ can be evaluated by an instance of the parallel formula (3.4) as follows: DP$[k,$ $i]$ $= (n-i)!$, $1 \leq k \leq n$. For development of the instance of the sequential formula (3.3), we observe that DP$[k,$ $i]$ $= (n-i).$DP$[k,$ $i+1]$ or DP$[k,$ $i]$ $= \sum_{p(\mathcal{E}_i)\in\mathcal{E}_i^*}DP[p,$ $i+1]$, where $|\mathcal{E}_i^*| = n-i+1$.

For instance, if $n=6$, we have the following DP table.

$$\text{DP}[6\times 6] = \left|\begin{array}{cccccc} 120 & 24 & 6 & 2 & 1 & 1 \\ 120 & 24 & 6 & 2 & 1 & 1 \\ 120 & 24 & 6 & 2 & 1 & 1 \\ 120 & 24 & 6 & 2 & 1 & 1 \\ 120 & 24 & 6 & 2 & 1 & 1 \\ 120 & 24 & 6 & 2 & 1 & 1 \end{array}\right|$$

If $m<n$, then $\{h\}_{\mathcal{E}_i}$ represents the set of variations, the corresponding table D we denote by DV.

Let the table DCM$[u\times m]$ be given, where $u = n-m+1$.

The table DV, we evaluate using the following formula

$$\text{DV}[i,\ j] = \text{DCM}[1,\ j].\text{DP}[i,\ j], \tag{3.7}$$

where $1 \leq i \leq n$, $1 \leq j \leq m$. Correctness of the formula (3.7) one can prove by considering the elementary properties of permutations.

Example 3.7.1 *For* $n=9$ *and* $m=5$, *we have the table* $DV[9\times 5]$.

$$\mathrm{DV} = \begin{vmatrix} 1680 & 210 & 30 & 5 & 1 \\ 1680 & 210 & 30 & 5 & 1 \\ 1680 & 210 & 30 & 5 & 1 \\ 1680 & 210 & 30 & 5 & 1 \\ 1680 & 210 & 30 & 5 & 1 \\ 1680 & 210 & 30 & 5 & 1 \\ 1680 & 210 & 30 & 5 & 1 \\ 1680 & 210 & 30 & 5 & 1 \\ 1680 & 210 & 30 & 5 & 1 \end{vmatrix}$$

□

Another method for evaluating the entries of the table DV can be done using the following instance of the sequential formula (3.3):

$$\mathrm{DV}[k,i] = \mathrm{DV}[k,i+1].(n-i), 1 \leq i \leq m-1, k \leq n \qquad (3.8)$$

The formula (3.8) is simpler than the formula (3.7). Nevertheless, the formula (3.7) is an instance of the parallel formula (3.4).

3.7.1 Symmetric subsets of bijections

Given is the representation model $S_U = < < C_i >, 1 \leq i \leq m, C_i \subseteq \{1, 2, ..., n\}; W$ given in (1.19); $\{h\}_{C_i} >$ that is reduced and the Q property holds. For the model S_U the symmetry of the set $\{h\}_{C_i}$ depends on the indexed family $< C_i >$, $1 \leq i \leq m$. Therefore, the table D for representing the structure of $\{h\}_{C_i}$ does not exist for all the models S_U satisfying the general formula.

Example 3.7.2 *For the following family* $< C_i >$, $1 \leq i \leq m$, *the set* $\{h\}_{C_i}$ *is symmetric.*

$$C_1 = \{1,2\}\,,\ C_2 = \{3,4\}\,,\ C_3 = \{5\}\,,\ C_4 = \{1,2,3,4\}\,,\ C_5 = \{1,2,3,4\}\,.$$

We list the choice functions as permutations, so the set $\{h\}_{C_i}$ is represented by the following set of permutations {13524, 13542, 14523, 14532, 23514, 23541, 24513, 24531}. Observe if $h_1 : 1 \longrightarrow 1$ and $h_2 : 1 \longrightarrow 2$, then the number of their extensions equals four. For partial mappings $h_1 : 1 \longrightarrow 1,\ 2 \longrightarrow 3$; $h_2 : 1 \longrightarrow 1,\ 2 \longrightarrow 4$; $h_3 : 1 \longrightarrow 2,\ 2 \longrightarrow 3$; $h_4 : 1 \longrightarrow 2,\ 2 \longrightarrow 4$ the numbers of their extensions equal 2. For each

partial mapping h of the subfamily $< C_i >$, $1 \leq i \leq 3$ the number of its extensions equals two. For each partial mapping of the subfamily $< C_i >$, $1 \leq i \leq 4$ the number of its extensions is one. Therefore, the set of choice functions $\{h\}_{C_i}$ is symmetric.

□

Example 3.7.3 *For the following deformed indexed family* $< C_i >$, $1 \leq i \leq 5$, *the set* $\{h\}_{C_i}$ *is not symmetric.*

$$C_1 = \{1,2\},\ C_2 = \{3,4\},\ C_3 = \{5\},\ C_4 = \{1,2,3\},\ C_5 = \{1,2,3,4\}$$

$\{h\}_{C_i}$ = {13524, 14523, 14532, 23514, 24513, 24531}. One can observe asymmetry of this set noting that for $f_1 : 1 \longrightarrow 1,\ 2 \longrightarrow 4,\ 3 \longrightarrow 5$ we have two extensions 14523 and 14532, while for $f_2 : 1 \longrightarrow 1,\ 2 \longrightarrow 3,\ 3 \longrightarrow 5$ there is only one extension 13524.

□

Theorem 3.7.2 *The set of choice bijections* $\{h\}_{C_i}$ *of the model* S_U *is symmetric if and only if, for each two indexed sets* C_i, C_j *there is* $C_i \cap C_j = C_j$ *or* $C_i \cap C_j = C_i$ *or* $C_i \cap C_j = \emptyset$.

Proof. Let $< C_k >$, $1 \leq k \leq i-1$, be any subfamily of the indexed family $< C_i >$, $1 \leq i \leq m$ that the assumption holds.

Suppose, f_1, f_2 are two partial mappings of the subfamily $< C_k >$, $1 \leq k \leq i-1$, such that $f_1(k) = f_2(k)$ for $k \neq j$ and $f_1(k) \neq f_2(k)$ for $k = j$. Let f denote the partial mapping which is the common part of f_1 and f_2, while CD^f being the codomain of f. We denote by p_1 and p_2 extensions of f_1, f_2 for the subfamily $< C_k >$, $1 \leq k \leq i$, respectively. If $C_i \cap C_j = C_j$, then the number of extensions p_1 equals $| C_i - CD^f - \{f_1(j)\} |$, while the number of extensions p_2 equals $| C_i - CD^f - \{f_2(j)\} |$. If $C_i \cap C_j = \emptyset$, then the number of extensions p_1 equals $| C_i - CD^f |$, while the number of extensions p_2 equals $| C_i - CD^f |$. Therefore, the number of extensions p_1 equals the number of extensions p_2. If it is satisfied for every par j, i of indices such that $j < i$ and $i \leq m$, then the numbers of extensions of f_1 and f_2 onto choice functions $h \in \{h\}_{C_i}$ of the indexed family $< C_i >$, $1 \leq i \leq m$, equal. That proves sufficient condition.

For proving necessary condition, we take two choice bijections h_1, $h_2 \in \{h\}_{C_k}$, $1 \leq k \leq m$, such that $h_1(k) = h_2(k)$ for $k \neq i$ and $k \neq j$ and $h_1(k) \neq h_2(k)$ for $k = i$ or $k = j$. Let f denote the common part of h_1 and h_2, i.e., the partial mapping, where $h_1(k) = h_2(k)$. Then, the number of possible extensions of f to choice functions $h \in \{h\}_{C_k}$ equals $| C_j - CD^f - \{h_1(i)\} |$ or it is equal to $| C_j - CD^f - \{h_2(i)\} |$, respectively. We have

$| C_j - CD^f - \{h_1(i)\} | = | C_j - CD^f - \{h_2(i)\} |$ only if $((h_1(i) \in C_j - CD^f)$ and $(h_2(i) \in C_j - CD^f))$ or $((h_1(i) \notin C_j - CD^f)$ and $(h_2(i) \notin C_j - CD^f))$. That happens if $C_i \cap C_j = C_j$ or $C_i \cap C_j = C_i$ or $C_i \cap C_j = \emptyset$. Hence, the necessary condition is proved.

□

Consider now the table DS that represents the structure of symmetric subsets of bijections. The indexed family $< C_i >$ satisfies conditions specified in Theorem 3.7.2. We denote by d_i the number of indexed sets C_j such that $C_j \subseteq C_i$, $1 \leq j \leq i-1$. If assumption of Theorem 3.7.2 holds, then $C_n = \{1, 2, ..., n\}$. For each value $k \leq n$, we have $\mathrm{DS}[k, n] = 1$. The entries $\mathrm{DS}[k, i]$, $1 \leq i \leq m-1$, can be evaluated by (3.9) starting from $m-1$ downto 1.

$$\mathrm{DS}[k, i] = \mathrm{DS}[k, i+1].(|\ C_{i+1}\ | - d_{i+1}),\ 1 \leq i \leq m-1, k \leq n \qquad (3.9)$$

Example 3.7.4 *For the following indexed family* $C_1 = \{1, 2\}$, $C_2 = \{3, 4\}$, $C_3 = \{5, 6\}$, $C_4 = C_5 = \{1, 2, 3, 4, 5, 6\}$, *we have the corresponding table DS.*

$$\mathrm{DS} = \begin{vmatrix} 24 & 0 & 0 & 2 & 1 \\ 24 & 0 & 0 & 2 & 1 \\ 0 & 12 & 0 & 2 & 1 \\ 0 & 12 & 0 & 2 & 1 \\ 0 & 0 & 6 & 2 & 1 \\ 0 & 0 & 6 & 2 & 1 \end{vmatrix}$$

□

If we have the table DS, then specification of the indexed family is not needed since the table DS alone represents the full model.

If Theorem 3.7.2 were used for testing symmetry of a given indexed family and if we assume that the operations on a set have asymptotic complexity $O(1)$, then the complexity of the corresponding algorithm would be $O(m^2)$.

3.7.2 Bijections piecewise non-decreasing

There are given numbers $k_1, k_2, ..., k_m$ that $k_1 + k_2 + ... + k_m = n$. The set of indices $\{1, 2, ..., n\}$ is distributed into m blocks $I_1, I_2, ...I_m$, such that $I_z = \{k_1 + k_2 + ... + k_{z-1} + 1, k_1 + k_2 + ... + k_{z-1} + 2, ..., k_1 + k_2 + ... + k_{z-1} + k_z\}$. We use the following requirement W.

$$h(i) \geq h(i-1)+r \text{ for every } i, i-1 \in I_z,\ 1 \leq z \leq m,\ r \geq 0 \qquad (3.10)$$

We are concerned with the model $S_U =$

$$<< \mathcal{T}_i >, 1 \leq i \leq n, \mathcal{T}_i \subseteq \{1,2,...,n\}; W \text{ see (1.19)}; W_1 \text{ see (3.10)}; \{h\}_{\mathcal{T}_i} >, \qquad (3.11)$$

see (2.39).

We remind here that satisfiability 1.19 means that $h \in \{h\}_{\mathcal{T}_i}$ is the choice bijection.

Proposition 3.7.1 *The set of choice functions* $\{h\}_{\mathcal{T}_i}$ *is symmetric for each enabled value* r.

Proof. Suppose, $r = 0$.

Then, for each set of indexes I_z we have to represent the set of monotonic partial mappings $h_z : I_z \longrightarrow \{1, 2, ..., n\} - CD^1 - CD^2 - ... - CD^{z-1}\}$, where CD^1 is the codomain of the partial mapping h_1 and so on. Since, the cardinal $|\ \{1, 2, ..., n\} - CD^1 - CD^2 - ... - CD^{z-1}\}\ |$ is fixed, so the set of partial mappings $\{\ h_z\}$ is symmetric. Then, the partial mapping $h(1)$, $h(2)$, ..., $h(k_1+k_2+...+k_z)$ is concatenation of the partial mappings $< h_1$, h_2, ..., $h_z >$. The number of extensions of the partial mapping $< h_1$, h_2, ..., $h_z >$ to the partial mapping $< h_1$, h_2, ..., $h_{z+1} >$ depends only on values $|\ \{1, 2, ..., n\} - CD^1 - CD^2 - ... - CD^z\}\ |$ and m. That holds for each $z : 1 \leq z \leq m-1$. Therefore, the set $\{h\}_{\mathcal{T}_i}$ is symmetric for $r = 0$.

Let now $r \neq 0$.

We conclude basing on Proposition 3.6.1 that for each model $S_U^z =<< \mathcal{T}_i >, i \in I_z;\ W$ given in (3.10), where $r = 0$; $\{f\}_{\mathcal{T}_i} >$ there is a congruent model $S_U^z =<< \mathcal{T}_i >, i \in I_z;\ W$ given in (3.10), where $r > 0$; $\{f\}_{\mathcal{T}_i} >$. Hence, symmetry of the model S_U with $r = 0$ implies symmetry of a model with $r \neq 0$, Therefore, the given model S_U possess symmetric set $\{h\}_{\mathcal{T}_i}$ for the enabled values of parameters n, m and r. That finishes the proof.

□

3.7.3 Table DBM

The structure of the sets of choice functions $\{h\}_{\mathcal{T}_i}$ is represented by the tables DBM[$n \times n$].

Let $< \mathcal{T}_i >, (k_1+k_2+...+k_{z-1}+1) \leq i \leq (k_1+k_2+...+k_z),\ 1 \leq z \leq m$, be a subfamily of the family $< \mathcal{T}_i >,\ 1 \leq i \leq n$, while f_{z-1} denotes a partial mapping for this subfamily. Then, each choice function $h \in \{h\}_{\mathcal{T}_i}$ can be seen as concatenation of the partial mappings $< f_1\ ,\ f_2\ ,\ ...,\ f_{z-1}\ >$.

Each codomain of f_{z-1} is denoted by CD^{z-1}. Since $f_z : I_z \longrightarrow \{1, 2, ..., n\} - CD^1 - CD^2 - ... - CD^{z-1}\}$, so for the fixed $< f_1 , f_2 , ..., f_{z-1} >$ the structure of the set of all the choice functions $\{f_z\}$ is represented by the table DCM$[u_z \times k_z]$, where $u = | \{1, 2, ..., n\} - CD^1 - CD^2 - ... - CD^{z-1}\} | - k_z + 1$. One observe it by noting that f_z is the increasing choice function for the subfamily $< \mathcal{T}_i >$, $(k_1 + k_2 + ... + k_{z-1} + 1) \leq i \leq (k_1 + k_2 + ... + k_z)$. Since each partial mapping $< f_1 , f_2 , ..., f_z >$ has to be extended to the full choice function $< f_1 , f_2 , ..., f_m >$ of the indexed family $< \mathcal{T}_i >$, $1 \leq i \leq n$, so instead of table DCM$[u_z \times k_z]$ we have to have a table DBM$^z[u_z \times k_z]$ that represents the numbers of extensions of each partial mapping $< f_1 , f_2 , ..., f_{z-1} , f_z(k_1+k_2+ ...+k_{z-1}+1), f_z(k_1+ k_2 + ... + k_{z-1}+2 >, ... f_z(i) >$ to the choice functions $< h(1), h(2), ..., h(n) >$. That number is $\sum_{l=1}^{u_{z+1}} \text{DBM}^{z+1}[l, k_1+k_2+...+k_{z+1}+1]$ times greater than the entry DCM$[x_i \times i]$. Hence, in order to get the table DBM$^z[u_z \times k_z]$ we have to multiply the table DCM$[u_z \times k_z]$ by $\sum_{l=1}^{u_{z+1}} \text{DBM}^{z+1}[l, k_1 + k_2 + ... + k_{z+1} + 1]$.

Basing on this observation, we evaluate the table DBM starting from DBM^m down to 1. The table DBM$[(n-k_1+1)\times n] = |$ DBM$^1[(n-k_1+1)\times k_1]$, DBM$^2[(n-k_1-k_2+1)\times k_2]$, ..., DBM$^m[(n-k_1-k_2-...-k_m+1)\times k_m]$ $|$. Observe, that table DBM$^m[(n - k_1 - k_2 - ... - k_m + 1) \times k_m] |=$ DCM $[(n - k_1 - k_2 - ... - k_m + 1) \times k_m] |$. If a subsets represented by tables DDCM or W(1.5)DCM or any other table are concerned, then the general procedure for evaluation of the table DBM requires only replacement of the table DCM by the proper one.

We have the algorithm TABDBM for making the table DBM$[n \times n]$.

Algorithm TABDBM(TABle D for Bijections piecewise Monotonic)
Input: the model $S_U = (2.39)$, n, m, k_1, k_2, ..., k_m
Output: the table DBM$^z[n \times k_z]$[2]
Method: The table D$^z[u_z \times k_z]$ is the table DCM$^z[u_z \times k_z]$ or the table DDCM$^z[u_z \times k_z]$ or the table W(1.5)DCM$^z[u_z \times k_z]$ or any other table for increasing choice functions.
1. Make the table D$^z[n \times k_z]$ for each set of indices I_z, $1 \leq z \leq m$, .
2. DBM$^m[u_m \times k_m] \longleftarrow$D$^m[u_m \times k_m]$
3. **for** $z \longleftarrow m - 1$ **downto** 1 **do**
3.1. $d_{z+1} \longleftarrow \sum_{l=1}^{u_z} \text{DBM}^z[l, 1]$
3.2. DBM$^z[u_z \times k_z] \longleftarrow d_{z+1}$.D$^z[u_z \times k_z]$
4. DBM$[n \times n] \longleftarrow <$ DBM$^1[u_1 \times k_1]$, DBM$^2[u_1 \times k_2]$, ..., DBM$^m[u_m \times k_m] >$
5. **return** DBM$[n \times n]$

---

[2] The size $n \times n$ is maximal for the table DBM. We get it if $m = n$, so $k_1 = k_2 = .. = k_n = 1$.

The table DBM can be computed in time $O(n^2)$. The size of table DBM can be reduced at least to DBM$[u_1 \times n]$ for non−deformed families $< \mathcal{T}_i >$, $1 \leq i \leq n$, where $u_1 = n - k_1 + 1$. If we use a deformed family $< \mathcal{T}_i' >$, $1 \leq i \leq n$, instead, then the size of DBM can be even more reduced. Therefore, the table DBM can be computed in time $O(u_1.n)$.

Example 3.7.5 *The set* $\{1, 2, ..., 10\}$ *is to be divided into* 4 *blocks whose cardinals are as follows* $|B_1| = 4$, $|B_2| = 2$, $|B_3| = 3$, $|B_4| = 1$. *Compute the table DBM for representing the full set of the corresponding decompositions.*

The tables D^z are the corresponding tables DCM$^z[u_z \times n]$. Since the table D^1 has 7 non 0−rows, therefore the size of the table DBM can be reduced to 7×4.

$$D^1[7 \times 4] = \begin{vmatrix} 84 & 28 & 7 & 1 \\ 56 & 21 & 6 & 1 \\ 35 & 15 & 5 & 1 \\ 20 & 10 & 4 & 1 \\ 10 & 6 & 3 & 1 \\ 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{vmatrix}, \quad D^2[7 \times 2] = \begin{vmatrix} 5 & 1 \\ 4 & 1 \\ 3 & 1 \\ 2 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{vmatrix}, \quad D^3[7 \times 3] = \begin{vmatrix} 3 & 2 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix},$$

$$D^4[7 \times 1] = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

$d_3 = 3 + 1 = 4$, $d_2 = 20 + 16 + 12 + 8 + 4 = 60$

DBM$[7 \times 10]$ <=DBM$^1[7 \times 1]$, DBM$^2[7 \times 2]$, DBM$^3[7 \times 3]$, DBM$^4[7 \times 1]$ =

$$= \begin{vmatrix} 4880 & 1680 & 420 & 60 & 20 & 4 & 3 & 2 & 1 & 1 \\ 3360 & 1260 & 360 & 60 & 16 & 4 & 1 & 1 & 1 & 0 \\ 2100 & 900 & 300 & 60 & 12 & 4 & 0 & 0 & 0 & 0 \\ 1200 & 600 & 240 & 60 & 8 & 4 & 0 & 0 & 0 & 0 \\ 600 & 360 & 180 & 60 & 4 & 4 & 0 & 0 & 0 & 0 \\ 240 & 180 & 120 & 60 & 0 & 0 & 0 & 0 & 0 & 0 \\ 60 & 60 & 60 & 60 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

□

3.7.4 Table DPM

Let $K \subset I$ that $K = \{1, k_1 + 1, k_1 + k_2 + 1, ..., k_1 + k_2 + ... + k_{m-1} + 1\}$.

Given is the model $S_U = << \mathcal{N}_i >, 1 \leq i \leq n$; W given in 2.36; $\{h\}_{\mathcal{N}_i} >$ as specified in 2.44, i.e., $\mathcal{N}_i = \mathcal{T}_i$ for $i \notin K$, and $\mathcal{N}_k = \min(\mathcal{U}_k)$ for $k \in K$, where $\mathcal{U}_k = \{1, 2, ..., n\} - CD^1 - CD^2 - ... - CD^{k-1}$.

The set $\{h\}_{\mathcal{N}_i}$ is the set of bijections piecewise non-decreasing except we have assertion that the choice partial mappings for the subfamily $< \mathcal{N}_k >$, $k \in K$ are the increasing choice functions. The table D that represents the structure of $\{h\}_{\mathcal{N}_i}$ is a case of the table DBM. Because the considered model represents a class of set partitions, we denote this table with separate symbol DPM. Since $\mathcal{N}_k = \min(\mathcal{U}_k)$ for $k \in K$, so each $k-$th column contains only one entry D$[1, k]$ different from zero, while the other entries D$[j, k] = 0$, $j \neq 1$. The following algorithm TABDPM produces the table DPM.

Algorithm TABDPM(TABle D for representing Partitions by piecewise Monotonic choice functions)

Input: the model S_U given in (2.39), n, m, k_1, k_2, ..., k_m

Output: the table DPM$^z[n \times k_z]$

Method: We use the tables DDCM$^z[u_z \times k_z]$ for each partial mapping h_z.

1. Make the table DDCM$^z[n \times k_z]$ for each set of indices I_z, $1 \leq z \leq m$,

2. DPM$^m[u_m \times k_m] \longleftarrow$DDCM$^m[u_m \times k_m]$

3. **for** $z \longleftarrow m - 1$ **downto** 1 **do**

3.1. DPM$^z[u_z \times k_z] \longleftarrow$ DPM$^{z+1}[1, 1]$.DDCM$^z[u_z \times k_z]$

4. DPM$[n \times n] \longleftarrow <$ DPM$^1[u_1 \times k_1]$, DPM$^2[u_1 \times k_2]$, ..., DPM$^m[u_m \times k_m] >$

5. **return** DPM$[n \times n]$

Since we do not need to evaluate the sums of entries $d_{z+1} = \sum_{l=1}^{u_z}DBM^z[l, 1]$, so time complexity is slightly lower than for the algorithm TABDBM.

Example 3.7.6 *Given are* $n = 10$, $k_1 = 4$, $k_2 = 2$, $k_3 = 3$, $k_4 = 1$. *Make the table DPM that represents the corresponding set of partitions.*

We have the following tables DDCM$^z[u_z \times k_z]$.

$$\text{DDCM}^1[7 \times 4] = \begin{vmatrix} 84 & 28 & 7 & 1 \\ 0 & 21 & 6 & 1 \\ 0 & 15 & 5 & 1 \\ 0 & 10 & 4 & 1 \\ 0 & 6 & 3 & 1 \\ 0 & 3 & 2 & 1 \\ 0 & 1 & 1 & 1 \end{vmatrix}, \quad \text{DDCM}^2[7 \times 2] = \begin{vmatrix} 5 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{vmatrix},$$

$$\mathrm{DDCM}^3[7\times 3]=\begin{vmatrix} 3 & 2 & 1\\ 0 & 1 & 1\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0 \end{vmatrix},\ \mathrm{DDCM}^4[7\times 1]=\begin{vmatrix} 1\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0 \end{vmatrix},$$

$d_4 = 1$, $d_3 = 3$, $d_2 = 5$

The maximal size has the table $\mathrm{DDCM}^1[7\times 4]$, so the size of the table DBM is $[7\times 10]$.

Then,

$$\mathrm{DPM}^4[7\times 1]=1.\mathrm{DDCM}^4[7\times 1]=\begin{vmatrix} 1\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0 \end{vmatrix},\mathrm{DBM}^3[7\times 3]=1.\mathrm{DDCM}^3[7\times 3]=\begin{vmatrix} 3 & 2 & 1\\ 0 & 1 & 1\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0 \end{vmatrix},$$

$$\mathrm{DPM}^2[7\times 2]=3.\mathrm{DDCM}^2[7\times 2]=3.\begin{vmatrix} 5 & 1\\ 0 & 1\\ 0 & 1\\ 0 & 1\\ 0 & 1\\ 0 & 0\\ 0 & 0 \end{vmatrix}=\begin{vmatrix} 15 & 3\\ 0 & 3\\ 0 & 3\\ 0 & 3\\ 0 & 3\\ 0 & 0\\ 0 & 0 \end{vmatrix},$$

$$\mathrm{DPM}^1[7\times 4]=15.\mathrm{DDCM}^1[7\times 2]=$$

$$=15.\begin{vmatrix} 84 & 28 & 7 & 1\\ 0 & 21 & 6 & 1\\ 0 & 15 & 5 & 1\\ 0 & 10 & 4 & 1\\ 0 & 6 & 3 & 1\\ 0 & 3 & 2 & 1\\ 0 & 1 & 1 & 1 \end{vmatrix}=\begin{vmatrix} 1260 & 420 & 105 & 15\\ 0 & 315 & 90 & 15\\ 0 & 225 & 75 & 15\\ 0 & 150 & 60 & 15\\ 0 & 90 & 45 & 15\\ 0 & 45 & 30 & 15\\ 0 & 15 & 15 & 15 \end{vmatrix}$$

$$\mathrm{DPM}[7\times 10]=<\mathrm{DPM}^1[7\times 1],\mathrm{DPM}^2[7\times 2],\mathrm{DPM}^3[7\times 3],\mathrm{DPM}^4[7\times 1]>=$$

$$=\begin{vmatrix} 1260 & 420 & 105 & 15 & 15 & 3 & 3 & 2 & 1 & 1\\ 0 & 315 & 90 & 15 & 0 & 3 & 0 & 1 & 1 & 0\\ 0 & 225 & 75 & 15 & 0 & 3 & 0 & 0 & 0 & 0\\ 0 & 150 & 60 & 15 & 0 & 3 & 0 & 0 & 0 & 0\\ 0 & 90 & 45 & 15 & 0 & 3 & 0 & 0 & 0 & 0\\ 0 & 45 & 30 & 15 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 15 & 15 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

□

3.8 Conclusions

We have shown that the structure of the sets of choice functions can be specified by the tables D. The methodology developed here makes new contribution to understanding the structural numbers and especially the Stirling's numbers. We have shown that the class of sets of combinatorial objects that can be represented by Stirling's like tables is much wider than it was classically investigated. We have shown relations between the tables D of distinct classes of the representation models. The concept of congruence of models developed on the basis of the canonical form of choice functions has valid applications as to development of the representations of a given set of combinatorial objects by distinct classes of unranking models. The most important applications of the tables D will be studied in next chapters for development of ranking representation methodology.

4
Ranking representation models

4.1 The rank and rank range

Given is unranking representation model $S_U =<< \mathcal{G}_i >, i \in I, \mathcal{G}_i \subseteq \mathcal{U}; W; \{h\}_{\mathcal{G}_i} >$. The sets $I \subseteq \{1, 2, ..., m\}$ and $\mathcal{U}$ are ordered increasingly with $<^I$ and $<^{\mathcal{U}}$, respectively. The pair of orders $(<^{\mathcal{U}}, <^I)$ generates the lexical order of the codomains of the choice functions. Since each choice function is uniquely assigned to its codomain, so the pair of orders $(<^{\mathcal{U}}, <^I)$ generates the lexical order of the choice functions $\{h\}_{\mathcal{G}_i}$ as well. The lexical order of the choice functions is denoted by $<^h$ or simply by $<$. Replacing the increasing order $<^{\mathcal{U}}$ by the decreasing order $>^{\mathcal{U}}$, we generate the anti–lexical ordering the choice functions $\{h\}_{\mathcal{G}_i}$ that is denoted by '$>^h$' or simply by '$>$'. The orders '$<^h$ 'or '$>^h$' or other linear orders determine the **rank** of each choice function $R(h)$, i.e., the number corresponding to the position of a given choice function $h \in \{h\}_{\mathcal{G}_i}$ in the string of all the choice functions $\{h\}_{\mathcal{G}_i}$ ordered, respectively. For the following part of this text we use the **lexical rank** according to the lexical order '$<^h$', however, the **anti–lexical rank** '$>^h$' could also be used.

We are concerned now with the pair $M =< S_U, '<^h'>$ .With denotation '$R(\{h\})$ in $R(h)$' or if it does not make ambiguity with $R(\{h\})$, we mean the set of all the ranks of the set $\{h\}_{\mathcal{G}_i}$. We call $R(\{h\})$ the **rank range** in $R(h)$ or simply the rank range $R(\{h\})$. Let $\{h'\}$ be a subset of $\{h\}_{\mathcal{G}_i}$.

Definition 4.1.1 *The rank range of a subset $\{h'\} \subset \{h\}_{\mathcal{G}_i}$ in $R(h)$, denoted by '$R(\{h'\})$ in $R(h)$', is the set of the ranks of all the choice functions $h' \in \{h'\}$ in $R(h)$.*

Definition 4.1.2 *We say that a given rank range* $R(\{h'\})$ *is continuous in* $R(h)$, *if* $R(h') \in [R(h'_{\min}), R(h'_{\max})]$, *where* $R(h'_{\min})$ *denotes the minimum rank of a choice function* $h' \in \{h'\}$, *while* $R(h'_{\max})$ *denotes the maximal rank of a choice function* $h' \in \{h'\}$.

Observe, the continuity of the rank range '$R(\{h'\})$ in $R(h)$' is a relative notion dependent on the model S_U. If the model $S'_U =<< \mathcal{G}'_i >, i \in I, \mathcal{G}'_i \subseteq \mathcal{U}; W; \{h'\}_{\mathcal{G}'_i} >$ were chosen that $\{h'\} = \{h'\}_{\mathcal{G}'_i}$, then the rank range '$R(\{h'\})$ in $R(h')$' would equal $\{1, 2, ..., \mid \{h'\} \mid\}$. So, the rank range '$R(\{h'\})$ in $R(h')$' would be continuous, independently from discontinuity of the given rank range '$R(\{h'\})$ in $R(h)$'. For a more general case a continuous rank range can be specified with $R(\{h'\}) = (R(h'_{\min}), e)$, where e is the cardinal $\mid \{h'\} \mid$.

If a given subset $\{h'\}$ is not continuous in $R(h)$, then it can always be split into a number of disjoint subsets $\{h'_1\}, \{h'_2\}, ..., \{h'_z\}$ that the corresponding rank ranges $R(\{h'_1\}), R(\{h'_2\}), ..., R(\{h'_z\})$ are continuous in $R(h)$. Then, the rank range $R(\{h'\})$ can be specified with $R(\{h'\}) =< (R(h'_{\min})_1, e_1), (R(h'_{\min})_2, e_2), ..., (R(h'_{\min})_z, e_z) >$. The continuous subrange $(R(h'_{\min})_p, e_p)$ we denote also with $R(\{h'_p\}), 1 \leq p \leq z$.

Definition 4.1.3 *The rank range* $R(\{h'\})$ *is* $e-$*continuous in* $R(h)$ *if* $R(\{h'\}) = < (R(h'_{\min})_1, e_1), (R(h'_{\min})_2, e_2), ..., (R(h'_{\min})_z, e_z) >$ *that* $e_1 = e_2 = ..., = e_z = e$.

Definition 4.1.4 *The rank range* $R(\{h'\})$ *is* $z-$*tuple in* $R(h)$ *if* $R(\{h'\}) = < (R(h'_{\min})_1, e_1), (R(h'_{\min})_2, e_2), ..., (R(h'_{\min})_z, e_z) >$ *and if each subrange* $R(\{h'_p\}), 1 \leq p \leq z$, *is isolated, i.e.,* $R(h'_{\min})_p > R(h'_{\max})_{p-1} + 1$.

If the rank range $R(\{h'\})$ is $e-$continuous in $R(h)$, then

$$R(\{h'\}) =< (R(h'_{\min})_1, e), (R(h'_{\min})_2, e), ..., (R(h'_{\min})_z, e) > . \qquad (4.1)$$

Given is the following hierarchical unranking representation model

$$\begin{cases} J = \{1, 2, ..., t\}, \ t < m, \\ << \mathcal{G}_j >, \ j \in J; W_j; \ \{f\}_{\mathcal{G}_j} >, \qquad \text{see (1.23).} \\ << \mathcal{G}_i >, 1 \leq i \leq m; W; h \in \{h\}_{\mathcal{G}_i} >, \end{cases}$$

The requirement W_j is a restriction of the requirement W, while the indexed family $< \mathcal{G}_j >, \ j \in J;$ is a subfamily of the family $< \mathcal{G}_i >, 1 \leq i \leq m$, that $\mathcal{G}_i = \mathcal{G}_j$ for every $j = i$. Then, '$<^f$' is the lexical order of the partial mappings f that is generated by the pair of orders $(<^J, <^{\mathcal{U}})$, i.e., '$<^f$' is a restriction of '$<^h$'. With denotation $R(f)$ we mean the rank of f according to $<^f$, while the rank range of the set $\{f\}_{\mathcal{G}_j}$ is denoted with '$R(\{f\})$ in $R(f)$'.

Let $\{f'\} \subset \{f\}_{\mathcal{G}_j}$, while $\{h'\}$ being the set of all the extensions of $\{f'\}_{\mathcal{G}_j}$. With denotation h'_1 we mean an extension of f_1, while h'_2 being an extension of f_2. Since all the partial mappings $\{f\}_{\mathcal{G}_j}$ are the choice functions of one indexed subfamily $< \mathcal{G}_j >, i \in J$ of the family $< \mathcal{G}_i >, i \in I$, so for any

pair (f_1, f_2) the sets of their corresponding extensions $\{h'_1\}$ and $\{h'_2\}$ are disjoint.

Lemma 4.1.1 *For the hierarchical system defined in (1.23), we have* $R(f_1) <^f R(f_2)$ *implies* $R(h_1) <^h R(h_2)$.

Proof. The choice functions f_1 and f_2 are prefixes of the choice functions h_1 and h_2. For the lexical order $<^h$ of the choice functions the greater rank of the fixed length prefix implies the greater rank of the corresponding choice function h.

□

We observe that if the assumptions given in (1.23) do not hold, then "$R(f_1) <^f R(f_2)$ does not imply $R(h_1) <^h R(h_2)$", in general.

Lemma 4.1.2 *For the hierarchical system given in (1.23) the rank range* $R(\{h'\})$ *is continuous in* $R(h)$ *if and only if the rank range* $R(\{f'\})$ *is continuous in* $R(f)$.

Proof. We are concerned now with the sufficient condition.

(i)

Suppose, $\mid \{f'\} \mid = 1$. Then, the set of choice functions $\{h'\}$ is a subset of $\{h\}$ possessing the common prefix f'. Therefore, the rank range $R(\{h'\})$ is continuous in $R(h)$.

(ii)

Suppose, now $\mid \{f'\} \mid > 1$. Then, for each $f' \in \{f'\}$ the set of the corresponding extensions possess the continuous rank subrange $R(\{h'\})$ in $R(h)$.

Since $R(\{f'\})$ is continuous in $R(f)$, so for every $f'_1 \in \{f'\}$ there is $f'_2 \in \{f'\}$ that $R(f_2) = R(f_1) + 1$.

With denotation $h'_{1\,\max}$, we mean the choice function being the extension of f'_1 that $R(h'_{1\,\max})$ is a maximum in $R(h)$, while $h'_{2\,\min}$ is an extension of f'_2 that $R(h'_{2\,\min})$ is a minimum in $R(h)$. Then, we have $R(h'_{2\,\min}) = R(h'_{1\,\max}) + 1$.

Since $R(f'_2) = R(f'_1) + 1$ holds for every $f' \in \{f'\}$ but $f'_{\max}$, so $R(\{h'\})$ is continuous in $R(h)$, where $f'_{\max} \in \{f'\}$ and $R(\ f'_{\max})$ is maximum. So, the sufficient condition is proved.

We are concerned now with the necessary condition.

If the rank range $R(\{f'\})$ is not continuous in $R(f)$, then there exists a pair $(\ f'_1, f'_2)$ of the partial mappings belonging to $\{f'\}$ that $R(\ f'_1) < R(f'_2)$ and for any other $f' \in \{f'\}$ we have $(R(f') < R(\ f'_1)$ and $R(f') < R(\ f'_2))$ or $(R(f') > R(\ f'_2)$ and $R(f') > R(\ f'_1))$. Therefore, $R(f'_2) > R(f'_1) + 1$. Hence, $R(h'_{2\,\min}) > R(h'_{1\,\max}) + 1$. Then, the rank range $R(\{h'\})$ is not continuous in $R(h)$. So, discontinuity of the rank range $R(\{f'\})$ in $R(f)$ implies discontinuity of the rank range $R(\{h'\})$ in $R(h)$. That finishes the proof of necessary condition.

□

Given are indexed families $< \mathcal{G}'_i >$, $i \in I$ and $< \mathcal{G}_i >$, $i \in I$.

Definition 4.1.5 *The indexed set $\mathcal{G}'_i$, is a compact subset of the set $\mathcal{G}_i$ if $\mathcal{G}'_i = \{x'_{\min}(\mathcal{G}_{i_1}) \div x'_{\max}(\mathcal{G}_{i_1})\}$ that $x'_{\min} \geq x_{\min}$ and $x'_{\max} \leq x_{\max}$, $i \in I$, i.e., the indexed set $\mathcal{G}'_{i_1}$ contains a number of consecutive elements of $\mathcal{G}_i$.*

If every set $\mathcal{G}'_i$ is a compact subset of $\mathcal{G}_i$, then the indexed family $< \mathcal{G}'_i >$, $i \in I$ is a compact subfamily of the family $< \mathcal{G}_i >$, $i \in I$. We say that the indexed family $< \mathcal{G}'_i >$, $i \in I$ is a compact subfamily of the family $< \mathcal{G}_i >$, $i \in I$ for a single $j \in I$ if $x'_{\min} > x_{\min}$ or $x'_{\max} < x_{\max}$, while for every $i \neq j$ we have $\mathcal{G}'_i = \mathcal{G}_i$.

Lemma 4.1.3 *If $< \mathcal{G}'_i >$, $i \in I$ is a compact subfamily of the family $< \mathcal{G}_i >$, $i \in I$ for a single index j, then the rank range $R(\{h'\})$ is continuous in $R(h)$.*

Proof. The model $<< \mathcal{G}'_i >, i \in I\ ; W; \{h\}_{\mathcal{G}_i} >$ is equivalent to the hierarchical model given in (1.23) that $j = t$. Then, the rank range $R(\{f'\}_{\mathcal{G}_j})$ is continuous in $R(f)$.

Therefore, basing on the previous lemma we have the rank range $R(\{h'\})$ is continuous in $R(h)$.

□

4.2 The ranking model

If for a given unranking representation model S_U exists the table D, then we can develop very special methods concerning evaluation and the generation of the choice functions and their ranks. That happens since the table D contains important information concerning the rank and the rank range. In order to emphasize that our concern is related to the models S_U for which the table D exist, we investigate a ranking representation model as follows:

Definition 4.2.1 *The ranking representation model S_R for symmetric sets $\{h\}_{\mathcal{G}_i}$ is the following object $< S_U$; '$<^h$'; table D, $>$.*

The order '$<^h$' determines the rank $R(h)$, so the rank $R(h)$ and the rank range $R(\{h'\})$ are components of the ranking representation model, however they are not the independent components, therefore we do not specify them in the above definition.

Given is the ranking representation model $S_R = < S_U$; '$<^h$'; table D, $>$, where $S_U = << \mathcal{G}'_i >, i \in I\ ; W; \{h\}_{\mathcal{G}_i} >$. Let $\{h'\}$ be a subset of choice functions of the set $\{h\}_{\mathcal{G}_i}$ such that $\{h'\} = \{h'\}_{\mathcal{G}'_i}$, where $\{h'\}_{\mathcal{G}'_i}$

is the set of choice functions for the model $S'_U = << \mathcal{G}'_i >, i \in I ; W; \{h'\}_{\mathcal{G}'_i} >$. The indexed family $< \mathcal{G}'_i >, i \in I$, is a subfamily of the indexed family $< \mathcal{G}_i >, i \in I$ such that for the set of indexes $J = \{i_1, i_2, ..., i_q\}$, $J \subset I$ we have $\mathcal{G}'_{i_p} \subset \mathcal{G}_{i_p}$, $i_p \in J$ and $\mathcal{G}'_{i_q}$ is a compact subset of $\mathcal{G}_{i_q}$, i.e., $x'(\mathcal{G}_{i_q}) \in \mathcal{G}'_{i_q}$, where $x'_{\min} \leq x' \leq x'_{\max}$, while $x(\mathcal{G}_{i_q}) \in \mathcal{G}_{i_q}$ for $x_{\min} \leq x \leq x_{\max}$; $x'_{\min} > x_{\min}$ or $x'_{\max} < x_{\max}$. With denotation $\sum_y \mathrm{D}[y, i_q]$ we mean the sum of entries of the table D that $y(\mathcal{G}_{i_q}) \in \mathcal{G}'_{i_q}$ and the value $h'(i_q) = y(\mathcal{G}'_{i_q})$ is enabled for the choice functions possessing a fixed prefix $h'(1), h'(2), ..., h'(i_q - 1)$. Since the enabled values $h'(i_q) = y(\mathcal{G}'_{i_q})$ can differ for each valid prefix $h'(1), h'(2), ..., h'(i_q - 1)$, so we denote by $\sum_{y_k} \mathrm{D}[y_k, i_q]$ the sum of the enabled entries for the $k-th$ valid prefix. Let S''_U be the unranking representation model for all the prefixes $h'(1), h'(2), ..., h'(i_q - 1)$, i.e., $S''_U = << \mathcal{G}''_i >, i \in I' ; W; \{h''\}_{\mathcal{G}''_i} >$, where $I' = \{1, 2, .., i_{q-1}\}$ and $\mathcal{G}''_i = \mathcal{G}'_i$ for each $i \in J$ while $\mathcal{G}''_i = \mathcal{G}_i$ for each $i \in I'$ and $i \notin J$.

Proposition 4.2.1 *If the above given assumptions hold, then the rank range* $R(\{h'\}) = < (R(h'_{\min})_1, e_1), (R(h'_{\min})_2, e_2), ..., (R(h'_{\min})_z, e_z) >$, *where* $e_k = \sum_{y_k} D[y_k, i_q]$ *and* $z = | \{h''\}_{\mathcal{G}''_i} |$.

Proof. We observe that for the $k-th$ prefix $h'(1), h'(2), ..., h'(i_q - 1)$, we have a set of choice functions $\{h'\}_k$ that take values $h'(i_q) \in \mathcal{G}_{i_q}$ and the corresponding rank range is continuous in $R(h)$. Therefore, for the set of choice functions $\{h'\}_k$, we have the corresponding component $(R(h'_{\min})_k, e_k)$ of the rank range. The set of choice functions $\{h'\}$ can be given by the form $\{h'\} = \{h'\}_1 \cup \{h'\}_2 \cup ... \cup \{h'\}z$, hence, $e_k = | \{h'\}_k |$. Basing on the definition of the table D we can write $e_k = \sum_{y_k}(\mathrm{D}(y_k, i_q))$, where $y_k(\mathcal{G}_{i_q}) \in \mathcal{G}'_{i_q}$ and the value $h'(i_q) = y_k(\mathcal{G}'_{i_q})$ is enabled for the given prefix $h'(1), h'(2), ..., h'(i_q - 1)$.

We observe that the choice functions for each prefix $h'(1), h'(2), ..., h'(i_q - 1)$ are represented by isolated continuous rank range $(R(h'_{\min})_k, e_k)$. Hence, the number z equals the number of all the prefixes $h'(1), h'(2), ..., h'(i_q - 1)$. The number of all the prefixes $h'(1), h'(2), ..., h'(i_q - 1)$ equals the number of choice functions $\{h''\}_{\mathcal{G}''_i}$ for the model S''_U. That finishes the proof.

□

If the set of choice functions $\{h''\}_{\mathcal{G}''_i}$ is symmetric for the model S''_U, then we can create the ranking representation model $S''_R = < S''_U, <^h$, table D"$> . Then, $z = \sum_x \mathrm{D}[x, 1]$, i.e., z equals to the sum of all the entries of the first column. Depending on the type of choice functions modeled the general formula $e_k = \sum_{y_k} \mathrm{D}[y_k, i_q]$ takes a form more suitable for evaluation. For instance, if the increasing choice functions are concerned,

then we can use the form $\sum_{x=x'_{\min}}^{x'_{\max}} D[x, i_q]$, where $x'_{\min}(\mathcal{G}'_{i_q})$ is the minimum value for $h'(i_q)$ that can be taken for a prefix $h'(1), h'(2), ..., h'(i_q - 1)$.

Example 4.2.1 *Given is the model* $S_U =<< A_i$ *for* $n = 8 >$, $1 \leq i \leq 4$, ; W *given in* (1.1) ; $\{h\}_{\mathcal{G}_i} >$ *and the unranking model* $S'_U =<< \mathcal{P}_i >$, $1 \leq i \leq 4$, ; W *given in* (1.1) ; $\{h'\}_{\mathcal{P}_i} >$, *where* $\mathcal{P}_1 = \{1, 2, 3\}$, $\mathcal{P}_2 = \{2, 3, 4, 5\}$, $\mathcal{P}_3 = \{3, 4, 5, 6, 7\}$, $\mathcal{P}_i = \{4, 5, 6, 7, 8\}$. *The task is to specify the rank range* $R(\{h'\})$ *in* $R(h)$.

We observe that the indexed family $<\mathcal{P}_i >$, $1 \leq i \leq 4$, is a subfamily of the family $< \mathcal{A}_i >$, $1 \leq i \leq 4$, where $J = \{1, 2\}$. Then, the set $\{h'\}$ is symmetric. We have the model $S''_R = < S''_U, <^h$, table D"$>$ as follows: $S''_U =<<\mathcal{P}_i >, i = 1, W, \{h''\}_{\mathcal{P}_i} >$, table DCM"$= \begin{vmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{vmatrix}$. Then, $z =$DCM"$[1, 1]$ + DCM"$[2, 1]$ + DCM"$[3, 1] = 3$. For the model S_U we have the following DCM table DCM$= \begin{vmatrix} 35 & 15 & 5 & 1 \\ 20 & 10 & 4 & 1 \\ 10 & 6 & 3 & 1 \\ 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{vmatrix}$

Then, the concerned rank range is as follows

$R(\{h'\}) = < (R(h'_{\min})_1, e_1), (R(h'_{\min})_2, e_2), (R(h'_{\min})_3, e_3) >$. The ranks $R(h'_{\min})_1$, $R(h'_{\min})_2$, $R(h'_{\min})_3$ are the ranks of the choice functions $< 1, 2, 3, 4, 5 >, < 2, 3, 4, 5, 6 >, < 3, 4, 5, 6, 7 >$, correspondingly. We evaluate e_1, e_2, e_3 basing on the entries of the DCM table:

$e_1 = \sum_{x=x'_{\min}}^{x'_{\max}} D[x, 2] = \sum_{x=1}^{4} D[x, 2] =$ DCM$[1, 2]$ + DCM$[2, 2]$ + DCM$[3, 2]$ + DCM$[4, 2] = 15 + 10 + 6 + 3 = 34$.

$e_2 = \sum_{x=x'_{\min}}^{x'_{\max}} D[x, 2] = \sum_{x=2}^{4} D[x, 2] =$ DCM$[2, 2]$ + DCM$[3, 2]$ + DCM$[4, 2] = 10 + 6 + 3 = 19$.

$e_3 = \sum_{x=x'_{\min}}^{x'_{\max}} D[x, 2] = \sum_{x=3}^{4} D[x, 2] =$ DCM$[3, 2]$ + DCM$[4, 2] = 6 + 3 = 9$.

The ranks $R(h'_{\min})_1 = 1$, $R(h'_{\min})_2 = 36$, $R(h'_{\min})_3 = 57$. Then,

$R(\{h'\}_{\mathcal{G}_i}) = < (R(h'_{\min})_1, e_1), (R(h'_{\min})_2, e_2), (R(h'_{\min})_3, e_3) > =< (1, 34), (36, 19), (57, 9) >$

□

If $\mid \mathcal{G}'_{i_q} \mid = 1$, then the rank range $R(\{h'\})$ is $e-$continuous in $R(h)$, where $e = D[x_{i_q}, i_q]$.

The existential consequences of the above propositions for modelling are developed in the next section.

4.3 Ranking Theorem

We are concerned with two ranking models $S_R =< S_U; '<^h'$; table D > and $S'_R =< S'_U; '<^h'$; D' table> that a given set of choice functions $\{h'\} \subset \{h\}_{\mathcal{G}_i}$ and simultaneously $\{h'\} \subseteq \{h'\}_{\mathcal{G}'_i}$, where $\{h\}_{\mathcal{G}_i}$ is the component of S_U and $\{h'\}_{\mathcal{G}'_i}$ is the component of S'_U, while $\{h'\}$ is a subset of $\{h\}_{\mathcal{G}_i}$. We emphasize that the models S_U and S'_U can have different indexed families and/or the requirements W can be constructed in different ways.

Lemma 4.3.1 *If the set of choice functions $\{h'\}$ is symmetric, then there exists ranking model S'_R that the rank range $R(\{h'\})$ is continuous in $R(h')$.*

Proof. We know that for any given set $\{h'\}$ there is unranking model S'_U that $\{h'\} = \{h'\}_{\mathcal{G}'_i}$. Since the set $\{h'\}_{\mathcal{G}'_i}$ is symmetric, so there is a table D' that represents the structure of $\{h'\}_{\mathcal{G}'_i}$. Therefore, we can build the model S'_R that represents $\{h'\}_{\mathcal{G}'_i}$. Since $\{h'\} = \{h'\}_{\mathcal{G}'_i}$, so $R(\{h'\})$ is continuous in $R(h')$.

□

If the set $\{h'\}_{\mathcal{G}_i}$ is symmetric and the rank range $R(\{h'\})$ is $z-$tuple in $R(h)$, then we can always build a model S'_R that $R(\{h'\})$ is continuous in $R(h')$. In fact there is a game of models S''_R that $R(\{h'\})$ is $z''-$tuple in $R(h'')$ but $z'' < z$. Then, we have a hierarchy of models with respect to the value of z. That hierarchy is isomorphic with the hierarchy of indexed families $< \mathcal{G}_i^1 = \mathcal{G}_i >$, $1 \le i \le m$; $< \mathcal{G}_i^2 >$, $1 \le i \le m$; ...; $< \mathcal{G}_i^w = \mathcal{G}'_i >$, $1 \le i \le m$; that $\mathcal{G}_i^{r+1} \subseteq \mathcal{G}_i^r$; $1 \le r \le w$.

Assumptions:

(i) Given is a model $S_R =< S_U; '<^h'$; table D > and the subset of choice functions h' that $h' \subset h_{\mathcal{G}_i}$.

(ii) The rank range $R(\{h'\})$ in $R(h)$ is specified with $R(\{h'\}) =< (R(h'_{\min})_1, e_1), (R(h'_{\min})_2, e_2), .., (R(h'_{\min})_z, e_z) >$.

(iii) The k components $(R(h'_{\min})_p, e_p)$ represent non-symmetric subsets of choice functions.

We conclude basing on the above assumptions, if k components $(R(h'_{\min})_p, e_p)$ were removed from $R(\{h'\})$, then the obtained rank range $R(\{h'\}^{\Psi})$ in $R(h)$ would represent a symmetric subset $\{h'\}^{\Psi}$ of $\{h'\}$. Then, the following theorem holds.

Theorem 4.3.1 (Ranking Theorem) *If the above assumptions hold, then there is representation model S'_R that $\{h'\} = \{h'\}_{\mathcal{G}'_i}$ and the rank range $R(\{h'\})$ is $z'-$tuple in $R(h)$, where $z' \le 2.k + 1$.*

Proof. Let $(R(h'_{\min})_{p-1}, e_{p-1})$ and $(R(h'_{\min})_p, e_p)$ be two neighboring subranges, i.e., the rank range $(R(h'_{\min})_{p-1}, e_{p-1})$ predeceases the rank range $(R(h'_{\min})_p, e_p)$ following the order $<^h$. Suppose, $(R(h'_{\min})_{p-1}, e_{p-1})$ and $(R(h'_{\min})_p, e_p)$ do not contradict the symmetry of $\{h'\}$. Then, basing

on Lemma 4.3.1, we can modify the representation model that the neighboring subranges $(R(h'_{\min})_{p-1}, e_{p-1})$ and $(R(h'_{\min})_p, e_p)$ became replaced by one rank subrange $(R(h'_{\min})_{p-1}, e_{p-1} + e_p)$. Therefore, we can have a representation model that the symmetric and neighboring subranges concatenate.

Suppose, now the subset $\{h'\}$ is non–symmetric. Then, there is a corresponding symmetric subset $\{h\}^{\overline{\pm}}$ of $\{h\}_{\mathcal{C}}$ that $\{h'\} \subset \{h\}^{\overline{\pm}}$ and the cardinal $\mid \{h\}^{\overline{\pm}} \mid$ is minimum.

Since $\mid \{h\}^{\overline{\pm}} \mid$ is minimum, so each rank subrange $(R(h'_{\min})_p, e_p)$ corresponds to the rank subrange $(R(h'_{\min})^{\overline{\pm}}_p, e'_p)$ that $e'_p > e_p$ if $(R(h'_{\min})^{\overline{\pm}}_p, e_p)$ corresponds to a non–symmetric subset or $e'_p = e_p$ if $(R(h'_{\min})^{\overline{\pm}}_p, e_p)$ corresponds to a symmetric subset.

Since $\{h\}^{\overline{\pm}}$ is symmetric, so basing on Lemma 4.3.1 there is a ranking representation model S'_R that the rank range $R(\{h\}^{\overline{\pm}})$ is continuous in $R(h')^{\overline{\pm}}$.

We observe that insertion of one subrange $(R(h'_{\min})_p, e_p)$ that makes the set $\{h'\}$ asymmetric into a string of subranges effects existence of at most two additional subranges that can not concatenate. One can note it observing that $e'_p > e_p$ but transformation of $R(h'_{\min})_p$ into $R(h'_{\min})^{\overline{\pm}}_p$ is not defined. Therefore, the total number z' of subranges that can not concatenate is not greater than $2.k + 1$. Hence, for each set of choice functions, we can built the representation model S'_R that the corresponding rank range $R(\{h'\})$ is at most $2.k + 1$–tuple in $R(h)$. That finishes the proof.

□

The above given theorem has fundamental meaning for construction of the ranking representation models that the number z is kept as small as possible. Keeping z as small as possible is the main request of modelling in order to preserve greatest efficiency of the generation.

Example 4.3.1 *The subset $\{h'\}$ is defined to be the set of bijections of the indexed family $< \mathcal{C}_i >$, $1 \leq i \leq 4$, $\mathcal{C}_i = \{1 \div 4\}$ that the requirement W_1 holds, where W_1 : $(h'(1) = 1$ or $h'(1) = 3)$ and $(h'(2) = 2$ or $h'(4) = 4)$ and if $h'(1) = 3$ and $h'(2) = 4$, then $h'(3) > h'(4)$.*

The model S_R is as follows:

$S_U = << G_i >, 1 \leq i \leq 4; W$ given in (1.19); $W_1; \{h\}_{\mathcal{C}_i} >; S_R = < S_U, <^h$, table D=DP> . The bold fonts specify the subset $\{h'\}$.

$R(h)$	h	$R(h)$	h	$R(h)$	h	$R(h)$	h
1	**1234**	7	2134	13	3124	19	4123
2	**1243**	8	2143	14	3142	20	4132
3	1324	9	2314	**15**	**3213**	21	4213
4	1342	10	2341	**16**	**3231**	22	4231
5	**1423**	11	2413	17	3412	23	4312
6	**1432**	12	2431	**18**	**4321**	24	4321

Then, the rank range $R(\{h'\})$ in $R(h)$ is as follows: $< (1, 2), (5, 2), (15, 2), (18, 1) >$, therefore it is 4−tuple. The set $\{h'\}$ is non−symmetric.

Consider the model S'_R as follows: $S'_{UN} = << \mathcal{G}'_i >$, $1 \leq i \leq 4$, where $\mathcal{G}'_1 = \{1, 3\}$, $\mathcal{G}'_2 = \{2, 4\}$, $\mathcal{G}'_3 = \mathcal{G}'_4 = \{1 \div 4\}$; W given in (1.19); W_1; $\{h'\}_{\mathcal{G}'_i} >$; $S_R = < S'_U, <^h$, table D'=DS, $> .$

table DS= $\begin{vmatrix} 4 & 0 & 1 & 1 \\ 0 & 2 & 1 & 1 \\ 4 & 0 & 1 & 1 \\ 0 & 2 & 1 & 1 \end{vmatrix}$

Then, the set $\{h\}^{\preceq}$ is specified in the following table

$R(h)^{\preceq}$	h	$R(h)^{\preceq}$	h
1	**1234**	**5**	**3214**
2	**1243**	**6**	**3241**
3	**1423**	7	3423
4	**1432**	**8**	**3432**

The rank range $R(\{h'\})$ in $R(h')$ is as follows: $< (1, 6), (8, 1) >$, therefore it is a pair (2−tuple).

□

4.4 Conclusion

We have shown how subsets of choice functions can be represented by rank. The rank of basic combinatorial objects was investigated in the classical theory but it was investigated only for full sets of combinations or permutations or partitions and so on. Consequently, the classical theory gives only one possibility of representing a set of combinatorial objects by rank. The given here approach is essentially different. We have in advance a set of choice functions to be represented by rank and we look for a model for which ranking representation of the given set would be the most suitable. Suitability of the representation is measured by compactness of the corresponding rank range. If more compact the rank range representing a given set of choice function, then the representation is better. The next chapters of this text justify the accepted criterion of optimization of the representation. We will show that if more compact rank range, then the representation is more suitable for the generation. Ranking Theorem says what we can expect as the best solution of our modelling problem since we specify individually how much compact the rank range can be for the best model. The next chapter brings also the methodology for making the suitable ranking representation.

5

Modeling sets of combinatorial objects

5.1 The goals of modelling

A given set of combinatorial objects can be represented in a number of ways using the presented general approach. We can ask what a model is the most suitable one. Unfortunately, a detailed response to this question is rather composed and we do not think that specification of a linear hierarchy of models concerning their suitability would be possible for a general case. For selection of the most suitable model, we have to take a number of circumstances into account. These circumstances can be examined only if a model is fully specified. So, we can rather select a most suitable representation among a number of proposals instead of saying in advance what type of model is the most suitable one. Nevertheless, certain general principles concerning suitability of the models can be specified in advance.

(*i*) Generally a model S_R is more flexible than the model S_U, since for the model S_R we can use methodology both for ranking and unranking generation.

(*ii*) The S_R models that the given set $\{h'\}$ possess the rank range $R(\{h'\})$ in $R(h)$ compact or $z - tuple$ with as small z as possible and with greatest possible parameters e_k are more suitable.

(*iii*) The models S_U reduced and possessing Q property are much more suitable than models that the Q property does not hold.

(*iv*) If simpler and easier for testing there are requirements W and W_1, then more suitable models.

(*v*) Compact indexed families are better than random ones.

(*vi*) If the representation contains a smaller number of the partial models S_U or S_R , then it is more suitable.

The specified criteria of suitability can be mutually contradictory for given instances of the general problem. Therefore, making a most suitable model is a matter of a compromise.

The process of searching for a suitable model can be organized as follows:

---

1. Make set(s) of choice functions $\{h\}$ or $\{h_1\}$, $\{h_2\}$, .., $\{h_q\}$ that each combinatorial object to be represented corresponds uniquely to one choice function $h \in \{h\}$ or to each possible collection of choice functions $< h_1, h_2,$ .., $h_q >$ that $h_p \in \{h_p\}$, $1 \leq p \leq q$.

2. Find an unranking representation for each $h_p \in \{h_p\}$. If it is possible, then use one S_U model else use as small as possible number of models S_U^1, S_U^2, ..., S_U^r that cover $\{h\}$ or $\{h_1\}$, $\{h_2\}$, .., $\{h_q\}$.

3. Examine suitability of each model S_U^1, S_U^2, ..., S_U^r as criteria use concepts of reducibility and satisfiability of the Q property, then try to reduce the requirements W and W_1 as much as possible.

4. If it is needed and if it is possible, then produce the corresponding tables D making the formal models S_R.

5. If the models obtained in step 3 and/or 4 are not suitable, then $\{h'_p\} \longleftarrow \{h_p\}$ and produce new set(s) $\{h\}$ or $\{h_1\}$, $\{h_2\}$, .., $\{h_q\}$ that $\{h'_p\} \subset \{h_p\}$. Go to step 2.

6. Specify the set(s) $\{h'_p\}$ by rank, if any.

The given scheme of the general methodology can be realized in many different ways especially each step can be performed using several different techniques. In the following sections of this chapter, we will specify these techniques in more detail.

5.2 Basic unranking modelling

In the course of the basic unranking modelling the first problem we have is specification of the requested collection of combinatorial objects by a set $\{h\}$ or by sets $\{h_1\}$, $\{h_2\}$, .., $\{h_k\}$ of functions. Formally the problem can be stated as follows. Given is a structure $S =< A_1, A_2, ..., A_u >$, where the sets A_1, A_2, ..., A_u represent the specified objects, i.e., numbers, vertices, edges, columns, rows, matrices and so on. Formally $A = < a(1), a(2), ..., a(n) >$, i.e., A is an indexed collection of elements. A combinatorial object to be represented is denoted by $S' = < A'_1, A'_2, ..., A'_u >$, where each component A' equals A or it is obtained from A in a specified manner. The specified manner means A' is a certain subset of A or A' can be obtained from A by permuting its elements in a specified way or by partitioning its elements and so on. The goal is to represent a collection of substructures

$< S' >=<< A'_1, A'_2, ..., A'_u >>$. The most obvious representation of the requested collection of subsets $<< S' >>$ is by assigning one function $h \in \{h\}$ into desired A'. Then, each choice function $h \in \{h\}$ represents uniquely a combinatorial object desired. Valid goal in searching for a most suitable representation is minimization of substitution instructions contained in the whole model, that reduces the generation process. We have three direct ways of assigning function $h =< h(1), h(2), ..., h(m) >$ into corresponding A':

$* \; A' =< a(h(1)), a(h(2)), ..., a(h(m)) >,$

$** \; A' =< a(1) \in B_{h(1)}, a(2) \in B_{h(2)}, ..., a(m) \in B_{h(m)} >,$

$*** \; A' =< a(h(1)) \in B_1, a(h(2)) \in B_2, ..., a(h(m)) \in B_m > .$

The assignment $*$ is usually used if A' is to be created by means of permutations or combinations or variations. The assignments $**$ or $***$ are used if A' is to be created by means of partitions or decompositions. Then, we usually make the representation as given in $*$ for each block of partition or decomposition. Apart of the direct assignment we can use an indirect assignment of h into A'. Then, we make another auxiliary one–to–one mapping η in order to assign each possible A' into h. The mapping η takes usually values $h(i)$ and $h(i-1)$ and determines an element $a(i)$ as the component of A'. We can have numerous different mappings η and numerous different determinations of structures but the most simple and the most common method is given for the unranking representations of number partitions and compositions, see formula (2.2), page 30. Using k sets $\{h_1\}, \{h_2\}, .., \{h_k\}$ we can transform k sets $A'_1, A'_2, ..., A'_k$ for getting $S' \in < S' >$. For certain structures a modification of a set $A_p, 1 \leq p \leq k$, makes also an automatic modification of some other sets. Then, the number of sets $\{h_1\}, \{h_2\}, .., \{h_k\}$ is smaller than the number of the modified sets A. For instance, if the structure S is a graph $< V, E >$, then selecting a subset of vertices V', we create simultaneously a subset of edges E' for the corresponding subgraph S'.

After specifying the set $\{h\}$ or the sets $\{h_1\}, \{h_2\}, .., \{h_k\}$, we make formal unranking representation model(s) S_U or $< S^1_U, S^2_U, ..., S^k_U >$, that $\{h\} = \{h\}_{\mathcal{G}_i}$ or $\{h_1\} = \{h\}_{\mathcal{G}^1_i}$ and $\{h_2\} = \{h\}_{\mathcal{G}^2_i}$ and , .., and $\{h_k\} = \{h\}_{\mathcal{G}^k_i}$, respectively. For a given set $\{h\}$, we can make several different S_U models. The reminder of this section demonstrates examples of the basic unranking modelling.

Example 5.2.1 *For the graph $G = (V, E)$ given in Figure 5.1 let us represent the group of authomorphism, i.e., all the graphs that are isomorphic with G.*

The group of authomorphism of the graph G corresponds to the set of choice bijections of the following indexed family $\mathcal{S}_1 = \{1,3,4,6\}$, $\mathcal{S}_2 = \{2,5\}$, $\mathcal{S}_3 = \{1,3,4,6\}$, $\mathcal{S}_4 = \{1,3,4,6\}$, $\mathcal{S}_5 = \{2,5\}$, $\mathcal{S}_6 = \{1,3,4,6\}$.

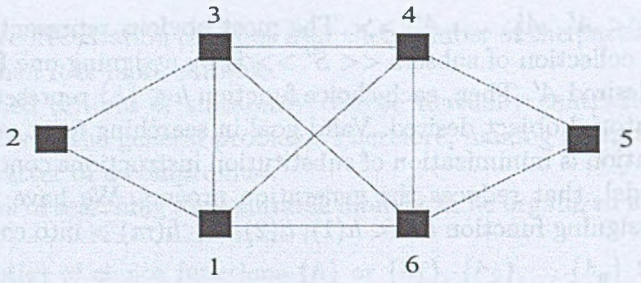

FIGURE 5.1.

For explanation of the method, we observe that the set of vertices V is partitioned into the blocks containing equivalent nodes. Then, the blocks are indexed with the numbers of nodes included in them getting the indexed sets. Ordering the indexed sets finishes production of the indexed family. We observe that such indexed families obtained are reduced and the Q property holds since the assumptions of Proposition 1.2.2 hold.

Specification of the whole model S_U is enough obvious.

□

It is known that for testing whether two given graphs are isomorphic it is enough to generate the group of authomorphism $A(G)$ for one graph and then to test whether there is a permutation belonging to $A(G)$ that represents the second graph. So, the set of choice functions $\{h\}_{S_i}$ corresponds to the maximal number of tests needed in order to state existence of isomorphism or its lack. In the similar way we can make the sets of tests for solving other isomorphic complete problems, i.e., for the problems that are polonomially transformed into the graph isomorphism [17], [22].

Example 5.2.2 *At a port there are n people and n boats enumerated. Taking consecutive boats a number of consecutive people has to get into the boat and the boat leaves the port. Then, a number of consecutive people has to get on the next boat. Procedure is to be continued until all the people leave the port. Each boat can take at most n and at least two people. The task is to represent all the possible distributions of the people on boats.*

Formally, we have the structure $S =< P, B >$, where $P =< p_1, p_2, ..., p_n >$ is the collection of all the people, while $B =< b_1, b_2, ..., b_m >$ is the collection of all the boats, $m = n/2$. Each substructure S' corresponding to a possible distribution of the people onto the boats is as follows $S' =< p_1, p_2, ..., p_{r_1} >, < p_{r_1+1}, p_{r_1+2}, ..., p_{r_1+r_2} >, ..., < p_{n-r_k+1}, p_{n-r_k+2}, ..., p_n >>$, where k is the number of the used boats, while r_i is a number of people on the $i-th$ boat; $1 \leq k \leq m$.

(i) The first possible representation.

Each S' corresponds to an assignment $\eta_k : i \longrightarrow r_i$, where $1 \leq i \leq k$; $2 \leq r_i \leq m$. For each k, we assign the set $\{h_k\}$ that each $h_k \in \{h_k\}$ corresponds to an assignment of the numbers of people into k boats used, so that $\eta_k(i) = h_k(i) - h_k(i-1)$. Therefore, all the possible distributions of the people into a number of boats used correspond to the collection of sets of choice functions $\{h_1\}$, $\{h_2\}$, .., $\{h_m\}$.

Now the goal is to built the models $S_U^k =<< \mathcal{G}_i^k >$, $1 \leq i \leq k$; W; $\{h\}_{\mathcal{G}_i^k} >$, $1 \leq k \leq m$, that $\{h_k\} = \{h\}_{\mathcal{G}_i^k}$. Using similar approach as given for the representation of compositions, we have $\mathcal{G}_i^k = \{2.i,\ 2.i+1,\ ...,\ n-2.(k-i)\}$, $1 \leq i \leq k$. The requirement W has to make assertion that $2 \leq \eta_k(i) \leq m$, so $W : h_k(i) > h_k(i-1) + 1$. So, we have the collection of the unranking models S_U^1, S_U^2, ..., S_U^m for representing $\{h_1\}$, $\{h_2\}$, .., $\{h_m\}$, correspondingly. Then, all the possible distributions of the people into possible numbers of boats used correspond to $\{h\} = \{h\}_{\mathcal{G}_i^1} \cup \{h\}_{\mathcal{G}_i^2} \cup ... \cup \{h\}_{\mathcal{G}_i^m}$. It is worthwhile to mention that evaluations on P are represented by the unranking models, while evaluations on B are represented by the number of models used.

(ii) The second possible representation.

Let $h \in \{h\}$ be a choice function of an indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$, that $h(i) \in \{h(i-1)+2,\ h(i-1)+3,\ ...,\ n\}$, $h(0) = 0$. Then, $\eta_i = h(i) - h(i-1)$ corresponds to a number of consecutively enumerated people that get onto the $i-th$ boat. Therefore, it must hold $2 \leq \eta_i \leq m$ or $\eta_i = 0$ and if $\eta_i = 0$, then $\eta_{i+1} = 0$. We built the model S_U as follows $\mathcal{G}_i = \{2.i,\ 2.i+1,\ ...,\ n\}$, $1 \leq i \leq m$; the requirement $W : h(i) > h(i-1)+1$ *and if* $h(i-1) = n-3$, *then* $h(i) = n$ *and if* $h(i-1) = n$, *then* $h(i) = n$. For the corresponding unranking model S_U, we have $\{h\} = \{h\}_{\mathcal{G}_i}$.

□

We will present now a case of modelling classes of planar figures using the image and the shortest path transforms [22], [21]. These representations are very convenient for massively parallel or massively distributed processing in real or virtual depth search machines [22].

Example 5.2.3 *With denotation* $\Im[F] = < \mathrm{n}_1,\ \mathrm{n}_2,\ ...,\ \mathrm{n}_n,\ \mathrm{f}_1,\ \mathrm{f}_2,\ ,\ ...,\ \mathrm{f}_n >$ *we mean the* $\Im$ *– transform of a planar figure* F *. The figure* F *is a polygon if* $< \mathrm{n}_1,\ \mathrm{n}_2,\ ...,\ \mathrm{n}_n,\ \mathrm{f}_1,\ \mathrm{f}_2,\ ,\ ...,\ \mathrm{f}_n >$ *can be partitioned into a number* k *of blocks* $< B_1,\ B_2,\ ...,\ B_k >$ *each one containing a string of consecutive elements that equal. By two consecutive elements, we understand one of the following pairs* $(\mathrm{n}_i,\ \mathrm{n}_{i+1})$, $(\mathrm{f}_i,\ \mathrm{f}_{i+1})$, $(\mathrm{n}_n,\ \mathrm{f}_1)$, $(\mathrm{f}_n,\ \mathrm{n}_1)$. *Therefore, a block of consecutive elements can be stated as follows* $B_j =< \mathrm{n}_{i_j},\ \mathrm{n}_{i_j+1},\ ...,\ \mathrm{n}_{i_j+r_i} >$, *or* $B_j =< \mathrm{f}_{i_j},\ \mathrm{f}_{i_j+1},\ ...,\ \mathrm{f}_{i_j+r_i} >$, *or* $B_j =< \mathrm{n}_{i_j},\ \mathrm{n}_{i_j+1},\ ...,\ \mathrm{n}_n,\ \mathrm{f}_1,\ \mathrm{f}_2,\ ...,\ \mathrm{f}_{n-i_j+r_i} >$ *or* $B_j =< \mathrm{f}_{i_j},\ \mathrm{f}_{i_j+1},\ ...,\ \mathrm{f}_n,\ \mathrm{n}_1,\ \mathrm{n}_2,\ ...,\ \mathrm{n}_{n-i_j+r_i} >$, *where* r_j *is the cardinal of the block* B_j. *Moreover,* $s \leq r_j \leq n-1$ *and* $d \leq k \leq m$, *where* n *and* m *are given. The*

task is to represent all the classes of the similar and mutually non–rotated polygons, i.e., all the possible distributions of $\Im[F]$ *into blocks* B_j.

The concept of an empty block B_j enables us to represent always m blocks $< B_1, B_2, ..., B_m >$. Among them the first k blocks are non–void, while the last $m-k$ blocks are empty. Each block is specified uniquely if we give its first and its last elements. For empty blocks the first and the last elements equal. With o_q we denote both the elements n_i and f_i that $q = i$ for n_i, while $q = n + i$ for representing f_i. Then, $\Im[F] =< o_1, o_2, ..., o_n, o_{n+1}, ..., o_{2n} >$. Let h be a function $h : j \rightarrow h(j)$, so that $h(j)$ represents the last element $o_{h(j)}$ of the block B_j. Then, we assign the mapping η:$B_j = \{o_{h(j-1)+1} \div o_{h(j)}\}$, $1 \leq j \leq m-1$; $B_m = \{o_{h(m)+1} \div o_{h(1)}\}$, i.e., $B_m = \{o_{h(m)+1} \div o_{2n}, o_1 \div o_{h(1)}\}$ if $h(m) > h(1)$ and $B_m = \{o_{h(m)+1}, o_{h(m)+2}, ..., o_{h(1)}\}$ if $h(m) < h(1)$. Now, we have to specify the model $S_U = << \mathcal{G}_j >, 1 \leq j \leq m; W; \{h\}_{\mathcal{G}_j} >$ that $\{h\} = \{h\}_{\mathcal{G}_j}$.

We have the following formulas specifying the indexed sets $\mathcal{G}_j$.

(1) $\mathcal{G}_j = \{1 + s(j-1) \div 2n - (d-j).s\}, 1 \leq j \leq d-1$;

(2) $\mathcal{G}_j = \{1 + s.d \div 2n\}$, $d \leq j \leq m-1$;

(3) $\mathcal{G}_m = \{1 \div 2n, \}$, $j = m$.

Correspondingly, the requirement W possess the following specifications:

(1) $W : h(j) \geq h(j-1) + s, 1 \leq j \leq d-1$;

(2) $W : h(j) \geq h(j-1) + s$ or if $2n - h(j-1) + h(1) < r_m$, then $h(j) = h(j-1)$, $d \leq j < m$;

(3) $W : [h(1) - h(m) < r_m$ for $h(m) < h(1)]$ or $[(2n - h(m)) + h(1) \leq r_m$ and $h(m) - h(m-1) \leq r_m\ ]$ for $h(m) > h(1)$.

One can show that the model is reduced and the Q property holds.

Using similar approach as the given in the previous example, we can make the unranking representation by means of a number of the unranking models S_U^d, S_U^{d+1}, ..., S_U^m that correspond into the possible number of vertices of the classified polygons.

□

Very similar thinking can be used for classifications of the shortest paths within planar polygon following the approach given in [21].

The next example shows that we can make only one model S_U for representing wanted collection of substructures S', while using the classical approach making such a single model is really a difficult task.

Example 5.2.4 *Given is a graph* $G =< V, E >$, *where* $V = \{v_1, v_2, ..., v_{10}\}$ *and each pair* $(v_i, v_{i+1}) \in E$. *The task is to represent a collection of subgraphs* $G' =< V', E' >$ *that* $| V' | = 4$ *and each subgraph contains exactly two vertices from the set* $\{v_1 \div v_5\}$, *moreover for each edge* $(v_p, v_q) \in E'$, *we have to have* $q \neq p \pm 1$. *The set* E' *is created by restriction of the set* E *into the set of edges containing pairs* $(v_p, v_q) \in V'$.

The first task is to determine the set of the choice functions that represent the requested collection of the subgraphs. We observe that each subgraph G' is created by selecting a subset of four vertices that no neighboring vertices belong to V' and $V' = \{v_p, v_q, v_s, v_t\}$, where $p \leq 5$, $q \leq 5$, $s \geq 5$, $t \geq 5$. Then, each subset V' can be represented by a choice function $h(1)$, $h(2)$, $h(4)$, $h(5)$ whose values correspond to the indexes p, q, s and t. We will model the requested set of choice functions $\{h\}$ as the set $\{h\}_{\mathcal{R}_i}$ that $W : h(i) \geq h(i-1) + 2$. Then, the indexed family $< \mathcal{R}_i >$, $1 \leq i \leq 4$ is as follows:

$\mathcal{R}_1 = \{1, 2, 3\}$, $\mathcal{R}_2 = \{3, 4, 5\}$, $\mathcal{R}_3 = \{6, 7, 8\}$, $\mathcal{R}_4 = \{8, 9, 10\}$.

The set of choice functions $\{h\}_{\mathcal{R}_i}$ for the unranking model $S_U =<< \mathcal{R}_i >, 1 \leq i \leq 4; W : h(i) \geq h(i-1) + 2; \{h\}_{\mathcal{R}_i} >$ is symmetric. Then, the table D for this model is the table DCM$[3 \times 4] = \left| \begin{array}{cccc} 10 & 6 & 3 & 1 \\ 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{array} \right|$. So, the ranking representation model is $S_R =< S_U, <^h$, table DCM$[3 \times 4] >$. Observe that the task is very simple for modelling when using the choice functions approach. Simultaneously this problem becomes very difficult for modelling, when using the classical concept of combination. Moreover, when we use the classical concept of combination then after generation of each combination, we have to use four substitution instructions in order to represent the corresponding subgraph G'.

□

We are concerned now with modelling sets of combinatorial objects that are specified by a number of the objects that belong to that set of the combinatorial objects. That problems comes from an implementation of genetic or evolutionary algorithms [48]. Frankly speaking the problem can be specified as follows. In advance, we have a set of choice functions $\{h\}$ representing a given set of combinatorial objects A. Then, a random subset of choice functions $\{p\} \subset \{h\}$ is selected that the choice functions $\{p\}$ satisfy in a best way a given in advance criterion. The goal is to make a model S_U corresponding to a set of choice functions $\{h'\} \subset \{h\}$ that $\{p\} \subset \{h'\}$ and $|\{h\}| >> |\{h'\}|$. The method of making the set $\{h'\}$ must also be exactly precessed for a given task. For instance, we can request that each feature of $h' \in \{h'\}$ is a corresponding feature of at least one $p \in \{p\}$ or we can request that each feature of each $h' \in \{h'\}$ is the corresponding common feature of the set $\{p\}$ or if the feature is not common for $\{p\}$, then a corresponding feature of $\{h\}$ is to be selected. The following example explains methods of making the models S_U desired.

Example 5.2.5 *Suppose, three permutations, for instance* $< 1, 7, 3, 2, 4, 5, 6 >$, $< 2, 3, 5, 4, 6, 7, 1 >$ *and* $< 3, 2, 5, 1, 7, 6, 4 >$ *have been selected from the full set of permutations for* $n = 7$, *i.e., from the group of symmetry* $Sym(7)$. *The goal is to model a set* $\{h\}_{G_i}$ *that is a subset*

of Sym(7) obtained by "crossing" the given three permutations, i.e., each permutation $h \in \{h\}_{\mathcal{G}_i}$ possess every pair of consecutive elements that is representative at least for one given permutation. Additionally the set $\{h\}_{\mathcal{G}_i}$ has to be isomorph-free.

The permutation $< 1,7,3,2,4,5,6 >$ defines the following pairs of consecutive elements: after the element 1, we can have the element 6 or 7, so there are the pairs $(!,6)$ or $(1,7)$. Similarly, we have $(7,1)$ or $(7,3)$; $(2,4)$ or $(2,3)$; $(4,5)$ or $(4,2)$; $(5,6)$ or $(5,4)$; $(6,1)$ or $(6,5)$.

The permutation $< 2,3,5,4,6,7,1 >$ defines the following pairs of the consecutive elements: $(2,3)$ or $(2,1)$; $(3,5)$ or $(3,2)$; $(5,4)$ or $(5,3)$; $(4,6)$ or $(4,5)$; $(6,7)$ or $(6,4)$; $(7,1)$ or $(7,6)$; $(1,2)$ or $(1,7)$.

The permutation $< 3,2,5,1,7,6,4 >$ defines the following pairs of the consecutive elements: $(3,2)$ or $(3,4)$; $(2,5)$ or $(2,3)$; $(5,1)$ or $(5,2)$; $(1,7)$ or $(1,5)$; $(7,6)$ or $(7,1)$; $(6,4)$ or $(6,7)$; $(4,3)$ or $(4,6)$.

The specified pairs of the consecutive points are the basic data to be used for producing the needed unranking model $S_U = << \mathcal{G}_i >, 1 \leq i \leq 7; W; W_1; \{h\}_{\mathcal{G}_i} >$. Since the goal is to produce permutations, so we have to model a set of bijections, hence the requirement W is to be selected as specified in 1.19. The produced pairs of the consecutive elements are to be used for defining the indexed family $< \mathcal{G}_i >, 1 \leq i \leq 7$, and the requirement W_1. For designing the model S_U, we have also to make assertion that the set $\{h\}_{\mathcal{G}_i}$ must contain isomorph-free permutations. The term isomorph-free is generally ambiguous, since that means the modeled set $\{h\}_{\mathcal{G}_i}$ should not contain any subset of permutations that could be obtained one from another by using a selected operation or selected operations. Such operations that could be applied for the subset of $\{h\}_{\mathcal{G}_i}$ are selected basing on the application of the requested modeling. Suppose, the requested set $\{h\}_{\mathcal{G}_i}$ is to be tested for containing possibly the best solutions of TS (travelling salesman) problem [48].[1]

Then, the operations producing isomorphic solutions are:

(i) cyclic shifting $\xrightarrow{k}$ that is specified as follows: $\xrightarrow{1}(< 1, 7, 3, 2, 4, 5, 6 >) = < 7, 3, 2, 4, 5, 6, 1 >$, while $\xrightarrow{3}(< 1, 7, 3, 2, 4, 5, 6 >) = < 2, 4, 5, 6, 1, 7, 3 >$ and so on.

(ii) rotation that is here denoted and defined as follows: $< 1, 7, 3, 2, 4, 5, 6 >^{-1} = < 1, 6, 5, 4, 2, 3, 7 >$.

In order to make the set $\{h\}_{\mathcal{G}_i}$ being a kernel, i.e., no two h_1 and h_2 can be obtained one from the other by using rotation or cyclic shifting. In order to make assertion that cyclic shifting is eliminated it is enough to assign a fixed value $h(1)$ for each $h \in \{h\}_{\mathcal{G}_i}$. Asserting that we have eliminated rotation is much more difficult to get for the general case. Nevertheless, for

[1] We selct TS problem since it is widely known and understanding the term "isomorph-free" can be supported intuitively but we do not claim that such approach to TS is the best possible.

our example, if we assume that $h(1) = 1$ and we enable only one pair of the consecutive elements for each permutation that is additionally common for all of them, i.e., the pair $(1,7)$, then we make assertion that $\{h\}_{\mathcal{G}_i}$ would be the kernel.

From our considerations, we conclude that the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq 7$, should be as follows:

$\mathcal{G}_1 = \{1\}$; $\mathcal{G}_2 = \{2\}$; $\mathcal{G}_3 \div \mathcal{G}_7$ are subsets of $\{2, 3, 4, 5, 6\}$, more accurate specification requires examination of the pairs of the consecutive points. Since $h(2) = 7$, so $\mathcal{G}_3 = \{3, 6\}$, $\mathcal{G}_4 = \{2, 4, 5\}$, $\mathcal{G}_5 = \mathcal{G}_6 = \mathcal{G}_7 = \{2, 3, 4, 5, 6\}$. Moreover, the pairs of the consecutive points are used for the formulation of the requirement W_1.

Then, we have the following specification of W_1 :

if $h(i-1) = 2$, then $h(i) = 3$ or $h(i) = 4$ or $h(i) = 5$;
if $h(i-1) = 3$, then $h(i) = 2$ or $h(i) = 4$ or $h(i) = 5$;
if $h(i-1) = 4$, then $h(i) = 2$ or $h(i) = 3$ or $h(i) = 5$ or $h(i) = 6$;
if $h(i-1) = 5$, then $h(i) = 2$ or $h(i) = 3$ or $h(i) = 4$ or $h(i) = 6$;
if $h(i-1) = 6$, then $h(i) = 4$ or $h(i) = 5$;
if $h(i-1) = 7$, then $h(i) = 3$ or $h(i) = 6$. .

Accidently the obtained model is reduced and the Q property holds. For instance, the permutation $h =< 1, 7, 6, 4, 5, 2, 3 >$ belongs to $\{h\}_{\mathcal{G}_i}$ and it is not isomorphic to any given input.

The given two methods of making the requirement W_1 are not the only possible ones. We can much extend the logic formulas for construction of the requirement W_1. For instance, a threshold enabling usage of an elementary request can be investigated. Suppose, we say that the requirement is valid if it comes at least from two permutations. Then, we would have if $h(i-1) = 2$, then $h(i) = 3$ or $h(i) = 4$, since 3 comes from all three permutations, while 4 comes from the first and from the third permutation.

□

The similar problem was considered earlier in [45] but the approach presented here enables us to specify the set of objects to be generated at the level of modeling. We also observe that a number of ways for making the model S_U can be developed depending on the given application. Nevertheless, we have no assertion that such obtained models S_U would be reduced and the Q property would hold, however it can happen that the obtained model is reduced and the Q property holds as the given example shows. Since for these two classes of models the generation problems are quite different, so the approach to the generation should not be oriented towards a representation but it should be dependent on structural properties of the model. That statement makes essential progress in approaching implementations of the genetic algorithms.

5.3 Forms of the requirement W

Simplicity of the requirement W has important meaning for efficiency of implementation of the algorithms concerning numerous evaluations. Moreover, data structure used for representing W has also important meaning for efficiency of the algorithms used. In fact, if the requirement W is easier for testing, then the model is more suitable for the generation of the corresponding choice functions. There are also forms of the requirement W that are suitable for model evaluations performed in order to produce a more convenient representation. So, several different data structures for representing W can be used at different levels of the model evaluation or we can use these data structures for the generation of choice functions. Till now, we have shown intuitive methods of modelling the requirement W. In this section, we will develop a formal methods of transformation of the requirement W. We mean the goal is to select a most suitable for evaluations data structure or we can even transform given unranking model in order to have the requirement W as simple as possible. The main advantage of the proposed method lies in possibility of arbitrary simplification of a given W. We can even transform a given representation with extended requirement W into a representation with void requirement W. The price we have to pay for getting simpler and simpler the requirement W is the growing number of models S_{ij}^k that together replace one given model S_{ij}. At the limit of this simplification process, we can always get void requirement W but then the number of models S_{ij}^k can equal the number of choice functions to be represented, so one model S_{ij}^k would contain one choice function. Of course, such simplification of the requirement W has no sense, so for a request of simplification of W a compromise is needed.

Given is the reduced unranking model $S_{ij} = << \mathcal{G}_i >, 1 \leq i \leq m; W; W_i; \{h\}_{x_i} >$.

Let $\pi_i =< \pi_i^1, \pi_i^2, \ldots, \pi_i^{n_i} >$ be a partition on the indexed set $\mathcal{G}_i$, $1 \leq i \leq m$, $1 \leq n_i \leq |\mathcal{G}_i|$. So, with denotation π_i^j we mean a block of the partition π_i. Consequently, we have a layer $< \pi_i^j >$, $1 \leq i \leq m$, $1 \leq j \leq n_i$, of the indexed family $\mathcal{G}_i$, $1 \leq i \leq m$, and by $x = < x(1), x(2), \ldots, x(m) >$ we denote the corresponding choice function, i.e., $x : i \longrightarrow \pi_i^j$. So, we identify $x(i)$ with a block π_i^j of the partition π_i.

Suppose, we have some unranking model $S_{ij} =<< \pi_i >, 1 \leq i \leq m; W; \{x\}_{x_i} >$ that the choice functions $\{x\}_{x_i}$ represent splitting the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$ into non-empty layers. Let the model S_{ij} be reduced, while the Q property being hold.

Definition 5.3.1 *We say that the model S_{ij} is consistent with requirement W_i if for each $x(i) \in \{x\}_{x_i}$ the following property holds: a value $q_i \in x(i)$ only if q_i satisfies W_i for all the choice functions (partial mappings) $h \in \{h\}_{x_i}$ that $h(1) \in x(1)$, $h(2) \in x(2)$, ..., $h(i-1) \in x(i-1)$.*

If the model S_U^{π} is consistent with requirement W_1, then $W^{\pi} \Longleftrightarrow W_1$, i.e., for producing the choice functions $\varkappa \in \{\varkappa\}_{\pi_i}$, we have to use the requirement W_1 transformed into the corresponding relational expression involving the blocks $\pi_i^j \in \pi_i$ instead of the elements $q \in \mathcal{G}_i$. For a given unranking model S_U, we can have a number of the models S_U^{π} consistent with W_1. We have assertion that for every model S_U there is at least trivial model S_U^{π} that is consistent with W_1. One can note it by observing that taking the minimum partition of each indexed block, i.e., identifying each element $q_j \in \mathcal{G}_i$ with the block π_i^j of the partition π_i, $1 \leq i \leq m$, we get the model S_U^{π} that is consistent with W_1, since for each choice function $h \in \{h\}_{\mathcal{G}_i}$ there is the choice function $\varkappa \in \{\varkappa\}_{\pi_i}$ and $\mid \{\varkappa\}_{\pi_i} \mid = 1$.

Definition 5.3.2 *A given model S_U^{π} consistent with W_1 is minimal if there is no other model $S_U'^{\pi}$ consistent with W_1 that* $\mid \{\varkappa'\}_{\pi_i} \mid < \mid \{\varkappa\}_{\pi_i} \mid$.

We will present now the following example in order to demonstrate the investigated concepts.

Example 5.3.1 *Given is the following model S_U :*

$\mathcal{G}_1 = \{1, 2, 5, 7, 9, 10\}$, $\mathcal{G}_2 = \{1, 2, 3, 4, 5, 6\}$, $\mathcal{G}_3 = \{2, 3, 4, 7, 8, 10\}$, $\mathcal{G}_4 = \{4, 5, 6, 7, 8, 9\}$,

$W : h(i) \neq h(i-1) \div h(i) \neq h(1)$; W_1 : if $h(i-1)$ is even then $h(i)$ is odd else $h(i)$ is even.

The first model S_U^{π}.

Each indexed set $\mathcal{G}_i$, we divide in two blocks, the block $\mathcal{G}_i^1$ contains odd elements while $\mathcal{G}_i^2$ contains even elements. So, we have $\mathcal{G}_1 = \{\mathcal{G}_1^1, \mathcal{G}_1^2\} = \{\{1, 5, 7, 9\}, \{2, 10\}\}$. Similarly, $\mathcal{G}_2 = \{\{1, 3, 5\}, \{2, 4, 6\}\}$, $\mathcal{G}_3 = \{\{3, 7\}, \{2, 4, 8, 10\}$, $\mathcal{G}_4 = \{\{5, 7, 9\}, \{4, 6, 8\}\}$. We make assignment: π_i corresponds into $\mathcal{G}_i$, while π_i^1 corresponds into $\mathcal{G}_i^1$ and π_i^2 corresponds into $\mathcal{G}_i^2$. Respectively, we have the indexed family $\pi_1 = \{\pi_1^1, \pi_1^2\}$, $\pi_2 = \{\pi_2^1, \pi_2^2\}$, $\pi_3 = \{\pi_3^1, \pi_3^2\}$, $\pi_4 = \{\pi_4^1, \pi_4^2\}$. The requirement W^{π}: if $\varkappa(i-1) = \pi_{i-1}^2$, then $\varkappa(i) = \pi_i^1$ else $\varkappa(i) = \pi_i^2$, $2 \leq i \leq 4$. The model S_U^{π} is consistent with W_1.

The second model $S_U'^{\pi}$.

We make the following partitioning the indexed sets $\mathcal{G}_i$: $\mathcal{G}_1 = \{\{1, 5\}, \{7, 9\}, \{2, 10\}\}$, $\mathcal{G}_2 = \{\{1, 3\}, \{5\}, \{2, 4\}, \{6\}\}$, $\mathcal{G}_3 = \{\{3\}, \{7\}, \{2, 4, 8, 10\}\}$, $\mathcal{G}_4 = \{\{5, 7\}, \{9\}, \{4, 6, 8\}\}$. Then, we have the corresponding indexed family $< \pi_i' >$, $1 \leq i \leq 4$ as follows: $\pi_1' = \{\pi_1'^1, \pi_1'^2, \pi_1'^3\}$, $\pi_2' = \{\pi_2'^1, \pi_2'^2, \pi_2'^3, \pi_2'^4\}$, $\pi_3' = \{\pi_3'^1, \pi_3'^2\}$, $\pi_4' = \{\pi_4'^1, \pi_4'^2, \pi_4'^3, \}$. The requirement W'^{π} is as follows: W'^{π} : if $\varkappa'(i-1) > 2$, then $\varkappa'(i) < 3$ else $\varkappa'(i) > 2$, $2 \leq i \leq 4$.

The models S_U^{π} and $S_U'^{\pi}$ are consistent with W_1 and the model S_U^{π} is the minimal one.

□

For the above example each choice function $\varkappa \in \{\varkappa\}_{\pi_i}$ corresponds to the non−empty layer $< \mathcal{G}_i^{q_i} >$, $1 \leq i \leq m$ of the family $< \mathcal{G}_i >$, $1 \leq i \leq m$.

Then, we have the formal model $S_U^{\varkappa} =<< \mathcal{G}_i^{q_i} >, 1 \leq i \leq m; W; \{h\}_{\mathcal{G}_i^{q_i}} >$. Comparing the model $S_U^{\varkappa}$ with the given model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; W_1; \{h\}_{\mathcal{G}_i} >$ one can observe lack of the requirement W_1 in $S_U^{\varkappa}$. That property is not incidental, we will present now considerations concerning equivalence of the unranking representations S_U and $< S_U^{\varkappa} >_{\varkappa \in \{\varkappa\}_{\pi}}$, where with $< S_U^{\varkappa} >_{\varkappa \in \{\varkappa\}_{\pi}}$ we mean the collection of all the models $< S_U^{\varkappa} >$ that each choice function $\varkappa$ is a choice function of the model S_U^{π} consistent with W_1.

Assumptions:

(*i*) Given is the model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; W_1; \{h\}_{\mathcal{G}_i} >$.

(*ii*) The model $S_U^{\pi} = S_U^{\pi} =<< \pi_i >, 1 \leq i \leq m; W^{\pi}; \{\varkappa\}_{\pi_i} >$ is consistent with W_1, so $W^{\pi} \Longleftrightarrow W_1$.

(*iii*) For each choice function $\varkappa$, we make the corresponding model $S_U^{\varkappa} =< < \mathcal{G}_i^{j_i} >, 1 \leq i \leq m; W; \{h\}_{\mathcal{G}_i^{j_i}} >$.

(*iv*) $< S_U^{\varkappa} >_{\varkappa \in \{\varkappa\}_{\pi}}$ is the collection of models obtained for every choice function $\varkappa$ of the model $S_U^{\varkappa}$.

Proposition 5.3.1 *The model S_U is equivalent to the collection of models $< S_U^{\varkappa} >_{\varkappa \in \{\varkappa\}_{\pi}}$.*

Proof. Since the model S_U^{π} is consistent with W_1, so any partial choice mapping $< \varkappa(1), \varkappa(2), ..., \varkappa(i) >$ represents a string of blocks $< \mathcal{G}_1^{j1}, \mathcal{G}_2^{j_2}, ..., \mathcal{G}_i^{j_i} >$ such that a string of elements $< h(1), h(2), ..., h(i) >$, $h(1) \in \mathcal{G}_1^{j1}, h(2) \in \mathcal{G}_2^{j_2}, ..., h(i) \in \mathcal{G}_i^{j_i}$ satisfies W_1, $1 \leq i \leq m$. Therefore, each choice function $h \in \{h\}_{\mathcal{G}_i^{j_i}}$ satisfies automatically the requirement W_1. Then, each choice function $< h(1), h(2), ..., h(m) >$ of the model $S_U^{\varkappa}$ must satisfy additionally the requirement W, see the assumption (*iii*). Hence, each choice function $h \in \{h\}_{\mathcal{G}_i^{j_i}}$ satisfies W and W_1. Since the indexed family $< \mathcal{G}_i^{j_i} >$, $1 \leq i \leq m$ is a layer of the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$ and since all the choice functions of the layer satisfy the same requirements as the choice functions $\{h\}_{\mathcal{G}_i}$, so $\{h\}_{\mathcal{G}_i^{j_i}} \subset \{h\}_{\mathcal{G}_i}$. The set $\{h\}_{\mathcal{G}_i}$ is partitioned into a number of sets $\{h\}_{\mathcal{G}_i^{j_i}}$ that each set is obtained for each choice function $\varkappa \in \{\varkappa\}_{\pi_i}$. Hence, the model S_U is equivalent to the collection of models $< S_U^{\varkappa} >_{\varkappa \in \{\varkappa\}_{\pi}}$.

□

The models $< S_U^{\varkappa} >_{\varkappa \in \{\varkappa\}_{\pi}}$ make assertion that the generation of the individual choice functions $\{h\}_{\mathcal{G}_i^{j_i}}$ does not require testing the requirement W_1. Then, we have the following corollary.

Corollary 5.3.1 *For every model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; \{h\}_{\mathcal{G}_i} >$, we can make a collection of models $S_U^{\varkappa} =<< \mathcal{T}_i >, 1 \leq i \leq m; \{h\}_{\mathcal{T}_i} >$ that is equivalent to S_U.*

Proof. Every requirement W can be given as a conjunction $T \wedge W_1 \wedge W_2 \wedge ... \wedge W_s$, where T is a tautology, while W_r $1 \leq r \leq s$ denotes any requirement of the general form, $1 \leq r \leq s$. Then, we can treat $T \wedge W_1 \wedge W_2 \wedge ..., \wedge W_{s-1}$ as a requirement W, while W_s would be treated as W_1. So, we can use Proposition 5.3.1 that the corresponding models $S_U^{\varkappa}$ need to use only new W. Continuing this procedure for each $S_U^{\varkappa}$ model, we get descending models $S_U^{\varkappa}$ that require the requirement T to be tested only. Since T is a tautology, so it does not need to be tested. Hence, we get descending models $S_U^{\varkappa}$ $=<< T_i >, 1 \leq i \leq m; \{h\}_{T_i} >$.

□

The general evaluations concerning simplification of the requirement W are difficult since the general form of the requirement W is as follows: $h(i)\mathrm{RE}[h(i-1), h(i-2), ..., h(1), i]$. If this general formula for W can not be simplified, then it makes real difficulties both for theoretical considerations concerning the discussed topic and also for development of corresponding computer algorithms. So, we make a simplification assuming that the concerned requirement W (W_1) is specified with the form $h(i)\mathrm{RE}[h(i-1), i]$.

The following sections are devoted into construction of the models S_U^{π} and $S_U^{\varkappa}$ under the above given assumptions. The reader not–interested in computer algorithms for making S_U^{π} neither in data structures representing W can skip these sections.

5.3.1 INITIAL W

Let a requirement W be of the form $h(i)\mathrm{RE}[h(i-1), i]$. Then, $h(i)\mathrm{RE}[h(i-1), i]$ can be represented as a list of pairs $(\{q_{i-1}\}, \{q_i\})$, $\{q_{i-1}\} \subset \mathcal{G}_{i-1}$, $\{q_i\} \subset \mathcal{G}_i$. If $h(1)$, $h(2)$, ..., $h(i-1)$ is a partial choice function and if $h(i-1) \in \{q_{i-1}\}$, $h(i) \in \{q_i\}$, then $h(1)$, $h(2)$, ..., $h(i-1)$, $h(i)$ is a partial choice function only if $h(i-1) \in \{q_{i-1}\}$ and $h(i) \in \{q_i\}$; $1 \leq i \leq m$. Observe, we do not make any other restrictions on the pairs $(\{q_{i-1}\}, \{q_i\})$, i.e., the sets $\{q_{i-1}\}$ and $\{q_i\}$ can contain only one item or a number of items. Moreover, for each element q_{i-1}, we can have a number of pairs and a fixed set $\{q_i\}$ can be the element of any number of the pairs. We declare the following abstract data structures:

WP=**record**
qi_1: set of integer;
qi : set of integer;
end;
WI= **indexed list of** WP;
W=**array**[2..m] of WI.

With denotation W[i,j].qi_1, we mean the set qi_1 that belongs to the $j-th$ item of the list WI being the $i-th$ element of the table W. For

denotation W[i,j].qi we have the similar meaning. We can have a number of representations of the requirement W using the data structures specified above. The basic representation we call INITIAL W and it is produced by the following algorithm INIT.

Algorithm INIT(for producing INITIAL W)

Input: The model $S_U = << \mathcal{G}_i >, 1 \leq i \leq m, W; \{h\}_{\mathcal{G}_i} >$ that W possess the structure $h(i)\mathrm{RE}[h(i-1), i]$

Output: INITIAL W that represents W.

1. for $i \longleftarrow 2$ to m do
1.1. $j \longleftarrow 0$;
1.2. for $q_{i-1} \longleftarrow 1$ to $\mid \mathcal{G}_{i-1} \mid$ do
1.2.1. for $q_i \longleftarrow 1$ to $\mid \mathcal{G}_i \mid$ do
1.2.1.1. if (q_{i-1}, q_i) satisfies W then
1.2.1.1.1. $j \longleftarrow j + 1$;
1.2.1.1.2. W[i,j].qi$\longleftarrow q_i$;
1.2.1.1.3. W[i,j].qi_1 $\longleftarrow q_{i-1}$;

Asymptotic complexity of the algorithm INIT is $O(n^2.m)$. The structure INITIAL W can be used for the choice function generation but for big values of the cardinals $\mid \mathcal{G}_i \mid$ this form of representation of the requirement W can be not very convenient since it can require examination of the number of records equal to all the pairs (q_{i-1}, q_i), we have for a fixed values q_{i-1}. The given indexing the list of records can reduce this difficulties, since each value q_{i-1} is represented by a number of consecutive indexes. Moreover, the values of indexes grow simultaneously with growing values of the matching elements q_i.

5.3.2 Consecutive representations of W

The next representation we produce is the representation COMBINED W that can be obtained form INITIAL W by the following algorithm COM.

Algorithm COM(for producing COMBINED W)

Input: The representation INITIAL W, $n_{i-1} = \mid \mathcal{G}_{i-1} \mid$

Output: COMBINED W that represents W.

1. for $i \longleftarrow 2$ to m do
1.1. for $j \longleftarrow 1$ to $n_{i-1} - 1$ do
1.1.1. for $1 \longleftarrow j + 1$ to n_{i-1} do
1.1.1.1. if W[i,j].qi_1 =W[i,l].qi_1 then
1.1.1.1.1. W[i,j].qi $\longleftarrow$W[i,j].qi $\cup$ W[i,l].qi;
1.1.1.1.2. remove W[i,j] (i.e., remove W[i,l].qi_1 and remove W[i,l].qi from W);
1.1.1.1.3. $n_{i-1} \longleftarrow n_{i-1} - 1$;

Each row of the table COMBINED W represents assignment of an element $q_{i-1} \in \mathcal{G}_{i-1}$ into a set of elements $\{q_i\}$ that each $q_i \in \{q_i\}$ satisfies W for the given q_{i-1}. We have assertion that each value q_{i-1} is represented exactly once as W[i,j].qi_1 in the table COMBINED W. Asymptotic complexity of the algorithm is $O(n^2.m)$. The table COMBINED W represents compact data that can be used easily for the algorithms concerning the generation of the choice functions and their usage. Any direct usage of the structure COMBINED W requires assertion that the records are not too big for handling. The representation COMBINED W can not be used for making the model S^{π}_{U} since collection of sets W[i,j].qi for a given i does not create any partition on the set $\mathcal{G}_i$.

Next, we make the representation PARTITION W that each set W[i,j].qi is a block of a partition on $\mathcal{G}_i$ for fixed i.

Algorithm PART(for producing PARTITION W)
Input: The representation COMBINED W,
Output: PARTITION W that represents W.
1. **for** $i \longleftarrow 2$ **to** m **do**
1.1. **for** $j \longleftarrow 1$ **to** $n_{i-1} - 1$ **do**
1.1.1. **for** $1 \longleftarrow j+1$ **to** n_{i-1} **do**
1.1.1.1. **if** W[i,j].qi $\cap$ W[i,l].qi $\neq \emptyset$ **then**
1.1.1.1.1. **if** W[i,j].qi=W[i,l].qi **then**
1.1.1.1.1.1. W[i,j].qi_1 $\longleftarrow$W[i,j].qi_1 $\cup$ W[i,l].qi_1;
1.1.1.1.1.2. remove W[i,l] {i.e., remove W[i,l].qi_1 and remove W[i,l].qi from W};
1.1.1.1.1.3. $n_{i-1} \longleftarrow n_{i-1} - 1$
else
1.1.1.1.2.1. **if** W[i,j].qi $\cap$ $\overline{\text{W[i, l].qi}} \neq \emptyset$ **then**
1.1.1.1.2.1.1 insert the set W[i,j].qi $\cap$ $\overline{\text{W[i, l].qi}}$ as W[i,j+1].qi;
1.1.1.1.2.1.2. insert the set W[i,j].qi_1 $\cup$ W[i,l].qi_1 as W[i,j+1].qi_1;
1.1.1.1.2.1.3. $n_{i-1} \longleftarrow n_{i-1} + 1$;
1.1.1.1.2.2. **if** W[i,l].qi $\cap$ $\overline{\text{W[i, j].qi}} \neq \emptyset$ **then**
1.1.1.1.2.2.1. insert the set W[i,l].qi $\cap$ $\overline{\text{W[i, j].qi}}$ as W[i,j+1].qi;
1.1.1.1.2.2.2. insert the set W[i,j].qi_1 $\cup$ W[i,l].qi_1 as W[i,j+1].qi_1;
1.1.1.1.2.2.3. $n_{i-1} \longleftarrow n_{i-1} + 1$;
1.1.1.1.2.3. W[i,j].qi $\longleftarrow$ W[i,l].qi $\cap$ W[i,j].qi;
1.1.1.1.2.4. W[i,j].qi_1 $\longleftarrow$ W[i,j].qi_1 $\cup$ W[i,l].qi_1;
1.1.1.1.2.5. remove W[i,l] {i.e., remove W[i,l].qi_1 and remove W[i,l].qi from W};

The step 1.1.1.1.2.1. is performed only if W[i,j].qi $\cap$ $\overline{\text{W[i, l].qi}} \neq \emptyset$. Similarly, the step 1.1.1.1.2.2. is performed only if W[i,l].qi $\cap$ $\overline{\text{W[i, j].qi}} \neq \emptyset$. These groups of steps are performed as the operations on lists. The block diagram given in the Table 1 explains performance of the step 1.1.1.1. Asymptotic complexity of the above algorithm is $O(n^2.m)$.

W[i,j].qi_1	W[i,j].qi
{1, 2}	{1, 3, 5, 6}
...	...
{5}	{1, 2, 4, 6}
...	...

W[i,j].qi_1	W[i,j].qi
{1, 2, 5}	{1, 6}
{1, 2}	{3, 5}
{ 5}	{2, 4}
...	...
...	...

Table1. The list WI before modification and after performing the step 1.1.1.1.

We observe that the subsets W[i,j].qi equal to the blocks of a partition on $\mathcal{C}_n$, but we have no assertion that the subsets W[i,j].qi_1 correspond to the blocks of a partition on $\mathcal{C}_n$. Nevertheless, the representation PARTITION W can be used for production of a non−trivial model S_W^n if the partition π_m is not minimum. The construction is as follows: $\pi_i =< \pi_i^1, \pi_i^2, ..., \pi_i^{r_i} >$, where π_i^j contains a single element q_i^j, $1 \leq i \leq m-1$, $1 \leq j \leq r_i$, while for the partition π_m its blocks correspond to different subsets W[m,j].qi. So, PARTITION W enables us to make a model S_W^n that is non-trivial. Nevertheless, such model is far from being optimal since the blocks of the indexed sets $\mathcal{C}_i$, $1 \leq i \leq m-1$, contain single elements.

5.3.3 CANONICAL W

Using the representation PARTITION W as input we can make the representation CANONICAL W that corresponds to a partition π_i for every indexed set $\mathcal{C}_i$, $2 \leq i \leq m$. For developing algorithms producing the CANONICAL W, we modify the data structures investigated in Section 5.3.1.

qi_1: set of integer;
qi : set of integer;
WP=record
QI_1: set of qi_1;
QI : set of qi;
end;
WI= indexed list of WP;
W=array[2..m] of WI.

Observe, the variables QI and QI_1 with capitals are collections of sets of integers, while their components are the sets qi and qi_1, respectively. Correspondingly the set operations $\cup$ and $\cap$ have doubled meaning depending whether we applied them to QI/QI_1 or to qi/qi_1. In fact, we use W[i,j].QI $\cap$ W[k,l].QI, W[i,j].QI $\cap$ $\overline{\text{W[k, l].QI}}$ in normal way intersecting all components W[i,j].qi with all components W[k,l].qi or $\overline{\text{W[k, l].qi}}$, respectively. The operation $\overline{\text{W[k, l].qi}}$ means denial, where the universal is $\mathcal{C}_n$. We perform W[i,j].QI $\cup$ W[k,l].QI only if all the elements W[i,j].qi and W[k,l].qi are disjoint. We use empty W[i,j].QI but $\emptyset$ component is not an element

of any not empty set. Then, we define the following procedures concerning the lists.

Procedure INSDEN(W[i,j].QI, W[i,l].QI)

1. FLAG$\longleftarrow$ *FALSE*;

2. **if** W[i,j].QI $\cap$ $\overline{\text{W[i, l].QI}} \neq \emptyset$, **then**

2.1. insert into the list a new record W[i,j+1]

2.2. W[i,j+1].QI $\longleftarrow$ W[i,j].QI $\cap$ $\overline{\text{W[i, l].QI}}$;

2.3. $n_i \longleftarrow n_i + 1$;

2.4. FLAG$\longleftarrow$ *TRUE*;

3. **return** WI, FLAG;

Procedure REMEQ(W[i,l].QI or W[i,l].QI_1)

1. remove from the list the record W[i,l] { i.e., remove both W[i,l].QI and W[i,l].QI_1}

2. $n_i \longleftarrow n_i - 1$;

3. **return** WI;

Procedure SPLIT(W[i,j].QI_1, W[i,l].QI_1)

1. **if** W[i,j].QI_1 $\cap$W[i,l].QI_1 $\neq \emptyset$ **then**

1.1. call INSDEN(W[i,j].QI_1, W[i,l].QI_1);

1.2. **if** FLAG= *TRUE* **then** W[i,j+1].QI$\longleftarrow$W[i,j].QI;

1.3. call INSDEN(W[i,l].QI_1, W[i,j].QI_1);

1.4. **if** FLAG= *TRUE* **then** W[i,j+1].QI$\longleftarrow$W[i,l].QI;

1.5. **if** W[i,j].QI_1 $\neq$W[i,l].QI_1 **then**

1.5.1. W[i,j].QI_1 $\longleftarrow$W[i,j].QI_1 $\cap$W[i,l].QI_1;

1.6. W[i,j].QI $\longleftarrow$W[i,j].QI $\cup$W[i,l].QI;

1.7. call REMEQ(W[i,l].QI); {observe, the record W[i,l] denotes it before entering SPLIT}

The Table 2 shows the result of running the procedure SPLIT.

W[i,j].QI_1	W[i,j].QI
{{1, 2, 5}}	{{1, 6}}
...	...
{{5, 6}}	{{2, 4}}
...	...

W[i,j].QI_1	W[i,j].QI
{{5}}	{{1, 6},{2, 4}}
{{1, 2}}	{{1, 6}}
{{6}}	{{2, 4}}
...	...
...	...

Table 2. The list WI before and after performing procedure SPLIT for two records.

After running procedure SPLIT for all pairs (W[i,j].QI_1, W[i,j].QI) and (W[i,l].QI_1, W[i,l].QI) that $W[i,j].QI\_1 \cap W[i,l].QI\_1 \neq \emptyset$ all the sets W[i,j].QI contain blocks of a partition on $\mathcal{G}_i$, while each W[i+1, l].QI_1 represents a block of a partition on $\mathcal{G}_i$. These two partitions must match, i.e., the blocks defined by W[i+1, l].QI_1 and W[i,j].QI can not be inconsistent. We develop the procedure MATCH for removing inconsistency that is introduced by a pair (W[i,j].QI ,W[i+1, l].QI_1). The result of running the procedure MATCH is given in Table 3.

		W[i+1, j].QI_1	W[i+1, j].QI
W[i,j].QI_1	W[i,j].QI	{{5, 6}}	{{1, 6},{2, 4}}
{{1, 2, 5}}	{{3, 5}}	...	...
...	...	...	...
...	...	...	...
...	...		
		...	...

		W[i+1, j].QI_1	W[i+1, j].QI
W[i,j].QI_1	W[i,j].QI	{{5}}	{{1, 6},{2, 4}}
{{1, 2, 5}}	{{3},{5}}	{{3}}	{}
...	...	{{6}}	{{1, 6},{2, 4}}
...	...	...	...
...	...		
		...	...

Table 3. The lists WI and W(I+1) before and after performing procedure MATCH for two records

We modify the procedure INSDEN producing the procedure INSFIX:

Procedure INSFIX(W[i+1, j].QI_1, W[i,l].QI)

1. FLAG ← *FALSE*;

2. if $W[i+1, j].QI\_1 \cap \overline{W[i, l].QI} \neq \emptyset$, then

2.1. insert into the list a new record W[i+1, j+1];

2.2. $W[i+1, j+1].QI\_1 \leftarrow W[i+1, j].QI\_1 \cap \overline{W[i, l].QI}$

2.3. $n_i \leftarrow n_i + 1$;

2.4. FLAG ← *TRUE*;

3. FL2 ← *FALSE*;

3.1. if $\overline{W[i+1, j].QI\_1} \cap W[i, l].QI \neq \emptyset$, then

3.1. insert into the list a new record W[i,j+1]

3.2. $W[i+1, j+1].QI \leftarrow \overline{W[i+1, j].QI\_1} \cap W[i, l].QI$

3.3. $n_i \longleftarrow n_i + 1$;

3.4. FL2 $\longleftarrow$ *TRUE*;

4. **return** WI, FLAG, FL2;

Then, the procedure MATCH(W[i+1, j].QI_1, W[i,l].QI) is as follows:

Procedure MATCH(W[i+1, j].QI_1, W[i,l].QI)
1. **if** W[i,l].QI $\cap$ W[i+1, j].QI_1 $\neq \emptyset$ **then**
1.1. call INSFIX((W[i+1, j].QI_1, W[i,l].QI);
2. **if** FLAG= *TRUE* **then** W[i+1, j+1].QI$\longleftarrow$W[i+1, j].QI;
3. **if** FL2 = *TRUE* **then**
3.1. **if** FLAG= *TRUE* **then** W[i+2, j+1].QI$\longleftarrow${}
else W[i+1, j+1].QI$\longleftarrow${};
4. **if** W[i,l].QI $\neq$ W[i+1, j].QI_1 **then** W[i+1, j].QI_1 $\longleftarrow$W[i,l].QI $\cap$ W[i+1, j].QI_1;
5. W[i,l].QI $\longleftarrow$ {W[i,l].QI $\cap$ W[i+1, j].QI_1, W[i,l].QI $\cap$ $\overline{\text{W[i + 1, j].QI\_1}}$, W[i, l].QI $\cap$ W[i+1, j].QI_1};
6. **return** WI.

We present now the algorithm CAN for producing the CANONICAL W.

Algorithm CAN(for producing CANONICAL W)
Input: The representation COMBINED W,
Output: CANONICAL W that represents W.
1.1.1. **for** $1 \longleftarrow j + 1$ **to** n_1 **do**
1.1.1.1. call SPLIT(W[1, j].QI_1, W[1, l].QI_1);
2. **for** $i \longleftarrow 2$ **to** m **do**
2.1. **for** $j \longleftarrow 1$ **to** $n_i - 1$ **do**
2.1.1. **for** $1 \longleftarrow j + 1$ **to** n_i **do**
2.1.1.1. call SPLIT(W[i,j].QI_1, W[i,l].QI_1);
2.1.2. **for** $1 \longleftarrow 1$ **to** n_{i-1} **do**
2.1.2.1 call MATCH(W[i,j].QI_1, W[i−1, l].QI);

Asymptotic complexity of the algorithm CAN is $O(n^2.m)$. The representation CANONICAL W specifies the requirement W^{π}, while the subsets that are present in W correspond to the blocks of the partitions π_i for $1 \leq i \leq m$. Therefore, the representation CANONICAL W defines uniquely the optimal model S_U^{π} that is consistent with W_1. In order to observe that the model is optimal one should note that the representation PARTITION W contains a minimum number of subsets enabling conversion of PARTITION W into INITIAL W. Then, CANONICAL W removes only conflicts that are present in PARTITION W.

5.4 Modelling sets specified by rank

Given is a model S_R and a subset of choice functions $\{h'\}$ is specified by a rank range $R(\{h'\})$ in $R(h)$. If the rank range $R(\{h'\})$ in $R(h)$ is $z-$tuple and if $z >> 1$, then basing on Ranking Theorem we can often make a more suitable model S'_R that the given set $\{h'\}$ would be specified by the rank range using this new model. The goal is to diminish z as much as possible that is possible only if $\{h'\}$ is symmetric or if it contains very few non-symmetric and compact subsets. If it happens that the set $\{h'\}$ is non-symmetric with a big number of compact and isolated non-symmetric subsets $\{h'\}_k$, then basing on Ranking Theorem, we know that an acceptable ranking model S'_R does not exist. Then, we can demand only an unranking model $S'_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; \{h'\}_{\mathcal{G}_i} >$ that the given set $\{h'\} = \{h'\}_{\mathcal{G}_i}$ instead we request S'_R and specify $\{h'\}$ by rank. Any way, we have always to make the unranking model S'_U that $\{h'\} = \{h'\}_{\mathcal{G}'_i}$ or if a ranking representation is needed and the set $\{h'\}$ is non-symmetric, we can ask $\{h'\} \subset \{h'\}_{\mathcal{G}'_i}$ but z' is to be as small as possible.

There are the following methods for making a suitable model S'_U or S'_R

(i) replacing indexed family $< \mathcal{G}_i >, 1 \leq i \leq m$ of the model S_U by its subfamily $< \mathcal{G}'_i >, 1 \leq i \leq m$ that $\mathcal{G}'_i \subseteq \mathcal{G}_i$

(ii) imposing an additional requirement W_1,

(iii) both making a subfamily and imposing additional requirement.

Consider now making a subfamily $< \mathcal{G}'_i >, 1 \leq i \leq m$. For a given set $\{h'\}$, we make the sets $\mathcal{G}'_i$ containing all the possible values $h'(i)$ taken by all the choice functions $h' \in \{h'\}$, $1 \leq i \leq m$.

If it happens that $\{h'\} = \{h'\}_{\mathcal{G}'_i}$, then unranking modelling is finished since $S'_U =<< \mathcal{G}'_i >, 1 \leq i \leq m; W; \{h'\}_{\mathcal{G}'_i} >$. Otherwise, we have to have a partial choice mapping $< h'(i_1), h'(i_2), ..., h'(i_p) >$ that the requirement W holds for it and no its extension belongs to $\{h'\}$, while $h'(i_1) \in \mathcal{G}'_{i_1}, h'(i_2) \in \mathcal{G}'_{i_2}, ..., h'(i_p) \in \mathcal{G}'_{i_p}$. In order to make assertion that the Q property holds for the model S'_R, we have to create the requirement W_1 that the partial mapping $< h'(i_1), h'(i_2), ..., h'(i_p) >$ contradicts it.

The first example shows only the modelling by making the indexed subfamily $< \mathcal{G}'_i >, 1 \leq i \leq m$.

Example 5.4.1 *Given is the model* $S_U = << \mathcal{G}_i >, 1 \leq i \leq 5, \mathcal{G}_i \subseteq \mathcal{U} = \{1, 2, ..., 9\}$; $W(i)$ given in (1.1); $\{h\}_{\mathcal{G}_i} >$, *and the set* $\{h'\}$ *is specified by the rank range* $R(\{h'\})$ *in* $R\{h\}$ *as follows:* $\{(9,3), (19,3), (25,3), (39,3), (45,3), (55,3), (74,3), (80,3), (90,3), (105,3)\}$. *The task is to make the model* S'_R *that the value* z' *would be as small as possible for the corresponding rank range* $R(\{h'\})$ *in* $R\{h'\}$, .

The indexed family for the model S_U is as follows:
$\mathcal{G}_1 = \{1,2,3,4,5\}$, $\mathcal{G}_2 = \{2,3,4,5,6\}$, $\mathcal{G}_3 = \{3,4,5,6,7\}$,
$\mathcal{G}_4 = \{4,5,6,7,\ 8\}$, $\mathcal{G}_5 = \{5,6,7,8,9\}$.

Since the set $\{h\}_{\mathcal{G}_i}$ is symmetric, so we have the corresponding table DCM as follows:

$$\left|\begin{array}{ccccc} 70 & 35 & 15 & 5 & 1 \\ 35 & 20 & 10 & 4 & 1 \\ 15 & 10 & 6 & 3 & 1 \\ 5 & 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array}\right| .$$

Then, we can specify the set $\{h'\}$ as follows:

$< 1, 2, 3, 6, 7 >, < 1, 2, 3, 6, 8 >, < 1, 2, 3, 6, 9 >,$
$< 1, 2, 4, 6, 7 >, < 1, 2, 4, 6, 8 >, < 1, 2, 4, 6, 9 >,$
$< 1, 2, 5, 6, 7 >, < 1, 2, 5, 6, 8 >, < 1, 2, 5, 6, 9 >,$
$< 1, 3, 4, 6, 7 >, < 1, 3, 4, 6, 8 >, < 1, 3, 4, 6, 9 >,$
$< 1, 3, 5, 6, 7 >, < 1, 3, 5, 6, 8 >, < 1, 3, 5, 6, 9 >,$
$< 1, 4, 5, 6, 7 >, < 1, 4, 5, 6, 8 >, < 1, 4, 5, 6, 9 >,$
$< 2, 3, 4, 6, 7 >, < 2, 3, 4, 6, 8 >, < 2, 3, 4, 6, 9 >,$
$< 2, 3, 5, 6, 7 >, < 2, 3, 5, 6, 8 >, < 2, 3, 5, 6, 9 >,$
$< 2, 4, 5, 6, 7 >, < 2, 4, 5, 6, 8 >, < 2, 4, 5, 6, 9 >,$
$< 3, 4, 5, 6, 7 >, < 3, 4, 5, 6, 8 >, < 3, 4, 5, 6, 9 > .$

Then, we make the indexed family $< \mathcal{G}'_i >$, $1 \leq i \leq 5$ as follows:

$\mathcal{G}'_1 = \{1, 2, 3\}$, $\mathcal{G}'_2 = \{2, 3, 4\}$, $\mathcal{G}'_3 = \{3, 4, 5\}$, $\mathcal{G}'_4 = \{6\}$, $\mathcal{G}'_5 = \{7, 8, 9\}$,

We observe that for the model $S'_U = << \mathcal{G}'_i >$, $1 \leq i \leq 5$; $W(i)$ given in (1.1); $\{h'\}_{\mathcal{G}'_i} >$ there is $\{h'\} = \{h'\}_{\mathcal{G}'_i}$. So, the process of unranking modelling is finished. Since the set $\{h'\}_{\mathcal{G}'_i}$ is symmetric, so we have the corresponding table DDCM.

$$\text{table DDCM}= \left|\begin{array}{ccccc} 6 & 3 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array}\right| .$$

□

The following example of modelling demonstrates the method of imposing the additional requirement W_1 in order to produce the right model S'_R.

Example 5.4.2 *Given is a table $A[m \times n]$. The task is to represent all tables $A'[m \times n]$ obtained from A by permuting its columns using permutations of m whose rank is even.*

The set $\{h'\}$ is a subset of the set $\{h\}_{\mathcal{E}_i}$ for the model $S_R =<< \mathcal{E}_i$ for $n >$, $1 \leq i \leq n$; W given in (1.19),$\{h\}_{\mathcal{E}_i} >$, $<^h$, table DP> . We can specify the set $\{h'\}$ by the rank range $R(\{h'\})$ in $R(h)$. Then, $R(\{h'\}) = (2k, 1)$, where $k = 1, 2, ...,$ $\mid \{h\}_{\mathcal{E}_i} \mid$. This model of the set $\{h'\}$ is very cumbersome, therefore we will replace it with a more suitable one. The set $\{h'\}$ is symmetric therefore, basing on Ranking Theorem, we can build a model S'_R that $\{h'\} = \{h'\}_{\mathcal{E}'_i}$. For making the corresponding unranking model, we observe that the indexed family $< \mathcal{E}'_i >, 1 \leq i \leq n$ is the same as the indexed family $< \mathcal{E}_i >, 1 \leq i \leq n$. So, the structure of the model S'_U

is as follows: $<< \mathcal{E}_i$ for $n >$, $1 \leq i \leq n$; W given in (1.19); W_1; $\{h\}_{\mathcal{E}_i} >$. The goal is to make the requirement W_1.

Observe, for permutations of n whose rank is even, we have $h(n-1) > h(n)$. For making the table D' we note that it is a modified table DP$[n \times n]$. Since for each prefix $h'(1)$, $h'(2)$, ..., $h'(n-2)$, we have exactly one choice function $h' \in \{h'\}_{\mathcal{E}'_i}$. Consequently, that is valid for fixed prefixes $h'(1)$, $h'(2)$, ..., $h'(n-1)$, so the entries D$[j, i] = 1$, $1 \leq j \leq n$, $n-2 \leq i \leq n$. Then, we observe that each D'$[j, i]$ =DCM$[j, i]/2$, $1 \leq j \leq n$, $1 \leq i \leq n-3$, since for each valid prefix $h'(1)$, $h'(2)$, ..., $h'(i)$, $1 \leq i \leq n-3$, we have half of all the possible permutations. For instance, the table D' for $n = 4$ is as follows:

$$\text{D'}= \begin{vmatrix} 3 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 \end{vmatrix}.$$

□

For other tasks of modeling, we can use both modelling the indexed sets and imposing additional requirements W.

5.5 Tabular criterion for congruence of representations

The main motivation for considerations given in this section supplies the Example 5.2.2 presented on page 96. We can have one representation model S_U or a collection of models $< S_U^k >$, $1 \leq k \leq m$. Correspondingly, we can have one model S_R or the collections of models $< S_R^k >$, $1 \leq k \leq m$. The models can be specified as follows: $S_R^k = << \mathcal{G}_i^k >>$, $1 \leq i \leq k$, $W^k; W_1^k$; $\{h\}_{\mathcal{G}_i^k} >$, Table D$^k[n \times k] >$, while $S_R =<< \mathcal{G}_i >>$, $1 \leq i \leq m$, $W; W_1$; $\{h\}_{\mathcal{G}_i} >$, Table D$[n \times m] >$. The goal is to develop a criterion enabling us to state formally that there is a congruence between these two representations.

Proposition 5.5.1 *If there is congruence between the representations S_R and $< S_R^k >$, $1 \leq k \leq m$, then, $D[i, j] = \sum_{k=1}^{m} D^*[i, j]$.*

Proof. If the representations S_R and $< S_R^k >$, $1 \leq k \leq m$, correspond to the same collections of combinatorial objects, then we have to have the mapping η and the collection of mappings $< \eta_k >$ that $\eta : \{h\}_{\mathcal{G}_i} \longrightarrow < S' >$,while $\eta_k : \{h\}_{\mathcal{G}_i^k} \longrightarrow < S'_k >$. Observe, $h =< h(1), h(2), ..., h(m) >$, while $h^k =< h^k(1), h^k(2), ..., h^k(k) >$. The substructure S" is a subset of $S =< A_1, A_2, ..., A_u >$ that contains $k-$elements. Then, $\eta(h(1), h(2), ..., h(m)) = \eta(h(1), h(2), ..., h(k) = \eta_k(h^k)$. It means, we could extend the choice function h^k into a choice function $h'^k =< h^k(1), h^k(2), ...,$

$h^k(k), h'^k(k+1), ..., h'^k(m) >$ that $\eta_k(h^k(1), h^k(2), ..., h^k(k), h'^k(k+1), ..., h'^k(m)) = h^k(1), h^k(2), ..., h^k(k))$. Since that holds for all the choice functions $\{h\}_{\mathcal{G}_i^k}$, so the model $S_R^k = << \mathcal{G}_i^k >, 1 \leq i \leq k, W^k; W_1^k; \{h\}_{\mathcal{G}_i^k} >$ can be replaced with the model $S_R^k = << \mathcal{G}_i^k >, 1 \leq i \leq m, W^k; W_1^k; \{h'\}_{\mathcal{G}_i^k} >$, where $| \mathcal{G}_i^j | = 1, 1 \leq j \leq k$.

Let $D^k[n \times k]$ be the table D for the model S_R^k, while $D'^k[n \times m]$ being the table D for the model $S_R'^k$. Since there is only one choice function h'^k for the prefix h^k, so $D'^k[i, j] = 1, 1 \leq i \leq n$, for $k+1 \leq j \leq m$ and $D'^k[i, j] = D^k[i, j]$ for $1 \leq j \leq k$.

Since each choice function of both representations is to be assigned to exactly one combinatorial object S', so we have to have $| \{h\}_{\mathcal{G}_i} | = \sum_{k=1}^{m} | \{h\}_{\mathcal{G}_i^k} |$. Then, for each $h^k \in \{h\}_{\mathcal{G}_i^k}$ exists $h \in \{h\}_{\mathcal{G}_i}$ that $\eta(h) = \eta_k(h^k)$, since both h and h^k are assigned to an object S'.

Hence, $D[i, j] = \sum_{k=1}^{m} D'^k[i, j]$

□

By similar arguments, we can also show that if there is a congruence between two representations S_R and S_R' regardless what indexed families are, then the conclusion of the above proposition holds.

5.6 Conclusions

We have shown the state of art concerning the models of sets of combinatorial objects. The sets of combinatorial objects are represented by means of S_U and S_R models. The choice function approach is much superior than the classical approach since the models developed here are both homogeneous and flexible enabling us to represent in a number of ways detailed specifications of a given set of combinatorial objects. We have also developed formal methods for testing equivalence of different models and transformations of models. The formal method for diminishing the requirement W to be tested has important practical meaning enabling us to diminish consequently the time of the generation of the concerned set of combinatorial objects. The tabular criterion for congruence of representations enables us to test quickly whether two different models represent the same set of combinatorial objects.

6
Generation models

This and the next chapters are concerning the models S_U and S_R that are reduced and the Q property holds. Therefore, we do not assume at each time that a model used is reduced and the Q property holds.

6.1 The systems $< Methods >$

The ranking and unranking models are input data to be used by computer systems for the generation and for processing combinatorial objects. We develop the basic $< Methods >$ systems containing algorithms (procedures) to be used as routines for sequential, parallel or distributed computations. In fact, we have generally two $< Methods >$ systems those using the unranking models S_U and those using the ranking models S_R. The systems $< M_U >$ and $< M_R >$ are related to S_U and S_R, accordingly. For both systems $< M_U >$ and $< M_R >$ we use the rank concept according to an accepted order $<^h$, usually it is the lexical order.

The system M_U contains only three algorithms $< FIRST, NEXT, LAST >$. The algorithm $FIRST$ takes a given unranking model S_U as input data and it returns the choice function $h \in \{h\}_{\mathcal{G}_i}$, whose rank $R(h) = 1$. The algorithm $NEXT$ takes a given model S_R, the choice function $h \in \{h\}_{\mathcal{G}_i}$ as input data and it returns the choice function $h' \in \{h\}_{\mathcal{G}_i}$ that $R(h') = R(h) + 1$. The algorithm $LAST$ takes the model S_U as input and it returns the choice function $h' \in \{h\}_{\mathcal{G}_i}$, whose rank $R(h') =\mid \{h\}_{\mathcal{G}_i} \mid$ as output.

The system M_R contains algorithms $< RANK, UNRANK, RANGE >$ The algorithm $RANK$ takes a given choice function $h \in \{h\}_{\mathcal{G}_i}$ as input and it returns its rank $R(h)$ as output. The algorithm $UNRANK$ takes an integer k; $1 \leq k \leq |\{h\}_{\mathcal{G}_i}|$ as input and it returns a choice function h whose rank $R(h) = k$ as output. The algorithm $RANGE$ takes a given subset of choice functions $\{h'\} \subset \{h\}_{\mathcal{G}_i}$ as input and it returns the rank range $R(\{h\})$ in $R(h)$ as output. Since the specified methods use a proper table D as input, so we could say that the system M_R contains also algorithms $MAKETABD$ to be used for producing a proper table D. The algorithms $MAKETABD$ were widely discussed in Chapter 3.

The system $< Methods >$ has very important general properties. Namely, we have the general algorithms $FIRST$, $NEXT$, $LAST$, $MAKETABD$, $RANK$, $UNRANK$, $RANGE$ that can be used for any particular model S_U or S_R, respectively. Such general algorithms are rather very complex, nevertheless they enable us to produce output at any particular instance of the model. For specific classes of models we develop algorithms dedicated to them. These algorithms use detailed properties of those classes of models and therefore they are less complex than their more general counterparts. In fact, the classes of models can be organized in a hierarchy represented by a tree whose nodes correspond into the mentioned classes of models. Then, dedicated optimal algorithms are produced for the nodes. For instance, we have general algorithm $NEXT$ that can be used for every given model S_U. We have also algorithm $NEXT$ for all the increasing choice functions and certain specific requirements W_1. Moreover, if the indexed family is the minimal non–deformed family then there is another $NEXT$ algorithm dedicated. The dedicated $NEXT$ algorithms are also developed for special models containing deformed indexed families. Similar hierarchy of other algorithms from the system $< Methods >$ can be produced for other specific classes of models. Existence of the general algorithm and of optimal dedicated algorithms is very convenient for development computer system using object oriented programming. Then, one having specific model S_U or S_R can use directly an algorithm existing in the system or he can modify it in order to make it less complex for the possessed data. Getting low time complexity of the algorithm $NEXT$ applied is especially important since that algorithm is called $O(2^n)$ times for the generation of the sets $\{h\}_{\mathcal{G}_i}$. Then, even reduction of one substitution in the applied algorithm gives reduction of $O(2^n)$ instructions if exhaustive generation is concerned. Complexity of the algorithm $UNRANK$ is also very essential especially if we have to generate a subset of choice functions $\{h'\} \subset \{h\}_{\mathcal{G}_i}$ and the rank range $R(\{h'\})$ in $R(h)$ is z–tuple and $z >> 1$. Time complexity of other algorithms from $< Methods >$ is less important, we use these algorithms a few times during the generation of a given set, if any.

6.2 Implementation notes

We could develop algorithms for the system $< Methods >$ using natural data structures, i.e., those used for the models S_R and S_U. Then, using the given algorithms one could modify them for developing the corresponding programs selecting simultaneously the most suitable data structures. If the programs are to be written in PASCAL or other high level language then, a programmer has to his disposal numerous data structures including set structures. For the algorithms belonging to $< Methods >$ the most common and most important operation is $q \in \mathcal{G}_i$. It is known that the set operation 'q in $\mathcal{G}$ ' takes three times more time than the corresponding tabular operation $G[q] = TRUE$ that implements 'q in $\mathcal{G}$ '. Therefore, implementation of indexed families is made mainly in order to get complexity of testing whether $q \in \mathcal{G}_i$ as low as possible. Moreover, we select a most suitable implementation in order to find a minimal value $q_i \in \mathcal{G}_i$ that for a given value $h(i-1) \in \mathcal{G}_i$ the value q_i satisfies the requirement W, so we can have $h(i) = q_i$.

The implementations of indexed families: (6.1)

* Each indexed set $\mathcal{G}_i$ is ordered increasingly. An indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$, can be represented by two tables of integers $Q[n \times m]$ and $Y[n \times m]$. Let q_1 and q_2 belong to $\mathcal{G}_i$, while y_1 and y_2 being the rank of q_1 and q_2 in $\mathcal{G}_i$, respectively. Suppose, $q_1 < q < q_2$ and $q \notin \mathcal{G}_i$, then $y_2 = y_1 + 1$. The entries of the tables Q and Y are defined as follows: $Q[q_1, i] = y_1$, $Q[q, i] = y_1$, $Q[q_2, i] = y_2$, while $Y[y_1, i] = q_1$, $Y[y_2, i] = q_2$. We emphasize the given assignments do not correspond to $x(\mathcal{G}_i)$ neither into $ord(q, \mathcal{G}_i)$, unless the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$ is the maximal or the minimal non-deformed indexed family. We can use for an algorithm both tables Q and Y or only one of them.

** An indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$ can be represented by Boolean tables $G[n \times m]$, where $G[j, i] = TRUE$ corresponds to $j \in \mathcal{G}_i$, while $G[j, i] = FALSE$ corresponds to $j \notin \mathcal{G}_i$. Moreover, we use additionally the table Y as given above. Then, $x(\mathcal{G}_i) \in \mathcal{G}_i$ corresponds to $G[Y[x, i], i] = TRUE$.

*** An compact indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$, i.e., $\mathcal{G}_i = \{\min_i, \min_i + 1, ..., \max_i\}$ can by represented by an array of integers $AG[2 \times m]$ that $AG[1, i] = \min_i$, while $AG[2, i] = \max_i$. Then, "q in $\mathcal{G}_i$" is represented by the following Boolean expression "$((q <= AG[2, i])\ AND\ (q >= AG[1, i])) = TRUE$" or simply by "$((q <= AG[2, i])\ AND\ (q >= AG[1, i]))$".

**** The canonical representation given by tables GI and X representing the minimal non-deformed indexed family $< \mathcal{G}_i^{\min} >$, $1 \leq i \leq m$, for the given $< \mathcal{G}_i >$, $1 \leq i \leq m$. The value $x = X[q, i]$ equals $ord(q, \mathcal{G}_i^{\min})$, while $GI[x, i]$ equals to $x(\mathcal{G}_i)$.

The given data structures proposed for representing indexed families enable us to diminish the time of performance of the operation "$q \in \mathcal{G}_i$".

Moreover, we can easily produce the next and reasonable value q that is to be tested.

6.3 The algorithm $NEXT$

Given is unranking model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; W_1; \{h\}_{\mathcal{G}_i} >$, the requirements W and W_1 are arbitrary, the indexed family $< \mathcal{G}_i >, 1 \leq i \leq m,$ is arbitrary except the model is reduced and the Q property holds. The algorithm $NEXT$ takes a choice function h as input and returns the choice function h' as output that $R(h') = R(h)+1$, however we do not know neither evaluate the ranks $R(h)$ and $R(h')$. The basic algorithm $NEXT$ concerning any model S_U is a part of the algorithm "choice function" given in [19]. We rewrite now the part of the algorithm "choice function" that corresponds to the general algorithm $NEXT$.

Algorithm $NEXT$ (general)

Input: A given reduced model S_U, the table IA that the value $IA[i]$ represents $h(i)$.

Output: The table $IA[i]$ representing the custom form of the next choice function.

Method: At the beginning the table IA represents the custom form of the previous choice function

1. $i \longleftarrow m$;

2. **while** ($IA[i]$ could not be increased for fixed $< IA[1], IA[2], ..., IA[i-1] >$) **do**

2.1. $i \longleftarrow i - 1$;

3. $IA[i] \longleftarrow \min\{j : j \in \mathcal{G}_i$ and $j > IA(i)$ and j satisfies $W \wedge W_1\}$;

4. **for** $p \longleftarrow i + 1$ **to** m **do**

4.1. $IA[p] \longleftarrow \min\{j : j \in \mathcal{G}_i$ and j satisfies $W \wedge W_1\}$;

5. **return** IA.

Asymptotic complexity of this algorithm is changing from $O(n.m)$ to $O(m)$ depending on the indexed families, on the requirements W and W_1 and on the implementation. The process of production h' depends also strongly on h. For the best case, we have also complexity $\Omega(1)$ since updating only the value $h(m) = IA[m]$ can be enough for producing the next choice function. In fact, for numerous practical models S_U the complexity $\Theta()$ is independent from m, then we have usually $\Theta(n)$ or $\Theta(1)$. Frankly speaking the number of runs of the step 2. on average was only two for numerous practical models tested. That result can be justified analytically as follows.

Let $< \mathcal{G}_i >, 1 \leq i \leq t$, be a subfamily of the indexed family $< \mathcal{G}_i >, 1 \leq i \leq m$, while the requirements W and W_1 being unchanged, $t < m$.

Suppose, the number of choice functions $\{h\}_t$ for a given t is two times greater than the number of choice functions $\{h\}_{t-1}$ for $t-1$. Then, for each prefix $h \in \{h\}_{t-1}$, we have two choice functions h_1 and h_2 belonging to the set $\{h\}_t$. We have $R(h_2) = R(h_1) + 1$. Then, we need to modify only value $h(t)$ in order to produce h_2 from h_1. Hence, the number of choice functions $\{h\}_t/2$ requires only one run of the step 2. to be produced. Similarly, the number of choice functions $\{h\}_t/4$ requires two runs of the step 2. and so on. Then, the average number of runs of the step 2. equals $\frac{\{h\}_t/2+2.\{h\}_t/4+3.\{h\}_t/8+....+m\{h\}_t/2^m}{\{h\}_t} < 2$. So, for the most of the models S_U practically used, we have $\Theta(n)$ or $\Theta(1)$. We have to emphasize that the above result does not concern all the possible models S_U. For instance if $|\ \mathcal{G}_1\ | > 1$ and $|\ \mathcal{G}_2\ | = |\ \mathcal{G}_3\ | = ... = |\ \mathcal{G}_m\ | = 1$, then the average number of runs of the step 2. is m.

Time complexity of the algorithm $NEXT$ is also dependent on particular implementation used, especially representation of the indexed family is important. The form of the requirements W and W_1 used and their implementation are also effecting complexity of the general algorithm. We will present now selected variants of this general algorithm, by variants we mean also different implementations concerning the data structures used. The whole number of specific variants of the given general algorithm is really huge, so optimal selection requires examination of the detailed properties of the model S_U. Therefore, our goal is not a presentation of all the possible variants of the general algorithm but rather presentation of mechanisms that can be used for production a most suitable variant for particular circumstances.

6.3.1 For increasing functions

Given is the unranking model $S_U = << \mathcal{G}_i >, 1 \leq i \leq m; W$ given in (1.7); W_1; $\{h\}_{\mathcal{G}_i} >$, the requirement W_1 is arbitrary, the indexed family $< \mathcal{G}_i >, 1 \leq i \leq m$ is arbitrary. Implementation: the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$, is implemented as Boolean table $G[n \times m]$, moreover, $AG[2 \times m]$ is the table of maximal and minimum values $\max(\mathcal{G}_i)$ and $\min(\mathcal{G}_i)$ that $AG[i, 1] = \min(\mathcal{G}_i)$ and $AG[i, 2] = \max(\mathcal{G}_i)$, see (6.1), page 119.

Algorithm NEXTI (Increasing choice function)

Input: The model S_U, the table IA represents current $h(i)$, see algorithm *NEXT*.

Output: The table $IA[i]$ representing the custom form of the next choice function.

Method:

1. $i \longleftarrow m$;

2.1. **while** $IA[i] = AG[2, i]$ **do**

2.1.1. $i \longleftarrow i - 1$;

3. $IA[i] \longleftarrow IA[i] + 1$;
4. **for** $p \longleftarrow i$ **to** m **do**
4.1. **while** $G[IA[p], p] = FALSE$ or W_1 does not hold **do**
4.1.1 $IA[p] \longleftarrow IA[p] + 1$;
4.2. $IA[p+1] \longleftarrow IA[p+1] + 1$;
5. **return** IA.

Asymptotic complexity of the above algorithm is $O(n+m)$, where $n = \max(\mathcal{U}) - \min(\mathcal{U})$. Since $n > m$, so we have $O(n)$. One can prove it by observing that the step 4.1. can be performed at most n times in the course of running the algorithm. Time complexity is much reduced in comparing with the algorithm $NEXT$. We reached it by proper implementation of the functions max and min. If the requirement W_1 is of the form $h(i)\mathrm{RE}[h(i-1), i]$, then we can diminish the number of runs of the step 4.1. It can be obtained by diminishing the number of cases, when $G[IA[p], p] = FALSE$ and/or by diminishing the number of cases when W_1 does not hold.

Given is the model S_U as specified in the previous case except n is big and the indexed sets $\mathcal{G}_i$ are not dense the requirement W_1 is arbitrary. In order to diminishing the number of the tests $G[IA[p], p] \stackrel{?}{=} FALSE$ we use additionally the tables Q and Y for representing the indexed family, see (6.1), p. 119. Then, we have the following algorithm $NEXT$IRG.

Algorithm $NEXT$IRG (Increasing choice function with Reduced number of test $G[IA[p], p] \stackrel{?}{=} FALSE$)

Input: The model S_U, IA.

Output: IA.

Method: The tables $Q[n \times m]$ and $Y[n \times m]$ are used for representing the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$.

1. $i \longleftarrow m$;
2. **while** $IA[i] = AG[2, i]$ **do**
2.1. $i \longleftarrow i - 1$;
3. $y \longleftarrow Q[IA[i], i] + 1$;
4. $IA[i] \longleftarrow Y[y, i]$;
5. **for** $p \longleftarrow i$ **to** m **do**
5.1. **while** W_1 does not hold for $IA[p]$ **do**
5.1.1. $y \longleftarrow y + 1$;
5.1.2. $IA[p] \longleftarrow Y[y, p]$;
5.2. **if** $Q[IA[p]+1, p+1] = Q[IA[p], p+1]$ **then** $y \longleftarrow Q[IA[p]+1, p+1] + 1$
else $y \longleftarrow Q[IA[p]+1, p+1]$;
5.3. $IA[p+1] \longleftarrow Y[y, p+1]$;
6. **return** IA.

We have assertion that the step 5.1. is performed only for values $IA[p] \in \mathcal{G}_i$, since $IA[p+1]$ evaluated in the step 5.3. takes only values that belong to $\mathcal{G}_i$. It means that the asymptotic complexity of the step 5.1. is dependent on the average cardinal $\mid \mathcal{G}_i \mid$. Then, we have $O(\mid \mathcal{G}_i \mid +m)$ and $\Omega(1)$. Using similar data structures for representing COMBINED W, we can produce an algorithm for diminishing the number of questions whether the requirement W_1 is satisfied or not. Observe, in order to get a simple structure of the above algorithm we accept performance of steps 5.2. and 5.3. for $m+1$ that is useless. Changing the range of the loop 5. to '**for** $p \longleftarrow i$ **to** $m-1$ **do**' and repeating the steps 5., 5.1., and 5.1.1. for $i = m$, we get minimum time complexity of the above algorithm.

If the model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W$ given in (1.7); $\{h\}_{\mathcal{G}_i} >$ does not contain the requirement W_1, then the following algorithm $NEXT$IVW that is a simplification of the above given algorithm could be used.

Algorithm $NEXT$IVW (Increasing choice function with Void requirement W_1)

Input: The model S_U, IA.

Output: IA.

Method: The tables $Q[n \times m]$ and $Y[n \times m]$ are used for representing the indexed family $< \mathcal{G}_i >, 1 \leq i \leq m$.

1. $i \longleftarrow m$;
2. **while** $IA[i] = AG[2, i]$ **do**
2.1. $i \longleftarrow i - 1$;
3. $y \longleftarrow Q[IA[i], i] + 1$;
4. $IA[i] \longleftarrow Y[y, i]$;
5. **for** $p \longleftarrow i$ **to** $m - 1$ **do**
5.1. **if** $Q[IA[p] + 1, p + 1] = Q[IA[p], p + 1]$ **then** $y \longleftarrow Q[IA[p] + 1, p + 1] + 1$
else $y \longleftarrow Q[IA[p] + 1, p + 1]$;
5.2. $IA[p + 1] \longleftarrow Y[y, p + 1]$;
6. **return** IA.

We have assertion that for evaluation of each value $IA[i]$ only a fixed number of steps is needed. Therefore, we have $O(m)$ and $\Omega(1)$. Nevertheless, time complexity of the step 5.1. is rather big. The next considerations are made in order to diminish time complexity if the model S_U satisfies additional assumptions.

Given is the model $S_U =<< \mathcal{R}_i >, 1 \leq i \leq m; W$ given in (1.7); $\{h\}_{\mathcal{R}_i} >$, while the indexed family $< \mathcal{R}_i >, 1 \leq i \leq m$, is defined in page 20, $\mid \mathcal{R}_i \mid = n$ and it is the maximal indexed family.

Algorithm $NEXT$IMAX (Increasing function of MAXimal families)

Input: The model S_U, the table $IA[m+1]$, the first $m-$ entries represent current $h(i)$, see algorithm $NEXT$.

Output: The table $IA[i]$ representing the custom form of the next choice function.

Method: We use only the table AG as the representation of the indexed family $< \mathcal{R}_i >$, $1 \leq i \leq m$.

1. $i \longleftarrow m$;
2. **while** $IA[i] = AG[2, i]$ **do**
2.1. $i \longleftarrow i - 1$;
3. $IA[i] \longleftarrow IA[i] + 1$;
4. **for** $p \longleftarrow i$ **to** $m - 1$ **do**
4.1. $IA[p + 1] \longleftarrow IA[p] + a_p + 1$;
5. **return** IA.

Asymptotic complexity of this algorithm is $O(m)$ since each value $IA[i]$ is updated at most once. If we replace the step 4.1. with "4.1. $IA[p] \longleftarrow IA[p] + 1$"; then we get the well known algorithm for the generation of the next combination of $m-$elements out of n [15]. One can observe it by noting that for all the elements $a_i = 1$ the indexed family $< \mathcal{R}_i >$, $1 \leq i \leq m$, becomes the family $< \mathcal{A}_i >$, $1 \leq i \leq m$ and the requirement W is the same as specified in (1.1).

Given is the unranking model $S_U =<< \mathcal{P}_i >$, $1 \leq i \leq m; W$ given in (1.7); $\{h\}_{\mathcal{P}_i} >$ that $\min(\mathcal{P}_i) > \max(\mathcal{P}_{i-1}) + a_i$, $2 \leq i \leq m$. Suppose, n is big and the indexed sets are not dense.

Algorithm $NEXT$ISEP (Increasing function for SEParated indexed sets)

Input: The model S_U, the table $IA[m+1]$, the first $m-$ entries represent current $h(i)$, see algorithm $NEXT$.

Output: The table $IA[i]$ representing the custom form of the next choice function.

1. $i \longleftarrow m$;
2. **while** $IA[i] = AG[i, 2]$ **do**
2.1. $i \longleftarrow i - 1$;
3. $y \longleftarrow Q[IA[i], i] + 1$;
4. $IA[i] \longleftarrow Y[y, i]$;
5. **for** $p \longleftarrow i + 1$ **to** m **do**
5.1. $IA[p] \longleftarrow AG[1, p]$;
6. **return** IA.

Asymptotic complexity of the above algorithm is $O(m)$.

Given is the unranking model S_U as specified above. Moreover, the indexed sets are compact. Then, we have the following algorithm.

Algorithm $NEXT$ICOM (Increasing function for COMpact sets)

Input: The model S_U, the table $IA[m+1]$, the first $m-$ entries represent current $h(i)$, see algorithm $NEXT$.

Output: The table $IA[i]$ representing the custom form of the next choice function.

1. $i \longleftarrow m$;
2. **while** $IA[i] = AG[2,i]$ **do**
2.1. $i \longleftarrow i-1$;
3. $IA[i] \longleftarrow IA[i]+1$;
4. **for** $p \longleftarrow i+1$ **to** m **do**
4.1. $IA[p] \longleftarrow AG[1,p]$;
5. **return** IA.

This algorithm seems to be the simplest for the generation of the next increasing choice function. Its asymptotic complexity is $O(m)$, while time complexity is the lowest possible.

The mechanism shown in this section applied for diminishing complexity of searching for a next element to be tested can be adopted for numerous requirements W.

6.3.2 For monotonic functions

For each variant of the general algorithm concerning the increasing choice functions, we can produce its counterpart dedicated to the monotonic choice functions. All the mechanisms are similar except substitution $IA[p+1] \longleftarrow IA[p]+1$ we replace with $IA[p+1] \longleftarrow IA[p]$. The other evaluations of $IA[i]$ treated similarly.

6.3.3 For bijections

Given is the unranking model $S_U =<< \mathcal{S}_i >,\ 1 \leq i \leq m; W$ given in (1.19); W_1; $\{h\}_{\mathcal{S}_i} >$, where $\mathcal{S}_i \subset \mathcal{U}$. For development of the algorithms $NEXT$ that are to be used for the generation of bijections the most important arrangement concerns testing satisfiability of the requirement W given in (1.19). We use one–dimensional array of Boolean variables B which size equals $\max(\mathcal{U})$. Initially, $B[q] = TRUE$ if $q \in \mathcal{U}$. Then, we have the following algorithm $NEXT$B.

Algorithm $NEXT$B (algorithm $NEXT$ for choice Bijections)

Input: The model S_U, the table IA represents current $h(i)$, the table B.

Output: The table $IA[i]$ representing the custom form of the next choice function.

Method: At the beginning we have $B[q] = TRUE$, if and only if, $q \in \mathcal{U}$ and $q \neq h(i)$ for every $i \in \{1, 2, ..., m\}$.

1. $i \longleftarrow m$;
2. **while** $IA[i] = AG[2,i]$ **do**
2.1. $B[IA[i]] \longleftarrow TRUE$;
2.2. $i \longleftarrow i-1$;

3. $IA[i] \longleftarrow IA[i]+1$;
3.1. **while** $B[IA[i]] = FALSE$ or W_1 does not hold **do**
3.1.1. $IA[i] \longleftarrow IA[i]+1$;
4. $B[IA[i]] \longleftarrow FALSE$;
5. **for** $p \longleftarrow i+1$ **to** m **do**
5.1. $IA[p] \longleftarrow AG[1,p]$;
5.2. **while** $B[IA[p]] = FALSE$ or W_1 does not hold **do**
5.2.1. $IA[p] \longleftarrow IA[p]+1$;
5.3. $B[IA[i]] \longleftarrow FALSE$;
6. **return** IA, B.

Asymptotic complexity of the algorithm $NEXY$B is $O(n.m)$ since for each value i one can need to perform $O(n)$ searches for a suitable value $IA(i)$.

Given is the unranking model $S_U =<< \mathcal{S}_i >, 1 \leq i \leq m; W$ given in (1.19); $\{h\}_{\mathcal{S}_i} >$, $\mathcal{S}_i \subset \mathcal{U}$, $\mid \mathcal{U} \mid$ and the indexed sets $\mathcal{S}_i$ are not dense, in other words $\max(\mathcal{U}) >> \mid \mathcal{U} \mid$. In order to diminish the number of runs of the loops "**while**", we make the similar arrangement as was given for the algorithm $NEXT$IRG.

Algorithm $NEXT$BNT (choice Bijection for Not–Thick indexed sets)

Input: The model S_U, the table IA, the table B, the tables Q, Y and AG.

Output: The table $IA[i]$ representing the custom form of the next choice function.

Method: At the beginning we have $B[q] = TRUE$, if and only if, $q \in \mathcal{U}$ and $q \neq h(i)$ for every $i \in \{1, 2, ..., m\}$.

1. $i \longleftarrow m$;
2. **while** $IA[i] = AG[2,i]$ **do**
2.1. $B[IA[i]] \longleftarrow TRUE$;
2.2. $i \longleftarrow i-1$;
3. $y \longleftarrow Q[IA[i], i]+1$;
4. $IA[i] \longleftarrow Y[y, i]$;
5. **while** $B[IA[i]] = FALSE$ **do**
5.1. $y \longleftarrow y+1$;
5.2. $IA[i] \longleftarrow Y[y, i]$;
6. $B[IA[i]] \longleftarrow FALSE$;
7. **for** $p \longleftarrow i+1$ **to** m **do**
7.1. $IA[p] \longleftarrow AG[1,p]$;
7.2. $y \longleftarrow 1$;
7.3. **while** $B[IA[p]] = FALSE$ **do**
7.3.1. $y \longleftarrow y+1$;
7.3.2. $IA[p] \longleftarrow Y[y, i]$;
7.4. $B[IA[p]] \longleftarrow FALSE$;
8. **return** IA, B.

The steps 5. and 7.3. can be performed at most $i-1$ times for each value i and p, respectively. Therefore, the asymptotic complexity of the above algorithm is $O(m^2)$. Since $n > m$, so the above algorithm is less complex than the previous algorithm $NEXT$B.

Given is the unranking model $S_U =<<\mathrm{U}_i >, 1 \leq i \leq m; W$ given in (1.19); $\{h\}_{\mathcal{U}_i} >$, $|\mathrm{U}_i| = m$, $\mathrm{U}_i = \mathcal{U}$, were $\max(\mathrm{U}_i) > |\mathrm{U}_i|$, so the sets U_i are not dense. Then, we have the algorithm $NEXT$BPER using a stock.

Algorithm $NEXT$BPER (choice Bijection or PERmutation)

Input: The model S_U, the table IA,

Output: The table $IA[i]$ representing the custom form of the next choice function.

Method:

1. $i \longleftarrow m-1$;
2. push $IA[m]$ on the stack;
3. **while** $IA[i] < IA[i+1]$ **do**
3.1. push $IA[i]$ on the stack;
3.2. $i \longleftarrow i-1$;
4. **for** $p \longleftarrow m$ **to** i **do**
4.1. pull $IA[p]$ from the stack;
5. **return** IA.

The asymptotic complexity of the algorithm $NEXT$BPER is $O(m)$, since the steps 3. and 4. can run at most $m-$times. This algorithm is well known for producing permutations [53].

A proper algorithm for producing bijections piecewise increasing we produce combining a proper version of the algorithm $NEXT$I with the corresponding version $NEXT$B. Then, the set of indexes $I = \{1, 2, ..., m\}$ is partitioned in two parts I_B and I_I for index i belonging to I_B, we have to test satisfiability of the requirement given in (1.19), while for the indexes belonging to I_I the requirement (1.1) is to be tested. Given is the model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W$ given in (1.19) for $i \in I_B$ and W given in (1.1) for $i \in I_C$; W_1; $\{h\}_{\mathcal{G}_i} >$, where $S_i \subset \mathcal{U}$. The other arrangements are the same as for the algorithms $NEXT$B and $NEXT$I.

Algorithm $NEXT$BI (choice Bijection piecewise Increasing)

Input: The model S_U, the table IA represents current $h(i)$, the tables B and I_C.

Output: The table $IA[i]$ representing the custom form of the next choice function.

Method: Similar as given for the algorithm $NEXT$B. Moreover, we make one−dimensional Boolean table IC whose size is m. If $i \in I_C$ then $IC[i] = TRUE$ else $IC[i] = FALSE$.

1. $i \longleftarrow m$;

2. **while** $IA[i] = AG[2,i]$ **do**
2.1. $B[IA[i]] \longleftarrow TRUE$;
2.2. $i \longleftarrow i-1$;
3. $IA[i] \longleftarrow IA[i]+1$;
3.1. **while** $B[IA[i]] = FALSE$ or W_1 does not hold **do**
3.1.1. $IA[i] \longleftarrow IA[i]+1$;
4. $B[IA[i]] \longleftarrow FALSE$;
5. **for** $p \longleftarrow i+1$ **to** m **do**
5.1. **if** $IC[i] = TRUE$ **then** $IA[p] \longleftarrow IA[p-1]+1$ **else** $IA[p] \longleftarrow AG[1,p]$;
5.2. **while** $B[IA[p]] = FALSE$ or W_1 does not hold **do**
5.2.1. $IA[p] \longleftarrow IA[p]+1$;
5.3. $B[IA[i]] \longleftarrow FALSE$;
6. **return** IA, B.

Observe, the above algorithm differs only from the algorithm $NEXT$TB by step 5.1. The algorithms $NEXT$ICOM and $NEXT$ISEP can also be used without any modifications for producing choice bijections if the indexed families of the corresponding models S_U applied satisfy the specified requirements for these algorithms. That illustrates the statement saying that the same set of choice functions can represent both a set of partitions and a set of compositions.

6.3.4 For hierarchical systems

Given is a hierarchical unranking representation model

$$\begin{cases} \textit{System for } J \textit{ generation} \\ << \mathcal{G}_j >,\ j \in J; W^{\ f};\ \{f\}_{\mathcal{G}_j} > \\ << \mathcal{G}_i >, 1 \le i \le m; W;\ \{h\}_{\mathcal{G}_i} > \end{cases} \quad \text{as specified in (1.22).}$$

The models $<< \mathcal{G}_i >, 1 \le i \le m; W;\ \{h\}_{\mathcal{G}_i} >$ and $<< \mathcal{G}_j >,\ j \in J; W^{\ f};\ \{f\}_{\mathcal{G}_j} >$ are reduced and the Q property holds for each model. The requirement W^f is the restriction of the requirement W and the set $f \in \{f\}_{\mathcal{G}_j}$ makes a partition on the set of choice functions $\{h\}_{\mathcal{G}_i}$. We have here two tasks:

(*i*) generate the next choice function f for the model $<< \mathcal{G}_j >,\ j \in J; W^{\ f};\ \{f\}_{\mathcal{G}_j} >$;

(*ii*) generate the next choice function h^f that is an extension of f.

Generation of the choice function f does not create any problem since the general algorithm $NEXT$ or any its version dependent on specific properties of the model $<< \mathcal{G}_j >,\ j \in J; W^{\ f};\ \{f\}_{\mathcal{G}_j} >$ can be applied directly. We use denotation $NEXT(f)$ in order to emphasize that the next choice function $f \in \{f\}_{\mathcal{G}_j}$ is to be generated. The generation of each next choice function h^f that is an extension of f creates additional problems. We use denotation $NEXT(h^f, f)$ in order to emphasize that the target is the generation of each next choice function h^f being an extension of f. Then, holding

the Q property for the model $<< \mathcal{G}_i >, 1 \leq i \leq m; W; \{h\}_{\mathcal{G}_i} >$ is not sufficient for making the backtracking-less algorithm $NEXT(h^f, f)$. In order to make assertion that the algorithm $NEXT(h^f, f)$ could be backtracking-less, we have to modify the model $<< \mathcal{G}_i >, 1 \leq i \leq m; W; \{h\}_{\mathcal{G}_i} >$ for each choice function $f \in \{f\}_{\mathcal{G}_j}$. This modification can be done by restricting the indexed sets $\mathcal{G}_i$ into the sets $\mathcal{G}_i^f$ and/or by imposing additional requirement W_j^Q. Then, for each choice function $f \in \{f\}_{\mathcal{G}_j}$, we have the model $<< \mathcal{G}_i^f >, 1 \leq i \leq m; W; W_f^Q; \{h^f\}_{\mathcal{G}_i} >$, where $\mathcal{G}_i^f \subseteq \mathcal{G}_i$ for every $i \in \{1, 2, ..., m\}$. The requirement W_f^Q has the same meaning as the requirement W_1 from the point of view of the generation of the choice functions h^f, i.e., we have to test satisfiability of $W \wedge W_f^Q$ for each next $h^f \in \{h^f\}$. Making the requirement W_f^Q is a matter of individual considerations, nevertheless if the requirement W specifies increasing choice functions or monotonic choice functions or bijections, then we can follow the general results concerning satisfiability of the Q property.

For instance, if W is given in (1.1), then W^f is specified as follows: $h(j_p) \geq h(j_{p-1}) + (j_p - j_{p-1} + 1)$. For creating the requirement W_f^Q or for restricting the sets $\mathcal{G}_i$ into the sets $\mathcal{G}_i^f$, we have to follow Proposition 1.2.1. Then, for every i that $j_{p-1} > i > j_p$, we have to have $h(i) \leq h(j_p) - (j_p - i + 1)$.

If the set $\{h\}_{\mathcal{G}_i}$ is a set of bijections, then W^f is a proper restriction of the requirement (1.19) into the set of indexes J. Then, the requirement W_f^Q can be specified as follows: $h(i) \neq h(j_1)$, $h(i) \neq h(j_2)$, ..., $h(i) \neq h(j_r)$. Observe, the requirement W_f^Q concerns all bijections, so it must be followed if we generate bijections piecemeal increasing or if we generate a set of increasing choice functions, since each increasing choice function is a bijection.

Algorithm $NEXT(f)$

Input: The hierarchical model specified in (1.22), the requirement W^f, current choice function f.

Output: The next choice function $f \in \{f\}_{\mathcal{G}_j}$

Method: We represent function f by the table IA that size is m since the partial choice function f is to be extended up to the choice functions h^f. Nevertheless, only elements LA indexed by $j_p \in J$ are to be updated.

1. $j_p \longleftarrow j_r$;
2. **while** $IA[j_p] = \max(\mathcal{G}_{j_p})$ such that W^f holds **do**
2.1. $p \longleftarrow p - 1$;
3. $IA[j_p] \longleftarrow \min\{q : q \in \mathcal{G}_{j_p}$ and $q > IA(j_p)$ and q satisfies $W^f\}$;
4. **for** $s \longleftarrow p + 1$ **to** r **do**
4.1. $IA[j_s] \longleftarrow \min\{q : q \in \mathcal{G}_{j_s}$ and q satisfies $W^f\}$;
5. **return** IA.

Asymptotic complexity of this algorithm is $O(\mid J \mid . \max(\mid \mathcal{G}_{j_p} \mid))$

The general algorithm $NEXT(h^f, f)$ for the generation of each next $h^f \in \{h^f\}$ is given as follows.

Algorithm $NEXT(h^f, f)$

Input: The hierarchical model specified in (1.22), the requirement W_f^Q, a choice function f.

Output: The next choice function $h \in \{h^f\}$

Method: Each choice function $h \in \{h^f\}$ is an extension of a choice function f.

1. $i \longleftarrow m$;
2. **while** $i \in J$ or $IA[i] = \max(\mathcal{G}_i^f)$ **do**
2.1. $i \longleftarrow i - 1$;
3. $IA[i] \longleftarrow \min\{q : q \in \mathcal{G}_i^f$ and $q > IA(i)$ and q satisfies $W \wedge W_f^Q\}$;
4. **for** $p \longleftarrow i + 1$ **to** m **do**
4.1. **if** $p \notin J$ **then** $IA[p] \longleftarrow \min\{q : q \in \mathcal{G}_i^f$ and q satisfies $W \wedge W_f^Q\}$;
5. **return** IA.

Asymptotic complexity of the algorithm $NEXT(h^f, f)$ is $O(n.m)$, so it is the same as for the algorithm $NEXT$. We can follow the general technics for diminishing the asymptotic complexity of the proper version of the algorithm $NEXT$ for creating the proper version of the algorithm $NEXT(h^f, f)$ or $NEXT(f)$.

6.4 The algorithms $FIRST$ and $LAST$

The algorithm $FIRST$ applicable for any given model S_U is as follows:

Algorithm $FIRST$

Input: A given reduced model S_U.

Output: The choice function $h_{first} \in \{h\}_{\mathcal{G}_i}$ whose rank $R(h)$ equals 1.

1. **for** $i \longleftarrow 1$ **to** m **do**
1.1. $h_{first}(i) \longleftarrow \min(\mathcal{G}_i)$;

Using these general $FIRST$ algorithm, we could make its different implementations depending on realizing the functions min() and max(). For instance, if we use the model S_R as specified in (1.2), then the step 1.1. can be replaced with 1.1. $h_{first}(i) \longleftarrow i$. Several different variants of the general $FIRST$ algorithm suitable for dedicated types of models S_U could be developed. Nevertheless, no variant of the general $FIRST$ algorithm would have any practical meaning since we run the algorithm $FIRST$ only once in the course of the generation of a set of choice functions, so its complexity does not matter.

If we replace the step 1.1. of the algorithm $FIRST$ with 1.1. $h_{last}(i) \longleftarrow \max(\mathcal{G}_i)$, then the general algorithm $LAST$ is obtained that produces the choice function h_{last}. The algorithm $LAST$ runs also only once in the course of the generation, so its optimization has no practical meaning. Nevertheless, the concept of usage of h_{last} leads to bulky computations since we have to test whether each generated choice function h equals h_{last}, in order to break the generation process. That gives necessity of additional $O(2^n)$ comparisons. The problem is even more important if testing whether h equals h_{last} requires performance of several instructions. In order to diminish this additional computations, we can change the concept of the algorithm $LAST$. It can produce not the last choice function h_{last} but only a message that the choice function generated currently is h_{last}. Then, for proper selection of the detailed output of $LAST$, we have to examine the proper algorithm $NEXT$. If the algorithms $NEXT$ given in the previous sections were returning the value i, then $i = 0$ could be used as the stop condition for the generation of the next choice function since $i = 0$ can happen only if the next choice function to be generated does not exist. Alternative is usage of $LAST$ that counts the number of calls of the algorithm $NEXT$. That method can be used only if we can easily count the cardinal $|\{h\}_{\mathcal{G}_i}|$, that happens if the set $|\{h\}_{\mathcal{G}_i}|$ is symmetric. If we have asymmetric sets and very irregular requirements W and W_1 evaluation of $|\{h\}_{\mathcal{G}_i}|$ in advance can be a difficult task. For numerous applications especially if exhaustive generation is not necessary the stop conditions are coming from the external applications. Then, the algorithm $LAST$ can be developed using these external stop conditions and the internal conditions mentioned earlier can be postponed. For instance, as the stop conditions, we can select the time of running algorithms or meeting requested conditions by currently generated object or so on.

Consider now the hierarchical model specified in (1.22). The algorithms $FIRST(f)$ and $FIRST(h^f, f)$ that are the counterparts of the algorithms $NEXT(f)$ and $NEXT(h^f, f)$ are given as follows.

Algorithm $FIRST(f)$

Input: The hierarchical model specified in (1.22), the requirement W^f, current choice function f.

Output: The next choice function $f \in \{f\}_{\mathcal{G}_j}$

Method: For the representation of f, see the algorithm $NEXT(f)$.

1. **for** $i \longleftarrow 1$ **to** m **do**

1.1. **if** $i \in J$ **then** $f(i) \longleftarrow \min\{q : q \in \mathcal{G}_i$ and W^f is satisfied$\}$;

Algorithm $FIRST(h^f, f)$

Input: The hierarchical model specified in (1.22), the requirement W_f^Q, a choice function f.

Output: The next choice function $h \in \{h^f\}$
1. **for** $i \longleftarrow 1$ **to** m **do**
1.1. **if** $i \notin J$ **then** $h(i) \longleftarrow \min\{q : q \in \mathcal{G}_i^f$ and W_f^Q is satisfied$\}$;

Similarly, we can specify the algorithms $LAST(h^f, f)$ and $LAST(f)$ that are the counterparts of the given algorithms $FIRST(f)$ and $FIRST(h^f, f)$.

6.5 The algorithm *UNRANK*

Given is the ranking model $S_R = < S_U, <^h$, table D$>$, where $S_U = << \mathcal{G}_i >, 1 \leq i \leq m; W; W_1; \{h\}_{\mathcal{G}_i} >$ and $<^h$ is the lexical order.

Algorithm UNRANK (general)
Input: The model S_R that the indexed family $< \mathcal{G}_i >, 1 \leq i \leq m;$ is reduced the Q property holds, the rank $R(h)$.
Output: The table $IA[i]$ representing the custom form of the choice function $h(i)$.
Method: We use the tables GI for representing the function $ord(q, \mathcal{G}_i^{\min})$, see (6.1), p.119.
1. **for** $i \longleftarrow 1$ **to** m **do**
1.1. $x \longleftarrow 1$;
1.2. $IA[i] \longleftarrow GI[1, i]$;
1.3. **while** $R(h) >$D$[x, i]$ **do**
1.3.1. **if** $IA[i] \in \mathcal{G}_i$ and $W \wedge W_1$ holds for $IA[i]$ **then** $R(h) \longleftarrow R(h)-D[x, i]$;
1.3.2. $x \longleftarrow x + 1$;
1.3.3. $IA[i] \longleftarrow GI[x, i]$;
2. **return** IA;

Asymptotic complexity of this algorithm is $O(n.m)$. For any model S_U used and for any variant of this algorithm complexity for the best case is $\Omega(m)$. Therefore, we would always have complexity $O(\mu(n).m)$ for any variant of the above algorithm, where $O(\mu(n)) = n$ and $\Omega(\mu(n)) = 1$. So, we can improve the given algorithm only by diminishing dependence of running time on n. The main goal of any improvement is to diminish the average number of performances of the step 1.3. We can do it using similar methods as those given for the variants of the algorithm $NEXT$. Before any improvement of the general algorithm will be discussed we present the following example for its using.

Example 6.5.1 *Unrank a choice function h that* $R(h) = 83$ *for the model given in the example (3.7.6).*

The model represents the set of partitions of the set {1, 2, ..., 10} into four blocks that $k_1 = 4$, $k_2 = 2$, $k_3 = 3$, $k_4 = 1$. The corresponding table DBM is as follows:

$$\text{DBM}[7 \times 10] = \left|\begin{array}{cccccccccc} 1260 & 420 & 105 & 15 & 15 & 3 & 3 & 2 & 1 & 1 \\ 0 & 315 & 90 & 15 & 0 & 3 & 0 & 1 & 1 & 0 \\ 0 & 225 & 75 & 15 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 150 & 60 & 15 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 90 & 45 & 15 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 45 & 30 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 15 & 15 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right|$$

We have $p = 84$.

The sets $\mathcal{N}_i$ are specified as follows:

$\mathcal{N}_1 = \{1\}, \mathcal{N}_2 = \{2 \div 8\}, \mathcal{N}_3 = \{3 \div 9\}, \mathcal{N}_4 = \{4 \div 10\}, \mathcal{N}_5 = \{\min(\{1 \div 10\} - \{h(1)\} - \{h(2)\} - \{h(3)\} - \{h(4)\})\}, \mathcal{N}_6 = \{3 \div 10\}, \mathcal{N}_7 = \{\min(\{1 \div 10\} - \{h(1)\} - \{h(2)\} - \{h(3)\} - \{h(4)\} - \{h(5)\} - \{h(6)\})\}, \mathcal{N}_8 = \{4 \div 9\}, \mathcal{N}_9 = \{5 \div 10\}, \mathcal{N}_{10} = \{\min(\{1 \div 10\} - \{h(1)\} - \{h(2)\} - \{h(3)\} - \{h(4)\} - \{h(5)\} - \{h(6)\} - \{h(7)\} - \{h(8)\} - \{h(9)\})\}$.

The canonical form of the sets $\mathcal{N}_i$ is as follows:

$\mathcal{N}_1 = \{x_1\}, \mathcal{N}_2 = \{x_1, x_2, ...x_7\}, \mathcal{N}_3 = \{x_1, x_2, ...x_7\}, \mathcal{N}_4 = \{x_1, x_2, ...x_7\}, \mathcal{N}_5 = \{x_1\}, \mathcal{N}_6 = \{x_1, x_2, ...x_5\}, \mathcal{N}_7 = \{x_1\}, \mathcal{N}_8 = \{x_1, x_2\}, \mathcal{N}_9 = \{x_1, x_2\}, \mathcal{N}_{10} = \{x_1\}$.

Since $p <$ DBM[1, 1], so $h(1) = 1$.

Since $p <$ DBM[1, 2] and $h(2) > h(1)$, so $h(2) = 2$.

Since $p <$ DBM[1, 3] and $h(3) > h(2)$, so $h(3) = 3$.

Then, $p >$ DBM[1, 4], so $x > x_1$ and $p = 84 - 15 = 69$

$p >$ DBM[2, 4], so $x > x_2$ and $p = 69 - 15 = 54$

$p >$ DBM[3, 4], so $x > x_3$ and $p = 54 - 15 = 39$

$p >$ DBM[4, 4], so $x > x_4$ and $p = 39 - 15 = 24$

$p >$ DBM[5, 4], so $x > x_5$ and $p = 24 - 15 = 9$

$p <$ DBM[6, 4], so $x = x_6$, hence $h(4) = 9$.

Since $\mathcal{N}_5 = \{x_1\}$, so $h(5) = x_1(\mathcal{N}_5) = 4$.

Then, $p >$ DBM[1, 6], so $x > x_1$ and $p = 9 - 3 = 6$

$p >$ DBM[2, 6], so $x > x_2$ and $p = 6 - 3 = 3$

$p =$ DBM[3, 6], so $x = x_4$ and $p = 3 - 3 = 0$.

Therefore, $h(6) = 8$.

Then, $h(7) = x_1(\mathcal{N}_7) = 5$, $h(8) = x_1(\mathcal{N}_8) = 6$, $h(9) = x_1(\mathcal{N}_9) = 7$, $h(10) = x_1(\mathcal{N}_{10}) = 10$,

Therefore, we have $h =< h(1) = 1, h(2) = 2, h(3) = 3, h(4) = 9, h(5) = 4, h(6) = 8, h(7) = 5, h(8) = 6, h(9) = 7, h(10) = 10 >$.

□

6.5.1 *For increasing functions*

Given is the ranking model S_R =<<< $\mathcal{G}_i$ >, $1 \leq i \leq m; W$ given in (1.7); W_1; $\{h\}_{\mathcal{G}_i}$ >, $<^h$, table D> the requirement W_1 is arbitrary, the indexed family < $\mathcal{G}_i$ >, $1 \leq i \leq m$ is arbitrary except the model is reduced and the Q property holds.

Algorithm $UNRANKI$ (Increasing choice function)

Input: The model S_R that the indexed family < $\mathcal{G}_i$ >, $1 \leq i \leq m$; is reduced the Q property holds, the rank $R(h)$.

Output: The table $IA[i]$ representing the custom form of the choice function $h(i)$.

Method: Similarly as for the general $UNRANK$ algorithm, we use the table GI in order to implement the function $ord(q, \mathcal{G}_i^{\min})$;

1. $x \longleftarrow 1$;
2. **for** $i \longleftarrow 1$ **to** m **do**
2.1. $IA[i] \longleftarrow GI[x, i]$;
2.2. **while** $R(h) >$D$[x, i]$ **do**
2.2.1. **if** $IA[i] \in \mathcal{G}_i$ and W_1 holds for $IA[i]$ **then** $R(h) \longleftarrow R(h)-D[x, i]$;
2.2.2. $x \longleftarrow x + 1$;
2.2.3. $IA[i] \longleftarrow GI[x, i]$;
3. **return** IA;

Asymptotic complexity of the above algorithm is $O(n+m)$. Since, $n > m$, so we have $O(n)$. For proving correctness of $O(n)$ it is enough to observe that the step 2.2. can be performed at most n times for the whole run of the algorithm, while the loop 2. justifies dependence on m. Time complexity of the above algorithm can be diminished for special instances of the model S_R. Moreover, for implementing $IA[i] \in \mathcal{G}_i$ at the step 2.2.1. we can use the table G.

Given is the ranking model S_R =<<< $\mathcal{G}_i$ >, $1 \leq i \leq m; W$ given in (1.7); $\{h\}_{\mathcal{G}_i}$ >, $<^h$, table D>, the indexed family < $\mathcal{G}_i$ >, $1 \leq i \leq m$. Then, we have the following algorithm $UNRANKIVW$.

Algorithm $UNRANKIVW$ (Increasing choice function with Void requirement W_1)

Input: The model S_R that the indexed family < $\mathcal{G}_i$ >, $1 \leq i \leq m$; is reduced the Q property holds, the rank $R(h)$.

Output: The table $IA[i]$ representing the custom form of the choice function $h(i)$.

Method: The same arrangements as for the algorithm $UNRANKI$.

1.$x \longleftarrow 1$;
2. **for** $i \longleftarrow 1$ **to** m **do**

2.1. $IA[i] \longleftarrow GI[x,i]$;
2.2. **while** $R(h) >$D$[x,\ i]$ **do**
2.2.1. $R(h) \longleftarrow R(h)-D[x,\ i]$;
2.2.2. $x \longleftarrow x+1$;
2.2.3. $IA[i] \longleftarrow GI[x,i]$;
3. **return** IA;

It is worthwhile to observe that the algorithm $UNRANK$IVW does not need to use any representation of the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$, except the table GI, since all the needed data are represented by the entries of the table D. Observe, the step 2.2.1. of the algorithm $UNRANK$IVW is equivalent to the step 2.2.1. of the algorithm $UNRANK$I since if $IA[i] \notin \mathcal{G}_i$, then D$[x,\ i] = 0$. Then, $R(h)$ remains unchanged. Moreover, performance of the step 1.1. makes assertion that the requirement W holds. If the number of entries D$[x,i] = 0$ is big, then we can apply the following algorithm $UNRANK$IRG in order to diminish the number of runs of the step 2.2.

Given is the model S_R as for the above case except the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$, is not "dense" that means the table D posses many entries equal 0. In fact, the cardinals of the sets $\mathcal{G}_i^{\min}$ of the minimum non-deformed indexed family $< \mathcal{G}_i^{\min} >, 1 \leq i \leq m$ are much greater than the cardinals of the corresponding sets $\mathcal{G}_i$.

Algorithm $UNRANK$IRG (Increasing choice function with Random sets $\mathcal{G}_i$)

Input: The model S_R, the rank $R(h)$.

Output: The table $IA[i]$ representing the custom form of the choice function h.

Method:

1. $y \longleftarrow 1$;
2. $IA[i] \longleftarrow Y[y,i]$;
3. **for** $i \longleftarrow 1$ **to** $m-1$ **do**
3.1. $x \longleftarrow X[IA[i],i]$;
3.2. **while** $R(h) >$D$[x,i]$ **do**
3.2.1. $R(h) \longleftarrow R(h)-D[x,i]$;
3.2.2. $y \longleftarrow y+1$;
3.2.3. $x \longleftarrow GI[Y[y,i],i]$;
3.3. $IA[i] \longleftarrow Y[y,i]$;
3.4. **if** $Q[IA[i]+1,i+1] = Q[IA[i],i+1]$ **then** $y \longleftarrow Q[IA[i]+1,i+1]+1$ **else** $y \longleftarrow Q[IA[i]+1,i+1]$;
3.5. $IA[i+1] \longleftarrow Y[y,i+1]$;
4. $x \longleftarrow X[IA[m],m]$;
4.1. **while** $R(h) >$D$[x,\ m]$ **do**
4.1.1. $R(h) \longleftarrow R(h)-D[x,m]$;
4.1.2. $y \longleftarrow y+1$;
4.1.3. $x \longleftarrow GI[Y[y,m],m]$;

4.2. $IA[m] \longleftarrow Y[y, m]$;
5. **return** IA;

The steps 1. and 3.2.2. make assertion that the values $IA[i] \notin \mathcal{G}_i$ would not be used for running the loop 3.2. The step 4. makes the same job for $i = m$ as does the step 3. for $i < m$ except the step 3.4. does not have its counterpart. Therefore, we have asymptotic complexity $O(\max(\mid \mathcal{G}_i \mid)+m)$.

6.5.2 For monotonic functions

For the monotonic choice functions we get similar algorithms as for increasing choice functions. The only difference is in evaluation $IA[i+1]$ that for the monotonic choice functions we use $IA[i+1] \longleftarrow IA[i]$. The other differences are the consequences of it.

6.5.3 For bijections

The proper $UNRANK$ algorithm for choice bijections can be developed using as the foundation the general $UNRANK$. Moreover, we adopt all those implementations developed for the corresponding variant of $NEXT$ algorithm that enable us to reduce the complexity of the general algorithm.

Given is the ranking model $S_R =<<< \mathcal{G}_i >, 1 \leq i \leq m$; W given in (1.19); W_1; $\{h\}_{\mathcal{G}_i} >$, $<^h$, table D>, the indexed family $< \mathcal{G}_i >, 1 \leq i \leq m$ is arbitrary except the model is reduced and the Q property holds.

Algorithm $UNRANK$B (choice Bijection)
Input: The model S_R, the rank $R(h)$, the table B.
Output: The choice bijection h represented by the table IA.
Method: We use the table B in the way specified for bijections.
1. **for** $i \longleftarrow 1$ **to** m **do**
1.1. $IA[i] \longleftarrow AG[i, 1]$;
1.2. $x \longleftarrow GI[IA[i], i]$;
1.3. **while** $R(h) >$D$[x, i]$ **do**
1.3.1. **if** $B[X[x, i]] = FALSE$ and W_1 holds **then** $R(h) \longleftarrow R(h)-D[x, i]$;
1.3.2. $x \longleftarrow x + 1$;
1.4. $IA[i] \longleftarrow X[x, i]$;
1.5. $B[IA[i]] \longleftarrow TRUE$;
2. **return** IA, B;

For this algorithm, we have $O(n.m)$. It is very essential to compare complexities of the proper instances of the algorithm $NEXT$ and its counterpart concerning the algorithm $UNRANK$. We have observed that the complexity of the algorithm $NEXT$ for an average case changes from $\Theta(n)$ into $\Theta(1)$ depending on the model S_U used and on an implementation.

The corresponding variant of the algorithm *UNRANK* changes its complexity for the most optimistic case from $\Omega(n.m)$ into $\Omega(m)$. That means **the algorithm *UNRANK* is generally much more complex than its *NEXT* counterpart**. It excludes usage of *UNRANK* where usage of *NEXT* would be enough.

6.6 The algorithms *RANK* and *RRANGE*

Given is the ranking model $S_R = < S_U, <^h$, table D$>$, where $S_U = << \mathcal{G}_i >, 1 \leq i \leq m; W; W_1; \{h\}_{\mathcal{G}_i} >$ and $<^h$ is the lexical order. The model is reduced and the Q property holds.

Let $R_{\min}h(i)$ denote the minimum rank $R(h')$ of a choice function h' that $h'(1) = h(1)$, $h'(2) = h(2)$, ..., $h'(i) = h(i)$, $1 \leq i < m$.

We observe that $R_{\min}h(i) = R_{\min}h(i-1)$, if and only if $h(i)$ is the minimum value for the extension of the partial mapping $h(1)$, $h(2)$, ..., $h(i-1)$ to the partial mapping $h(1)$, $h(2)$, ..., $h(i)$. We define $R_{\min}h(0) = 0$.

Definition 6.6.1 *The $i-th$ rank difference $\Delta_i R(h)$ is defined to be* $R_{\min}h(i) - R_{\min}h(i-1)$.

Let a partial mapping $< h(1), h(2), ..., h(i-1) >$ be given, while $< x_i^1, x_i^2, ..., x_i^{z} >$ being the ordered string of values that $q = x_i^j(\mathcal{G}_i)$ could be used as an extension of $< h(1), h(2), ..., h(i-1) >$, $x_i^j = ord(h(i), \mathcal{G}_i)$. Then, $\mathcal{G}_i^* = \{q : q = x(\mathcal{G}_i)$ that $x \in \{x_i^1, x_i^2, ..., x_i^{z}\}$, while $\overline{\mathcal{G}}_i^* = \{q : q = x(\mathcal{G}_i)$ that $x \in \{x_i^j, x_i^{j+1}, ..., x_i^{z}\}$. Hence, $\mathcal{G}_i^* - \overline{\mathcal{G}}_i^*$ is the set of all the values of $\mathcal{G}_i^*$ that do not belong to $\overline{\mathcal{G}}_i^*$, i.e., $(\mathcal{G}_i^* - \overline{\mathcal{G}}_i^*) = \{q :\ q = x(\mathcal{G}_i)$ that $x \in \{x_i^1, x_i^2, ..., x_i^{j-1}\}$.

Example 6.6.1 *For the unranking model $S_U = << \mathcal{E}_i >, 1 \leq i \leq 4$, $\mathcal{E}_i = \{5, 6, ...10\}$; $W(i)$ given in (1.19); $\{h\}_{\mathcal{E}_i} >$ and the following partial mapping: $h(1) = 6$, $h(2) = 10$, $h(3) = 8$, we have $\mathcal{E}_3^* = \{5, 7, 8, 9\}$ and $\overline{\mathcal{E}}_3^* = \{9\}$ and $\mathcal{E}_3^* - \overline{\mathcal{E}}_3^* = \{5, 7, 8\}$. So, $x \in \{1,3,4,5\}$ for $\mathcal{E}_3^*$; $x \in \{5\}$ for $\overline{\mathcal{E}}_3^*$ and $x \in \{1, 3, 4\}$ for $\mathcal{E}_3^* - \overline{\mathcal{E}}_3^*$.*

□

With denotation X_i^* we mean the set of values x that produce the set $\mathcal{G}_i^*$, i.e., $X_i^* = \{x : x(\mathcal{G}_i) \in \mathcal{G}_i^*\}$. Similarly, with denotation $\overline{X}_i^*$ we mean the set of values x that produce the set $\overline{\mathcal{G}}_i^*$, i.e., $\overline{X}_i^* = \{x : x(\mathcal{G}_i) \in \overline{\mathcal{G}}_i^*\}$. Correspondingly, $X_i^* - \overline{X}_i^* = \{x : x(\mathcal{G}_i) \in (\mathcal{G}_i^* - \overline{\mathcal{G}}_i^*)\}$.

The following lemma shows the relation between the $i-$th rank difference and the entries D$[x, i]$ of the table D.

Lemma 6.6.1 *The rank difference* $\Delta_i R(h) = \sum_{x \in (X_i^* - \overline{X}_i^*)} D[x, i]$.

Proof. In order to make an extension of $< h(1), h(2), ..., h(i-1) >$ to $< h(1), h(2), ..., h(i-1), h(i) >$ we can use all the elements that belong to $\mathcal{G}_i^*$.

If $h(i) = \min(\mathcal{G}_i^*)$, then $\Delta_i R(h) = 0$.

Then, $\Delta_i R(h)$ equals to the number of choice functions that the partial mapping $h(1), h(2), ..., h(i-1)$ is fixed and $h^*(i) < h(i)$, where $h(i)$ is fixed and $h^*(i)$ denotes all acceptable values that are smaller than $h(i)$. Hence, $h^*(i) \in (\mathcal{G}_i^* - \overline{\mathcal{G}}_i^*)$. So, $\Delta_i R(h)$ equals to the number of choice functions that $h^*(i) = x(\mathcal{G}_i^* - \overline{\mathcal{G}}_i^*)$ and the partial mapping $h(1), h(2), ..., h(i-1)$ is fixed. Therefore, $x \in (X_i^* - \overline{X}_i^*)$.

Since the set $\{h\}_{\mathcal{G}_i}$ is symmetric, so each number $\mathrm{N}[x(\mathcal{G}_i^* - \overline{\mathcal{G}}_i^*) : x \in (X_i^* - \overline{X}_i^*)]$ is independent from the partial mapping $h(1), h(2), ..., h(i-1)$ and it equals $\mathrm{D}[x, i]$. Hence, the number of all the choice functions that posses the rank $R(h)$ smaller than any extension of $< h(1), h(2), ..., h(i-1), h(i) >$ to a full choice function h equals $\sum_{x \in (X_i^* - \overline{X}_i^*)} \mathrm{D}[x, i]$.

Therefore, $\Delta_i R(h) = \sum_{x \in (X_i^* - \overline{X}_i^*)} \mathrm{D}[x, i]$

□

Example 6.6.2 *Evaluate* $\Delta_3 R(h)$ *for the partial mapping* $h : 1 \to 2,\ 2 \to 2,\ 3 \to 4$, *where* $S_U = << \mathcal{P}_i >, 1 \leq i \leq 5, \mathcal{P}_1 = \mathcal{P}_2 = \{2\}, \mathcal{P}_3 = \{2, 3, 4\}, \mathcal{P}_4 = \{2, 3, 4, 5\}, \mathcal{P}_5 = \{3, 4, 5\}; W(i) : h(i) \geq h(i-1); \{h\}_{\mathcal{P}_i} >$.

The table DDCM representing the structure of $\{h\}_{\mathcal{P}_i}$ is as follows:

$$\mathrm{DDCM}[4 \times 5] = \begin{vmatrix} 9 & 9 & 9 & 3 & 0 \\ 0 & 0 & 6 & 3 & 1 \\ 0 & 0 & 3 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{vmatrix}$$

We have $\mathcal{P}_3^* = \{2, 3\}$. Since $h(3) = 3(\mathcal{P}_3)$, so $\mathcal{P}_3^* - \overline{\mathcal{P}}_3^* = \{2\}$.

Hence, $\Delta_3 R(h) = \mathrm{DDCM}[2, 3] = 6$.

□

The investigated sets $\mathcal{G}_i^*$ and $\overline{\mathcal{G}}_i^*$ are valid concepts for explaining and for developing numerous algorithms concerning the generation of combinatorial objects. Their usage for explanation of the relation between the rank difference and the entries of the D tables is only a part of these more general applications.

6.6.1 Evaluations of the sets $\overline{\mathcal{G}}_i^*$ and $\mathcal{G}_i^*$

We are concerned now with the cases of the general algorithm $SET\overline{\mathcal{G}}_i^*$ for evaluation of the sets $\overline{\mathcal{G}}_i^*$ according to the Definition 3.2.3, p. 55.

Increasing choice functions

Given is the model $S_U =<< \mathcal{A}_i >, 1 \leq i \leq m; W$ given in (1.1); W_1; $h \in \{h\}_{\mathcal{A}_i} >$.

Algorithm $SET\overline{\mathcal{G}}_i^*(SET\mathcal{A}_i^*)$I(for Incresing choice functions)
Input: S_U, $<^h$, i, x_{i-1} that $h(i-1) = x_{i-1}(\mathcal{A}_{i-1})$.
Output: $\overline{\mathcal{A}}_i^*$.
Method: at the beginning $\overline{\mathcal{A}}_i^* = \emptyset$, we have $x_i^{\max}(\mathcal{A}_i) = \max(\mathcal{A}_i)$.
1. **for** $x \longleftarrow x_{i-1}$ **to** $x_i^{\max}$ **do**
1.1. **if** $x(\mathcal{A}_i)$ satisfies the requirement W_1 **then** $\overline{\mathcal{A}}_i^* \longleftarrow \overline{\mathcal{A}}_i^* \cup \{x(\mathcal{A}_i)\}$;

Monotonic choice functions

The algorithm $SET\overline{\mathcal{G}}_i^*$ is exactly the same as for the increasing choice functions, instead $\overline{\mathcal{G}}_i^* = \overline{\mathcal{A}}_i^*$ we have only to put $\overline{\mathcal{G}}_i^* = \overline{\mathcal{E}}_i^*$.

Choice bijections

Given is the model $S_U =<< \mathcal{E}_i >, 1 \leq i \leq m; W$ given in (1.19); $h \in \{h\}_{\mathcal{E}_i} >$.

Algorithm $SETB\overline{\mathcal{G}}_i^*(SET\overline{\mathcal{E}}_i^*)$(for choice Bijections)
Input: S_U, $<^h$, i, x_{i-1} that $h(i-1) = x_{i-1}(\mathcal{E}_{i-1}) 1 \leq i \leq m$.
Output: $\overline{\mathcal{E}}_i^*$.
Method: at the beginning $\overline{\mathcal{E}}_i^* = \emptyset$.
1. **for** $x \longleftarrow 1$ **to** $x_i^{\max}$**do**
1.1. **if** $x(\mathcal{E}_i) \neq h(1)$ and $x(\mathcal{E}_i) \neq h(2)$ and...$x(\mathcal{E}_i) \neq h(i-1)$ and $x(\mathcal{E}_i)$ satisfies W_1**then** $\overline{\mathcal{E}}_i^* \longleftarrow \overline{\mathcal{E}}_i^* \cup \{x(\mathcal{E}_i)\}$

The given instances of the algorithm $SET\overline{\mathcal{G}}_i^*$ for evaluation of the set $\overline{\mathcal{G}}_i^*$ can easily be adopted for evaluation of the corresponding sets $\mathcal{G}_i^*$. Then, it is enough to replace the loop 1. **for** $x \longleftarrow 1$ **to** $x_i^{\max}$**do** with the loop 1. **for** $x \longleftarrow x_i^*$ **to** $x_i^{\max}$**do**, where $x_i^* = h(i)$. We can implement the step 1.1. using the table B in the similar way as it was done for the algorithms *NEXT* and *UNRANK*. Then, the set $\overline{\mathcal{E}}_i^*$ can be produced in time $O(\mid \mathcal{E}_i \mid)$. Correspondingly, we can evaluate the set $(\mathcal{G}_i^* - \overline{\mathcal{G}}_i^*)$.

6.6.2 The algorithm RANK

Algorithm RANK (general)
Input: The model S_R, the choice function h given in the canonical form, i.e., $h(1) = x_1(\mathcal{G}_1)$, $h(2) = x_2(\mathcal{G}_2)$, ..., $h(m) = x_m(\mathcal{G}_m)$.

Output: The rank $R(h)$.
Method: Given are the sets $\overline{\mathcal{G}}_i^*$ for $1 \leq i \leq m$.
1. $R(h) \longleftarrow 0$;
2. **for** $i \longleftarrow 1$ **to** m **do**
2.1. **for** $x \longleftarrow AG[i, 1]$; **to** $x_i - 1$ **do**
2.1.1. **if** $x(\mathcal{G}_i) \in (\mathcal{G}_i^* - \overline{\mathcal{G}}_i^*)$ **then** $R(h) \longleftarrow R(h)$+D$[x, i]$;
2. **return** $R(h)$;

For this algorithm we have $O(n.m)$ and $\Omega(m)$. We have to emphasize that complexity of the algorithm $RANK$ has no practical meaning since it can run only several times during the generation process. Explanation of this fact will be given in the next chapter.

If we replace step 2. with the following 2. **for** $i \longleftarrow 1$ **to** t **do**, where $t < m$, then the algorithm $RANK$ produces the rank $R_{\min}(h(t))$.

6.6.3 Evaluation of the rank range

Given is the ranking model $S_R =< S_U$, table D;'$<^{h'}>$, where $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; \{h\}_{\mathcal{G}_i} >$. Moreover, we have a subsystem $< Methods > =< FIRST, NEXT, SET\overline{\mathcal{G}}_i^*, LAST >$. In order to present the algorithm $RRANGE$, we make the following specifications.

The algorithm $FIRST(S_U)$ generates the first choice function for the system S_U.

The algorithm $NEXT(h, S_U)$ takes choice function h of S_U and retrieves the next choice function h' i.e., $R(h') = R(h) + 1$.

The algorithm $SET\overline{\mathcal{G}}_i^*$ takes as input the partial mapping $h(1)$, $h(2)$, ..., $h(i-1)$ and it retrieves $\overline{\mathcal{G}}_i^*$.

The algorithm $LAST(S_U)$ retrieves $TRUE$ if a choice function h has a highest rank $R(h)$ for the given model.

The algorithm $RANK(t, g, R_{\min}h(t))$ takes the partial mapping g for a given value t and retrieves the rank $R_{\min}h(t)$.

We are concerned with evaluation of the rank range for the subset of choice functions $\{h'\} \subset \{h\}_{\mathcal{G}_i}$ that is defined as follows. Given is a subsystem $S'_U =<< \mathcal{G}'_i >, i \in I', I' \subset \{1, 2, ..., m\}; W; \{f\}_{\mathcal{G}'_i} >$ of S_U, i.e., $\mathcal{G}'_i \subseteq \mathcal{G}_i$, the requirement W is common for S'_U and for S_U. The set $\{h'\}_{\mathcal{G}_i}$ contains all the extensions of the partial mappings $\{f\}_{\mathcal{G}'_i}$ to the choice functions $h \in \{h\}_{\mathcal{G}_i}$. We consider two steps extension of each $f \in \{f\}_{\mathcal{G}'_i}$ to the corresponding subset of $\{h'\}$. The first step produces a partial mapping $g(1)$, $g(2)$, ..., $g(t)$, where $t = \max(I')$. The second step takes $g(1)$, $g(2)$, ..., $g(t)$ as input and produces the corresponding subset of $\{h'\}_{\mathcal{G}_i}$. We know that the set of extensions of $g(1)$, $g(2)$, ..., $g(t)$ possess a rank range that is continuous in $R(h)$. Therefore, for evaluation of the rank range, we have to produce $R_{\min}g(t)$.

We will give now the general algorithm $RRANGE$ for evaluation of the rank range $R(\{h'\})$ in $R(h)$.

Algorithm $RRANGE$(general)
Input: The models S_{RN} and S'_{UN}.
Output: The rank range $R(\{h'\}_{\mathcal{G}_i})$.
Method: For each consecutive $f \in \{f\}_{\mathcal{G}'_i}$ there is generated system $S''_{UN} = << \mathcal{G}''_i >, 1 \leq i \leq t, t = \max(I'), \mathcal{G}''_i = \{f(i)\}$ for $i \in I', \mathcal{G}''_i \subseteq \overline{\mathcal{G}}^*_i$ for $i \notin I'; W; \{f\}_{\mathcal{G}'_i} >$. The indexed family $<< \mathcal{G}''_i >, 1 \leq i \leq t$ is reduced and the Q property holds for S''_{UN}. With denotation $R(\{h'_g\}_{\mathcal{G}_i}) \cup (R_{\min}(h(t)), e)$ used at the step 2.3.3., we emphasize that consecutive rank range $(R(h'_{\min})_j, e)$ is attached to the string of the rank ranges.

1. $f \longleftarrow FIRST(S'_U)$
2. **while** $LAST(f) = FALSE$ **do**
2.1. generate system S''_U;
2.2. $g \longleftarrow FIRST(S''_U)$;
2.3. **while** $LAST(g) = FALSE$ **do**
2.3.1. $RANK(t, g, R_{\min}h(t))$;
2.3.2. $e \longleftarrow$D$[x_i, i]$ that $x_i(\mathcal{G}_i) = g(t)$;
2.3.3. $R(\{h'\}_{\mathcal{G}_i}) \longleftarrow R(\{h'\}_{\mathcal{G}_i}) \cup (R_{\min}h(t), e)$;
2.3.4. $g \longleftarrow NEXT(g)$;
2.4. $f \longleftarrow NEXT(f)$;
3. **return** $R(\{h'\}_{\mathcal{G}_i})$.

Complexity of the evaluation of $R(\{h'\})$ for each choice mapping g is determined by the generation of a consecutive g and by the evaluation of $R_{\min}h(t)$ according to $RANK(t, g, R_{\min}h(t))$. Hence, we have $O(n.m)$ for the worst case. Asymptotic complexity of the algorithm $RRANGE$ depends on $\mid f \mid$ and $\mid g \mid$. If for each g the number of extensions to f would equal $\mid f \mid$, then for evaluation of the rank range time $O(\mid g \mid . \mid f \mid .n.m)$ is needed.

6.7 Structure of libraries

The methodology for representing and for generating arbitrary sets of combinatorial objects is very homogenous. The classes of unranking and ranking models create a hierarchy ruled by more and more restricting properties of indexed families and requirements W & W_1. We have shown the dedicated optimal algorithms of the system $< Methods >$ for the main classes of models. If we do not have an optimal algorithm for a model represented by a lowest level class of models of the hierarchy, then an algorithm that is optimal for a more general class of models can be used. That makes the whole system of representation and generation very convenient for usage and for

development since we can use successfully the object oriented programming. The basic structure of a generation object is as follows: $<<$ *Representation* $>$, $<$ *Methods* $>>$. The object $<$ *Representation* $>$ contains methods to be used for building the models S_U and S_R, while $<$ *Methods* $>$ contains instances of the algorithms *FIRST*, *NEXT*, *LAST*, *RANK*, *UNRANK*, *RRANGE*, *TABD*. The components of the $<$ *Methods* $>$ subsystem are obtained by succession or by individual development. We can also have the object $<$ *Specification* $>$ that enables us to assign the generation object to the given task. For making libraries the systems $<$ *Methods* $>$ and $<$ *Representation* $>$ should be organized and developed individually since the same object $<$ *Methods* $>$ can be assigned to several instances of the $<$ *Representation* $>$ system. It is very important to emphasize that there is no correspondence between organization of the proposed system and the basic combinatorial objects. In order to develop the systems $<$ *Representation* $>$ and $<$ *Methods* $>$, we have to think in terms of classes of choice functions but we do not need to think what kind of objects can be represented. On the other hand the system $<$ *Specification* $>$ should take the classical description of the set of combinatorial objects to be generated as input and it should produce the proper object $<<$ *Representation* $>$, $<$ *Methods* $>>$ as output. This general approach to be developed by using object oriented programming would be suitable on one side of an application that does not require the best possible performance. On the other hand a skilled user gets very advanced tools suitable for solving extremely difficult generation tasks in a short time.

6.8 Conclusions

We have developed the basic algorithms for the generation of the choice functions of indexed families. The unranking and ranking algorithms are very general and can be applied for any model S_U or S_R that is reduced and the Q property holds, respectively. On the other hand specified indexed families and the requirements W and W_1 enable us to produce the most suitable dedicated variants or implementations of this general algorithms. So, we can have optimal algorithms for a concerned model. If we do not want to develop or to use dedicated algorithms, then we can use a more general algorithm that the class of generality of the models covers also our specific case. The given unranking algorithms are much more general than those known in the classical combinatorics and cover the sets of combinatorial objects never considered as subjects of representation and generation. We have shown that each ranking algorithm effectively uses corresponding table D. Consequently the ranking algorithms are optimal if are applied for the full sets of the basic combinatorial objects. Moreover, we have given their variants applicable for any instance of sets of combinatorial objects. The generality of the proposed approach enables us to develop computer systems using object oriented programming that can supply convenient tool for any need concerning combinatorial computations.

7 Generation of sets of choice functions

7.1 Sequential generation

We are concerned now with the sequential generation of choice functions. For a given set of choice functions $\{h'\}$ to be generated, we can use two objects the unranking generation object G_U and the ranking generation object G_R.

The unranking generation object is the system $G_U =< S_U, \prec^h, < M_U >>$. With denotation $\prec^h$ we mean the order in which the choice functions are to be generated. Usually $\prec^h = <^h$, i.e., the choice functions are to be generated in lexical order. Nevertheless, the anti−lexical order can also be used or we can use some other orders specified in the next section. The subsystem $< M_U >$ contains rutines $FIRST$, $NEXT$, $LAST$ specified in the previous section under a silent assumption that $\prec^h = <^h$. We can extend the subsystem $< M_U >$ even more in order to meet conditions of the generation that come from special sets $\{h'\}$ and from other environmental conditions like the requested order $\prec^h$. In order to solve those problems, we can add some procedures for controlling the generation order or procedures for transforming the given model S_U into a model that is suitable for meeting special requirement or special circumstances concerning properties of the set $\{h'\}$. Our approach would base on Ranking Theorem and on formal transformations of the models.

The ranking generation object is the system $G_{RN} =< S_R, \prec^h, < M_R >>$. The basic system $< M_R >$ contains $RANK$, $UNRANK$, $RRANGE$ and some auxiliary procedures as specified in the previous chap-

ter. Moreover, we use the system $< M_U >$. The subsystems $< Methods >$ can also be extended in order to meet the environmental conditions.

Using objects G_U and G_R, we develop sequential procedures for the generation of the choice functions.

* If the set $\{h'\} = \{h\}_{\mathcal{G}_r}$, then for the generation, we use the following algorithm UNSEQ1.

Algorithm UNSEQ1(h') (UNranked SEQuential generation 1)
Method: we use $< FIRST, NEXT, \overline{LAST} >$;
at the beginning h' is void.
1. if h' is void then call $FIRST$.
2. while $R(h') <\mid \{h'\} \mid$ do
2.1. call $NEXT(h')$
2.2. (process h')

Asymptotic complexity of this algorithm changes from $O(\mid \{h'\} \mid)$ into $O(n.m. \mid \{h'\} \mid)$ depending on the model properties and on the variants of the algorithm $NEXT$ used. Since we have shown that the algorithm $UNRANK$ is more complex than its $NEXT$ counterpart, so the algorithm UNSEQ1 is the best for sequential generation of the choice functions, unfortunately it can not be used if $\{h'\} \neq \{h\}_{\mathcal{G}_r}$.

** If the set $\{h'\} \neq \{h\}_{\mathcal{G}_r}$ but the rank range $R(\{h'\})$ is continuous in $R(h)$ and if the set $\{h'\}$ is symmetric, then we can transform the model S_U into the model S'_U that $\{h'\} = \{h'\}_{\mathcal{G}'_r}$ and we follow the procedure for the case *. Otherwise, the algorithm RANSEQ1 can be used.

Algorithm RANSEQ1(h') (RANked SEQuential generation 1)
Method: we use $< FIRST, NEXT, \overline{LAST}, UNRANK, RRANGE, >$
1. call $RRANGE(R(h'_{\min}), e)$
2. $h' \leftarrow UNRANK(R(h'_{\min}))$
3. while $R(h') < R(h'_{\min}) + e - 1$ do
3.1. call $NEXT(h')$
3.2. (process h')

Since algorithms $RRANGE$ and $UNRANK$ are called only once so it does not effect asymptotic complexity that is the same as for the previous algorithm. Time complexity is practically also not effected.

*** If $\{h'\} \neq \{h\}_{\mathcal{G}_r}$ and the rank range $R(\{h'\})$ is not continuous in $R(h)$, then the algorithm RANSEQ2 can always be used.

Algorithm RANSEQ2(h') (RANked SEQuential generation 2)
Method: we use $< FIRST, NEXT, \overline{LAST}, UNRANK, RRANGE, >$
1. repeat
1. 1. produce consecutive rank range $(R(h'_{\min})_j, e_j)$ using $RRANGE$
1. 2. $h' \leftarrow UNRANK(R(h'_{\min})_j)$

1. 3. while $R(h') < R(h'_{\min})_j + e_j - 1$ do
1. 3.1. call $NEXT(h')$
1. 3.2. (process h')
until the rank range $(R(h'_{\min})_j, e_j)$ is the last one

The algorithm RANSEQ2 possess higher asymptotic complexity and higher time complexity than the algorithms RANSEQ2 and UNSEQ1. It is effected by higher time complexity of the methods $UNRANK$ and $RANK$ in comparing with the method $NEXT$. In fact, if the number of the consecutive rank ranges $(R(h'_{\min})_j, e_j)$ grows, then complexity of the algorithm RANSEQ2 is greater and greater, since each rank range $(R(h'_{\min})_j, e_j)$ requires usage of one call of $RANK$ and one call of $UNRANK$. If the number of consecutive rank ranges $(R(h'_{\min})_j, e_j)$ grows exponentially on n and m, then for the algorithm RANSEQ2 we get $\Omega(m. \mid \{h'\} \mid)$ for the most optimistic case. Since for the algorithms UNSEQ1 or RANSEQ1 we can have $O(\mid \{h'\} \mid)$, so it demonstrates complexity of the algorithm RANSEQ2 in comparing with the algorithms UNSEQ1 and RANSEQ1. Suppose, the rank range $(R(h'_{\min})_j, e_j$ is $z-$tuple. So, if smaller value z, then smaller asymptotic and time complexity of the algorithm RANSEQ2. It demonstrates usefulness of Ranking Theorem if we are searching for a most suitable model. For illustration of the general methodology of building the models S', we refer the reader to the example (4.3.1).

7.1.1 For hierarchical systems

Given is a hierarchical unranking representation model as specified in (1.22).

The models $<< \mathcal{G}_i >, 1 \leq i \leq m; W; \{h\}_{\mathcal{G}_i} >$ and $<< \mathcal{G}_j >, j \in J; W^f; \{f\}_{\mathcal{G}_j} >$ are reduced and the Q property holds for each model. The requirement W^f is the restriction of the requirement W and the set $f \in \{f\}_{\mathcal{G}_j}$ makes a partition on the set of choice functions $\{h\}_{\mathcal{G}_i}$. Then, we have the following algorithm EXTEN for the generation of all the choice functions $\{h\}_{\mathcal{G}_i}$

Algorithm EXTEN
Input: The hierarchical model specified in (1.22).
Output: The set of choice functions $\{h\}_{\mathcal{G}_i}$
Method: Each choice function h is an extension of a choice function f.
1. call $FIRST(f)$;
1.1. call $FIRST(h^f, f)$;
1.2. **repeat**
1.2.1. call $NEXT(h^f, f)$;
1.2.1. process h^f
until all the choice functions h^f that are extensions of f are generated;
2. **repeat**

2.1. call $NEXT(f)$;
2.2. **repeat**
2.2.1. call $NEXT(h^f, f)$;
2.2.1. process h^f
until all the choice functions h^f that are extensions of f are generated
until all the choice functions $\{f\}_{\mathcal{G}_j}$ are generated;

7.1.2 Piecewise lexical order

For numerous applications the generation and processing a given set of choice functions using the lexical order is not convenient. For instance, we may require at first the generation of the combinations corresponding to $h(1) = 2$ and $h(2) = 5$, then the combinations corresponding to $h(1) = 6$ and $h(2) = 8$ should be generated.

We assume now that the set of choice functions to be generated $\{h'\}$ is symmetric. Then, we can build a model S'_U that the set $\{h'\} = \{h'\}_{\mathcal{G}'_i}$ basing on Ranking Theorem. Moreover, we can build the corresponding model S'_R since there is a table D that represents the structure of $\{h'\}_{\mathcal{G}'_i}$. Therefore, we are concerned with full sets of the choice functions $\{h\}_{\mathcal{G}_i}$ in the following part of this section.

With denotation S, we mean any ranking or unranking model that represents a given set of choice functions $\{h\}_{\mathcal{G}_i}$.

Let a given model S be split into a set of models $\{S_1, S_2, ..., S_w\}$ that the corresponding sets of choice functions $\{h\}^1_{\mathcal{G}^1_i}$, $\{h\}^2_{\mathcal{G}^2_i}$, ..., $\{h\}^w_{\mathcal{G}^w_i}$ are distributed i.e.,

(*i*) $\{h\}^j_{\mathcal{G}^j_i} \cap \{h\}^k_{\mathcal{G}^k_i} = \emptyset$, for any j and k that $j, k \in \{1, 2, ..., w\}$, $j \neq k$

(*ii*) $\{h\}^1_{\mathcal{G}^1_i} \cup \{h\}^2_{\mathcal{G}^2_i} \cup ... \cup \{h\}^w_{\mathcal{G}^w_i} = \{h\}_{\mathcal{G}_i}$.

Definition 7.1.1 *The piecewise lexical order* $\prec^h$ *is the order of the choice functions* $h \in \{h\}_{\mathcal{G}_i}$ *generated by the order of the submodels* $< S_1, S_2, ..., S_w >$ *and by the lexical order of the choice functions, i.e., if* $h_1 \in \{h\}^j_{\mathcal{G}^j_i}$, $h_2 \in \{h\}^k_{\mathcal{G}^k_i}$ *and if* $j < k$, *then* $h_1 \prec^h h_2$; *if* $h_1, h_2 \in \{h\}^j_{\mathcal{G}^j_i}$ *and if* $R(h_1) < R(h_2)$ *then* $h_1 \prec^h h_2$.

So, splitting the model S into the models $\{S_1, S_2, ..., S_w\}$ and making their string $< S_1, S_2, ..., S_w >$ is equivalent into defining a piecewise lexical order $\prec^h$ on the set $\{h\}$. It is easy to observe that any order on the set of choice functions $\{h\}$ can be seen as a case of the piecewise lexical order. We can note that w can equal $|\{h\}|$. Then, each model S_j represents exactly one choice function $h \in \{h\}$. So, making a string $< S_1, S_2, ..., S_w >$, we define an arbitrary order on the set $\{h\}$.

For splitting the model S into a string of submodels $< S_1, S_2, ..., S_w >$, we use a method $SPLIT$. The method $SPLIT$ takes the system S and the

order of the desired subsets $< \{h\}^1_{\mathcal{G}_i^1}, \{h\}^2_{\mathcal{G}_i^2}, ..., \{h\}^w_{\mathcal{G}_i^w} >$ as input and returns the corresponding string of the submodels $< S_1, S_2, ..., S_w >$ as output. We have the following basic *SPLIT* methods.

Method SPLIT_layer
Input: the model $S = S_U, \prec^h$.

1. Split the given indexed family $< \mathcal{G}_i >, 1 \leq i \leq m$ into the string of layers $<< \mathcal{G}_i^1 >, < \mathcal{G}_i^2 >, ..., < \mathcal{G}_i^w >; 1 \leq i \leq m >$ according to the given $\prec^h$, i.e., each $.\{h\}^1_{\mathcal{G}_i^1}, \{h\}^2_{\mathcal{G}_i^2}, ..., \{h\}^w_{\mathcal{G}_i^w}$ is the full set of choice functions of the corresponding layer.

2. **return** the string of layers $<< \mathcal{G}_i^1 >, < \mathcal{G}_i^2 >, ..., < \mathcal{G}_i^w >>$.

Usage of the method *SPLIT_layer* requires assertion that the distribution of the set $\{h\}_{\mathcal{G}_i}$ into collection of subsets $\{h\}^1_{\mathcal{G}_i^1}, \{h\}^2_{\mathcal{G}_i^2}, ..., \{h\}^w_{\mathcal{G}_i^w}$ is in accordance with splitting the indexed family $< \mathcal{G}_i >, 1 \leq i \leq m$, into collection of layers $<< \mathcal{G}_i^1 >, < \mathcal{G}_i^2 >, ..., < \mathcal{G}_i^w >>$. This method is efficient if w is not a big number and if the sets $\{h\}^1_{\mathcal{G}_i^1}, \{h\}^2_{\mathcal{G}_i^2}, ..., \{h\}^w_{\mathcal{G}_i^w}$ are specified by means of partial validation of the values $h(j)$ for $j \in J$, $J \subset \{1, 2, ..., m\}$. The method is shown in the following example.

Example 7.1.1 *Given is the set* $\{h'\}$ *of all the choice bijections of* n. *We define the piecewise lexical order* $\prec^h$ *as follows:*
(i) $w = 4$,
(ii) the set $\{h\}^1_{\mathcal{G}_i^1}$ *contains the choice bijections that* $h(1) > n/2$ *and* $h(2) > n/2$,

(iii) the set $\{h\}^2_{\mathcal{G}_i^2}$ *contains the choice bijections that* $h(1) > n/2$ *and* $h(2) \leq n/2$,

(iv) the set $\{h\}^3_{\mathcal{G}_i^3}$ *contains the choice bijections that* $h(1) \leq n/2$ *and* $h(2) \leq n/2$,

(v) the set $\{h\}^4_{\mathcal{G}_i^4}$ *contains the choice bijections that* $h(1) \leq n/2$ *and* $h(2) > n/2$.

For the model S, we have the indexed family $< \mathcal{E}_i >, 1 \leq i \leq n$. Then, $\mathcal{E}_1^1 = \mathcal{E}_2^1 = \{1, 2, ..., n/2\}$, $\mathcal{E}_1^2 = \mathcal{E}_2^2 = \{n/2+1, n/2+2, ..., n\}$. We make the following layers:
$< \mathcal{G}_i^1 > = < \mathcal{E}_1^2, \mathcal{E}_2^2, \mathcal{E}_3, ..., \mathcal{E}_n >, < \mathcal{G}_i^2 > = < \mathcal{E}_1^2, \mathcal{E}_2^1, \mathcal{E}_3, ..., \mathcal{E}_n >,$
$< \mathcal{G}_i^3 > = < \mathcal{E}_1^1, \mathcal{E}_2^1, \mathcal{E}_3, ..., \mathcal{E}_n >, < \mathcal{G}_i^4 > = < \mathcal{E}_1^1, \mathcal{E}_2^2, \mathcal{E}_3, ..., \mathcal{E}_n >$. The string of the layers $<< \mathcal{G}_i^1 >, < \mathcal{G}_i^2 >, < \mathcal{G}_i^3 >, < \mathcal{G}_i^4 >>$ represents the piecewise lexical order $\prec^h$.

□

Usage of the method *SPLIT_partial_mapping_f* requires assertion that the set of choice functions $\{h\}_{\mathcal{G}_i}$ is partitioned by the partial mappings $\{f\}_{\mathcal{G}_j}$. Observe, if the hierarchical model

$$\begin{cases} \textit{System for } J \textit{ generation} \\ << \mathcal{G}_j >,\ j \in J; W^{f}(j);\ \{f\}_{\mathcal{G}_j} > \\ << \mathcal{G}_i >, 1 \le i \le m; W;\ \{h\}_{\mathcal{G}_i} > \end{cases}$$

is given, then it can be seen as the product of the method *SPLIT*. The method *SPLIT_partial_mapping_f* is used if $w >> 1$ and if we have more complete and more detailed specification of the requested order than for the case *SPLIT_layer*. For the method *SPLIT_partial_mapping_f*, we use the algorithms $FRIST(f)$, $FRIST(h^f, f)$, $NEXT(f)$, $NEXT(h^f, f)$, $LAST(f)$, $LAST(h^f, f)$.

Method SPLIT_partial_mapping_f

Input: the model $S = S_U$, $\prec^h$.

1. Generate an indexed subfamily $<< \mathcal{G}_j >$, for every $j \in J >$, of the family $< \mathcal{G}_i >$, $1 \le i \le m$, $J \subset \{1, 2, ..., m\}$ that each $\{h\}^k_{\mathcal{G}_i^k}$ contains all the extensions of a choice function f^k of the subfamily $<< \mathcal{G}_j >$, for every $j \in J >$.
2. Generate consecutively choice functions f^k.
3. **return** the generated choice functions $\{f\}_{\mathcal{G}_j}$ according to $\prec^h$.

Example 7.1.2 *Implement the method SPLIT_partial_mapping_f for the following case. The set of all the increasing choice functions $\{h\}$ for $m = 4$ and $n = 7$. The order $\prec^h$ is defined as follows: $\{h\} =< \{h\}^1_{\mathcal{P}_i}$, $\{h\}^2_{\mathcal{P}_i}$, $\{h\}^3_{\mathcal{P}_i}$, $\{h\}^4_{\mathcal{P}_i} >$, where $\{h\}^1_{\mathcal{P}_i}$ is a subset of $\{h\}$ that $h(3) = 5$, $\{h\}^2_{\mathcal{P}_i}$ is a subset of $\{h\}$ that $h(3) = 4$, $\{h\}^3_{\mathcal{P}_i}$ is a subset of $\{h\}$ that $h(3) = 6$ and $\{h\}^4_{\mathcal{P}_i}$ is a subset of $\{h\}$ that $h(3) = 3$.*

The system for J generation is defined as specified in the above example. Then, we have the following models S_j

$S_1 :< \mathcal{P}_1 = \{1,2,3\}$, $\mathcal{P}_2 = \{2,3,4\}$, $\mathcal{P}_3 = \{5\}$, $\mathcal{P}_4 = \{6,7\}$, $1 \le i \le j$; W given in (1.1); $\{h\}^1_{\mathcal{P}_i} >$;

$S_2 :< \mathcal{P}_1 = \{1,2\}$, $\mathcal{P}_2 = \{2,3\}$, $\mathcal{P}_3 = \{4\}$, $\mathcal{P}_4 = \{5, 6, 7\}$, $1 \le i \le j$; W given in (1.1); $\{h\}^2_{\mathcal{P}_i} >$;

$S_3 :< \mathcal{P}_1 = \{1,2,3,4\}$, $\mathcal{P}_2 = \{2,3,4,5\}$, $\mathcal{P}_3 = \{6\}$, $\mathcal{P}_4 = \{7\}$, $1 \le i \le j$; W given in (1.1); $\{h\}^3_{\mathcal{P}_i} >$;

$S_4 :< \mathcal{P}_1 = \{1\}$, $\mathcal{P}_2 = \{2\}$, $\mathcal{P}_3 = \{3\}$, $\mathcal{P}_4 = \{4,5,6,7\}$, $1 \le i \le j$; W given in (1.1); $\{h\}^1_{\mathcal{P}_i} >$.

□

The above example shows that starting from the concept of the partial mapping, we can get the model equivalent to the result obtained by

SPLIT_layer.t. Sometime it may not be so easy since the specification of the indexed sets for $i > \max(J)$ can be a difficult task. Then, the full structure of the hierarchical systems should be applied.

The method *SPLIT_layer* bases on the splitting indexed family into layers given in [19].
The methods *SPLIT_partial_mapping_f* and *SPLIT_RANK* are developed basing on the given theory and methodology.

Observe, a method *SPLIT* generates models corresponding to the subsets $\{h\}^1_{\mathcal{G}_i^1}$, $\{h\}^2_{\mathcal{G}_i^2}$, ..., $\{h\}^w_{\mathcal{G}_i^w}$. Then, each such subset possess a continuous rank range. Therefore, the general methods for sequential generation of the set of choice functions $\{h\}_{\mathcal{G}_i}$ using the lexical order can be applied to each subset and simultaneously to each corresponding model.

Therefore, the sequential generation system for fixed S and $\prec^h$ is an object that contains the methods $< Methods >$ developed or used in the previous section for the lexical generation and the proper method *SPLIT*.

If a given set of choice functions $\{h'\}_{\mathcal{G}_i}$ to be generated is not symmetric, then we have to extend our considerations.

Let a given asymmetric set of choice functions $\{h'\}_{\mathcal{G}_i}$ be distributed by a given piecewise lexical order $\prec^h$ into collection of subsets $\{\{h\}^k_{\mathcal{G}_i^k}\}$, $1 \leq k \leq w$. Observe, symmetry of subset $\{h\}^k_{\mathcal{G}_i^k}$ is not determined. Then, the piecewise lexical order $\prec^h$ is sequencing the collection of subsets $\{\{h\}^k_{\mathcal{G}_i^k}\}$ into a string $< \{h\}^1_{\mathcal{G}_i^1}, \{h\}^2_{\mathcal{G}_i^2}, ..., \{h\}^w_{\mathcal{G}_i^w} >$.

Definition 7.1.2 *We say that a given piecewise lexical order $\prec^h$ is clustering the symmetric subsets, if all the asymmetric subsets are the predecessors of all the symmetric subsets or if the all asymmetric subsets are the successors of all the symmetric subsets.*

If a given lexical order $\prec^h$ is clustering the symmetric subsets, then we can build a model S' that the corresponding set of choice functions $\{h'\}_{\mathcal{G}_i'}$ equals union of all the symmetric subsets $\{h\}^k_{\mathcal{G}_i^k}$. Then, for the piecewise generation of $\{h'\}_{\mathcal{G}_i'}$ we follow the above given methodology for symmetric sets $\{h'\}_{\mathcal{G}_i}$. For the generation of asymmetric subsets $\{h\}^k_{\mathcal{G}_i^k}$, we can use the model S and the algorithm RANSEQ1 given in the previous section.

If a given lexical order $\prec^h$ is not clustering the symmetric subsets, then we do not solve the generation problem by building a model S' that the corresponding set of choice functions $\{h'\}_{\mathcal{G}_i'}$ equals union of all the symmetric subsets $\{h\}^k_{\mathcal{G}_i^k}$. Then, the simplest solution is to use the model S and the algorithm RANSEQ1 for each subset $\{h\}^k_{\mathcal{G}_i^k}$. Complete method *SPLIT_RANK* can be specified as follows.

Method SPLIT_RANK
Input: the model $S = S_R$, $\prec^h$.

1. For each consecutive $\{h\}^k_{\mathcal{G}^k_i}$ determine the corresponding rank range $R(\{h\}^k_{\mathcal{G}^k_i})$.

2. **return** the ordered string $< R(\{h\}^1_{\mathcal{G}^1_i}), R(\{h\}^2_{\mathcal{G}^2_i}), ..., R(\{h\}^w_{\mathcal{G}^w_i}) >$.

The method *SPLIT_RANK* is the most easy for implementation if the order $\prec^h$ is specified explicitly by the string of ranks. Then instead finding the rank ranges $R(\{h\}^k_{\mathcal{G}^k_i})$ we apply the algorithm *UNRANK* to each individual choice function.

7.2 Distributed generation in MIMD systems

Let P be the number of distributed processors used. The general distributed generation system is the following object $< S, NC, < Methods >>$, where S is unranking or ranking model, NC determines the number of submodels and their cardinals, the subsystem $< Methods >$ contains the methods used for the sequential generation. The subsystem $< Methods >$ contains one of the method *SPLIT* for splitting the model S into submodels S_1, S_2, ..., S_w and for dedicating the submodels into the processors p_1, p_2, ..., p_P. For the proper method *SPLIT* used, we have here $\prec^h = <^h$. The subsystem NC included in $< Methods >$ decides how many submodels S_1, S_2, ..., S_w are to be generated and/or evaluates the cardinals of the corresponding sets $\{h\}^1_{\mathcal{G}^1_i}$, $\{h\}^2_{\mathcal{G}^2_i}$, ..., $\{h\}^w_{\mathcal{G}^w_i}$. Then, the subsystem NC distributes submodels $S_1, S_2, ..., S_w$ into the distributed processors p_1, p_2, ..., p_P. Each processor p_j generates and processes choice functions of the dedicated submodels in the way that is specified in the previous section.

We use basically the following NC systems.

NC_balanced

Input: $w = P$,

Output: each processor generates and processes the set of choice functions $\{h\}^j_{\mathcal{G}^j_i}$ that $| \{h\}^j_{\mathcal{G}^j_i} | = | \{h\}_{\mathcal{G}_i} | / P$.

Method:

1. The model S is partitioned into the submodels S_1, S_2, ..., S_w using a method *SPLIT*[1].

2. Each model S_j is dedicated to the processor p_j.

3. **for** $j \longleftarrow 1$ **to** w **do concurrently**

3.1. Processor p_j generates and processes the choice functions $\{h\}^j_{\mathcal{G}^j_i}$.

Since the generated choice functions are to be processed, so for certain applications we may have big differences in the time of processing, when the

---

[1] Each method *SPLIT* can be used.

method *NC_balanced* is used. We develop the method *NC_decreasing* in order to balance the time of processing all the processors used instead of making the balance of the numbers of the choice functions to be generated in each processor .

NC_decreasing

Input: $w >> P$,

Output: each processor generates and processes a number of the subsets $\{h\}^j_{\mathcal{G}^j_i}$.

Method:

1. **while** a consecutive model S_k can be generated **do**

1.1. Generate consecutive model S_k that $\mid \{h\}^k_{\mathcal{G}^j_i} \mid \geq \mid \{h\}^{k-1}_{\mathcal{G}^{k-1}_i} \mid$ using a method *SPLIT*.

1.2. Dedicate currently generated model S_k to a free processor p_j.

1.3. Processor p_j generates and processes the choice functions $\{h\}^k_{\mathcal{G}^k_i}$

Developing the method *NC_decreasing* we have assumed that the time of processing and of the generation of smaller and smaller packages is shorter and shorter. That is statistically *TRUE*. Nevertheless, it may happen that the generation and processing a subset of choice functions whose cardinal is small requires much more time than the generation and processing much greater subset. Then, the method *NC_decreasing* does not produce a good balance of the running time for each processor.

The method *NC_redistributed* gives better balance of the time of processing since there is feed−back for redistribution of the subtasks if the running times for particular processors can much differ.

NC_redistributed

Input: We have $P >> 1$.

Output: Each processor generates and processes a random number of choice functions.

Method: At the beginning the number w equals P. Each $j-$th processor generates and processes a number of choice functions of the model S_k. If all the choice functions $\{h\}^k_{\mathcal{G}^k_i}$ are generated and processed, then the $j-$th processor that possesses the greatest number of choice functions still to be generated partitions its left task in two parts and the first part is dedicated to the free processor, while the second part is to be processed by itself. The cardinals $\mid \{h\}^k_{\mathcal{G}^k_i} \mid$ are random.

1. The model S is partitioned into submodels S_1, S_2, ..., S_w using the proper method *SPLIT*[2]

---

[2]The most suitable is the method *SPLIT_layer*, nevertheless other *SPLIT* can also be used.

2. Dedicate submodels S_j into the processors p_j.

3. **repeat**

3.1. **if** processor p_k finished the generation and processing the choice functions of the dedicated model **then** find processor p_j that it has the greatest number of the choice functions still to be generated.

3.2. The left part of the choice functions $(\{h\}^j_{\mathcal{G}_i^j})'$ is to be partitioned in two parts $(\{h\}^j_{\mathcal{G}_i^j})'_1$ and $(\{h\}^j_{\mathcal{G}_i^j})'_2$.

3.3. Make the subsystems S'_{j1} and S'_{j2} corresponding to $(\{h\}^j_{\mathcal{G}_i^j})'_1$ and $(\{h\}^j_{\mathcal{G}_i^j})'_2$, respectively.

3.4. Dedicate S'_{j1} to p_k.

3.5. Dedicate S'_{j2} to p_j.

3.6. Each processor p_j generates and processes choice functions of the dedicated model.

until all the choice functions $\{h\}_{\mathcal{G}_i}$ are generated.

The method *NC_redistributed* gives the best balance of time of processing for the MIMD systems. Nevertheless, the requirements as to communication are especially strong. That requirements were much stronger if a method *NC_redistributed* would be implemented by using *SPLIT_layer* or *SPLIT_partial_mapping* since full models of subsets would be sent between processors. Much weaker requirements as to communication we have if the method *SPLIT_ranking* is used. Then, the full model can be broadcast to all the processors and only the rank ranges must be transmitted and retransmitted to each processor. Obviously, if the continuous rank range characterize the model used, then the distribution and the redistribution are much easier for handling.

7.3 Parallel generation in SIMD systems

For parallel generation of the set $\{h\}_{\mathcal{G}_i}$ in a SIMD system we use widely the earlier concepts, except we have to make assertion that models S_j are created and choice functions are generated in the dedicated processor p_j. The natural requirement is to have the balanced cardinals of the corresponding sets $\{h\}^j_{\mathcal{G}_i^j}$. Therefore, the method *NC* used is the method *NC_balanced*. For splitting the model S into the models S_1, S_2, ..., S_P two methods *SPLIT_partial_mapping_f* and *SPLIT_ranking* are the most convenient, however for special applications the method *SPLIT_layer* could also be used.

7.3.1 *Generation using SPLIT_ranking*

We are concerned with the generation of a given set of choice functions $\{h'\}_{\mathcal{G}_i}$ assuming that the splitting model S is to be done by implementing the method *SPLIT_ranking*. If the set $\{h'\}_{\mathcal{G}_i}$ is symmetric and if the rank range $R(\{h'\})$ is not continuous in $R(h)$, then the most obvious and general method requires building the model S'_R that the corresponding rank range $R(\{h'\}_{\mathcal{G}'_i})$ is continuous in $R(h')$. Then, the following algorithms can be used.

Algorithm RANPAR1(h') (RANking PARallel generation 1)

Input: the rank range $R(\{h'\}_{\mathcal{A}_i}) = (1, \mid \{h'\}_{\mathcal{A}_i} \mid)$, P is the number of processors used. Given is the table D representing the structure of $R(\{h'\}_{\mathcal{A}_i})$.

Output: The set of choice functions $\{h'\}_{\mathcal{A}_i}$ is generated in the SIMD system that each processor p_i generates $\mid \{h'\}_{\mathcal{A}_i} \mid /P$ choice functions.

Method: Each processor p_i knows its index i

1. **for** $i \longleftarrow 1$ **to** P **do in parallel**
1.1. $R(h_i) \longleftarrow (i-1).\mid \{h'\}_{\mathcal{A}_i} \mid /P + 1$
1.2. $UNRANK(h_i)$
1.3. **for** $j \longleftarrow 1$ **to** $\mid \{h'\}_{\mathcal{A}_i} \mid /P - 1$ **do**
1.4. call $NEXT(h_i)$.

Each processors p_i performs only one time procedure $UNRANK$ and $\mid \{h'\}_{\mathcal{A}_i} \mid /P - 1$ times the procedure $NEXT$. If the cardinal $\mid \{h'\}_{\mathcal{A}_i} \mid$ can not be partitioned into $\mid \{h'\}_{\mathcal{A}_i} \mid /P$ equal parts, then the last processor generates only r choice functions, where r is the rest of the division $\mid \{h'\}_{\mathcal{A}_i} \mid /P$. If the rank range $R(\{h'\}_{\mathcal{A}_i}) \neq (1, \mid \{h'\}_{\mathcal{A}_i} \mid)$ but it is continuous in $R(h)$, then we do not need to create the model S'_R but the model S_R is enough good for handling the parallel generation. Then, the algorithm RUNPAR2 can be used instead of RUNPAR1.

Algorithm RANPAR2(h') (RANking PARallel generation 2)

Input: the rank range $R(\{h'\}_{\mathcal{A}_i}) = (R_{\min}\{h'\}, e)$.

The other conditioning is the same as for the algorithm RANPAR1.

1. **for** $i \longleftarrow 1$ **to** P **do in parallel**
1.1. $R(h_i) \longleftarrow (i-1).e/P + R_{\min}\{h'\}$
1.2. $UNRANK(h_i)$
1.3. **for** $j \longleftarrow 1$ **to** $e/P - 1$ **do**
1.4. call $NEXT(h_i)$.

The goal of the following example is to show that creation of the model S'_R is essential for handing the parallel generation of the set of choice functions in SIMD system.

Example 7.3.1 *Given is the following unranking model* $S'_U =<< \mathcal{A}_i >$, $1 \le i \le 4$, $n = 8$; W *given in* (1.1); W_1 *given in* (7.1); $\{h'\}_{\mathcal{A}_i} >$. *The task is to make parallel generation of the set* $\{h'\}_{\mathcal{A}_i}$ *using SIMD system that* $P = 7$.

$$W_1 : h'(3) = h'(2) + 1 \tag{7.1}$$

We can treat the set $\{h'\}_{\mathcal{A}_i}$ as the subset of the corresponding full set of increasing choice functions $\{h\}_{\mathcal{A}_i}$ of the model $S_U =<< A_i >$, $1 \le i \le 4$, $n = 8$; W given in (1.1); $\{h\}_{\mathcal{A}_i} >$. Then the table D is the DCM table as follows:

$$\left|\begin{array}{cccc} 35 & 15 & 5 & 1 \\ 20 & 10 & 4 & 1 \\ 10 & 6 & 3 & 1 \\ 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right|$$

The rank range $R(\{h'\}_{\mathcal{A}_i})$ in $R(h)$ is as follows: $R(\{h'\}_{\mathcal{A}_i}) =<$ (1, 5), (16, 4), (26, 3), (32, 2), (35, 1), (36, 4), (46, 3), (52, 2), (55, 1), (56, 3), (62, 2), (65, 1), (66, 2), (69, 1), (70, 1) > .

The rank range $R(\{h'\}_{\mathcal{A}_i})$ in $R(h)$ is discontinuous and very difficult to handle. Even if we solve the problem of partitioning this rank range into 7 subranges, then the generation algorithm would be very inefficient, since in every step at least one processor would run the $UNRANK$ algorithm.

The set $\{h'\}_{\mathcal{A}_i}$ is symmetric.

Therefore, we can use the model $<< \mathcal{A}_i >$, $1 \le i \le 4$, $n = 8$; W given in (1.1); W_1 given in (7.1); $\{h'\}_{\mathcal{A}_i} >$. So, the table D that represents the structure of the set $\{h'\}_{\mathcal{A}_i}$ is the table W(1.5)DCM.

$$\text{W(1.5)DCM}= \left|\begin{array}{cccc} 15 & 5 & 5 & 1 \\ 10 & 4 & 4 & 1 \\ 6 & 3 & 3 & 1 \\ 3 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right| .$$

Then, the rank range $R(\{h'\}_{\mathcal{A}_i})$ is continuous in $R(h')$. Therefore, the problem of the generation of the set $R(\{h'\}_{\mathcal{A}_i})$ can easily be handled. We can use the algorithm RANPAR1.

□

If the set of choice functions $\{h'\}_{\mathcal{G}_i}$ is not symmetric, then the algorithms RANPAR1 and RANPAR2 can not be used directly. Then, we can build the model S' representing the symmetric part of $\{h'\}_{\mathcal{G}_i}$. Simultaneously, the number of discontinuous asymmetric parts of $\{h'\}_{\mathcal{G}_i}$ can also be diminished. For generating and for processing the symmetric subset of $\{h'\}_{\mathcal{G}_i}$ we apply algorithms RANPAR1 or RANPAR2 Then, each asymmetric part of $\{h'\}_{\mathcal{G}_i}$ should be generated and processed using the algorithm RANPAR2. That

approach is reasonable if cardinals of each asymmetric subset of $\{h'\}_{\mathcal{G}_i}$ possess the cardinal $|\{h'\}_{\mathcal{G}_i}| >> P$.

If asymmetric subsets $\{h''\}_{\mathcal{G}_i}$ are included in $\{h'\}_{\mathcal{G}_i}$ and if the cardinals of each continuous part $\{h''\}_{\mathcal{G}_i}$ are fixed, then we can also distribute the generation of the discontinuous subsets into processors, so that each processor would generate choice functions of the dedicated asymmetric subsets.

7.3.2 Generation using *SPLIT_layer*

Suppose, given is the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$, that the set of choice functions $\{h\}_{\mathcal{G}_i} = \{h'\}_{\mathcal{G}_i}$ can be partitioned into w layers $\{\mathcal{G}_i^{j,l}\}$, $1 \leq i \leq m$ and all the cardinals $|\{h_l^l\}_{\mathcal{G}_i^{j,l}}|$ are fixed and equal e. Then, the set $\{h'\}_{\mathcal{G}_i}$ can be generated in parallel using SIMD system. The general algorithm LAYPAR for this generation uses *SPLIT_layer*, however, different implementations can also be used.

Algorithm LAYPAR(h') (*SPLIT_layer* for PARallel generation)

Input: The unranking representation model S_U.

Method: With denotation $< \mathcal{G}_i^{j,l} >$, $1 \leq i \leq m$, we mean the $j-$th layer dedicated to the processor p_l.

1. Split the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$, into w layers that $w \geq P$.
2. Dedicate $k = w/P$ layers to each processor p_i
3. **for** $l \longleftarrow 1$ **to** P **do in parallel**

3.1. **for** $j \longleftarrow 1$ **to** k **do**

3.2. call $FIRST$ for the currently processed layer $< \mathcal{G}_i^{j,l} >$, $1 \leq i \leq m$.

3.3. **repeat**

3.3.1. call $NEXT(h_j^l)$

3.3.1. Process the choice function h_j^l

until the last choice function of the layer $< \mathcal{G}_i^{j,l} >$, $1 \leq i \leq m$ is generated.

The algorithm LAYPAR is suitable for the generation of choice bijections, then it is an easy task to split the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$ into layers that the cardinal of each layer $< \mathcal{G}_i^{j,l} >$, $1 \leq i \leq m$ is fixed.

7.3.3 Generation using *SPLIT_partial_mapping*

We can also make the generation of a given set of choice functions $\{h'\}_{\mathcal{G}_i}$ in SIMD system using *SPLIT_partial_mapping* as the main vehicle. Similarly, as for the previous section we need to make assertion that cardinals of the sets of choice functions $\{h^f\}_{\mathcal{G}_i}$ equal e for each partial mapping f. For numerous practical applications that holds. For instance, if we generate the set of all the permutations of m out of n, then each partial mapping for the

subfamily $< \mathcal{E}_i >$, $1 \leq i \leq t$, $t < m$ possess a fixed number of extensions to full choice functions of the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$. Moreover, if the set of compositions or partitions with fixed cardinal of each block is to be generated, then for each partial mapping f_r the number of extensions of f to full choice functions h is dependent only on r, where f_r is a partial choice mapping the subfamily $< S_i >$, $1 \leq i \leq k_1 + k_2 + ..+ k_r$, see the model 2.38. The methods developed by using *SPLIT_partial_mapping* as the main vehicle are unranking methods, so they only require the basic unranking model S_U. Nevertheless, the problem is assignment of a set of partial mappings f to each processor p_l. For making this assignment we can use any method of splitting the set $\{f\}_{\mathcal{G}_i'}$ into P subsets $\{f\}^l_{\mathcal{G}_i'}$ that the cardinals $|\{f\}^l_{\mathcal{G}_i'}|$ equal and then we assign each such subset $\{f\}^l_{\mathcal{G}_i'}$ to the processor p_i.

Algorithm MAPAR(h') (*SPLIT_partial_mapping* for PARallel generation)

Input: The unranking representation model S_U.

Method: With denotation $< \mathcal{G}_i' >$, $1 \leq i \leq t$, we mean the subfamily of the indexed family $< \mathcal{G}_i >$, $1 \leq i \leq m$.

1. Create the indexed subfamily $< \mathcal{G}_i' >$, $1 \leq i \leq t$ that each choice function f possess the same number of extensions to the choice functions h of the family $< \mathcal{G}_i >$, $1 \leq i \leq m$. A number of choice functions f_l^j dedicated to p_l processor equals w.

2. **for** $l \longleftarrow 1$ **to** P **do in parallel**

2.1. **for** $j \longleftarrow 1$ **to** w **do**

2.2. **while** there is still an extension of f_l^j to h **do**

2.2.1. Generate the next extension h_l^j of f_l^j.

2.2.2. Process the choice function h_l^j

For generating an extension h_l^j of f_l^j, we use the modified algorithm *NEXT* that is restricted into the domain $\{t+1 \div m\}$.

7.4 Conclusions

The given in this chapter methods and algorithms use the investigated in the previous chapter models and rutines for developing sequential or distributed or parallel exhaustive generation of arbitrary sets of choice functions. The developed methods posses lowest time complexity when unranking methods are used. If we have to use ranking methods, then the main concern is to make assertion that the algorithms $RANK$ and $UNRANK$ would run a minimum number of times. For making that assertion, we use widely Ranking Theorem and developed earlier methodology for modelling and for the generation.

8
Miscellanea

The developed methodology for modeling and for the generation of combinatorial objects has concerned mostly the classical problems of the combinatorics. New problems, we have investigated till now posses rather auxiliary meaning to be used for better performance of the classical generation jobs. The goal of this chapter is to demonstrate the great potential of the developed theory for investigation quite new problems and their solving with the aid of the given or modified general methodology. Currently such jobs emerge from many branches of computer science mainly concerning the optimization theory, in particularly genetic and evolutionary algorithms [48]. Till now formulations and solutions of these additional generation tasks are very specific and contain only special cases of the general problems [45]. The presented in this text approach enables us to make the formulation of the problems and algorithms for their solving much more general. Apart from the optimization theory there is also strong need for stating new combinatorial problems corresponding to many new tasks concerning cryptography. In this chapter, we will also demonstrate such tasks and show their solutions.

8.1 Indexed families and CF DATABASES

The developed here methods of modeling and of the generation of combinatorial objects is rather self-contained in term of potential of formulation and solving the generation tasks. Nevertheless, there are structural simi-

larities between CF DATABASES [22] and indexed families. There is also certain structural similarity between the exhaustive generation of the full set of choice functions and sorting to be performed in SELRAM [25] or in DSM [22]. These structural similarities enable us to observe certain mutual correspondence between searching and sorting on one side and generation of choice functions on the other side. An object in term of CF DATABASES can be seen as certain combinatorial object, moreover there is a correspondence between the methods of searching a record (object) and methods of the generation of a combinatorial object. In fact, if unranking model is reduced and the Q property holds, then there is also similarity between certain algorithms for generation of choice functions and algorithms for making a search in CF DATABASES. Since we have also developed random and semi random search performed in CF DATABASES, so our goal is to reuse their structure for random and semi random generation. Studying structural similarity between the search algorithms and the generation algorithms promotes deeper understanding the generation problems, we have for the models that the Q property does not hold. That understanding became quite helpful in development of the general generation algorithms when these models are concerned.

Let us examine in more detail the basic structural similarities between CF DATABASES and relative notions on one side and indexed families and their choice functions on the other side. For CF DATABASES the main structure is the string of characteristic functions $< g_1(o_j), g_2(o_j), ..., g_m(o_j) >$, while the indexed family $< \mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m >$ is the fundamental structure for the developed here theory. In fact, we have the structural correspondence, between the string of codomains $< \Gamma_1, \Gamma_2, ..., \Gamma_m >$ of the characteristic functions and $< \mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m >$, since $< \Gamma_1, \Gamma_2, ..., \Gamma_m >$ is an indexed family[1]. Then, the string of values $< g_1(o_j), g_2(o_j), ..., g_m(o_j) >$ for a fixed object o_j corresponds to the string of values $< h(1), h(2), ..., h(m) >$, hence it is a representation of $h \in \{h\}_{\mathcal{G}_i}$. Then, we observe a structural and functional correspondence between the following algorithms:

CF DATABASES	combinatorial objects
MIN	*FIRST*
MAX	*LAST*
IDNEXTMIN	*NEXT* if $<^h$ is the lexical order *

* if $<^h$ used is the anti-lexical order, then the algorithm *NEXT* corresponds to IDNEXTMAX.

The rutin EXISTENCE is the basic component of the algorithms defined for processing CF DATABASES in DSMs. We show now that similar

[1] For numerous instances instead of the correspondence $\Gamma_i \longleftrightarrow \mathcal{G}_i$, we have a correspondence $\Gamma_i \longleftrightarrow \overline{\mathcal{G}}_i$ or $\Gamma_i \longleftrightarrow \overline{\mathcal{G}}_i^*$.

EXISTENCE function is the basic rutin for the algorithms concerning the generation of choice functions.

Let $\gamma_{i_1} \in \Gamma_1$, $\gamma_{i_2} \in \Gamma_2$, ..., $\gamma_{i_r} \in \Gamma_r$, while $1 \leq i_1 < i_2 < ... < i_r < m$. The function EXISTENCE(γ_{i_1}, γ_{i_2}, ..., γ_{i_r}) returns value TRUE if there exists an object $o_j \in \mathcal{O}$ that $g_{i_1}(o_j) = \gamma_{i_1}, g_{i_2}(o_j) = \gamma_{i_2}, ..., g_{i_r}(o_j) = \gamma_{i_r}$, else it returns value FALSE, see [22] for more detailed examination.

Let us investigate a similar concept for the generation of the combinatorial objects. We use the following denotations: $q_{i_1} \in \mathcal{G}_{i_1}$, $q_{i_2} \in \mathcal{G}_{i_2}$, ..., $q_{i_r} \in \mathcal{G}_{i_r}$, we have also $1 \leq i_1 < i_2 < ... < i_r < m$. Instead writing $q_{i_1} \in \mathcal{G}_{i_1}$, $q_{i_2} \in \mathcal{G}_{i_2}$, ..., $q_{i_r} \in \mathcal{G}_{i_r}$, we can be more specific putting $q_{i_1} \in \overline{\mathcal{G}}_{i_1}$, $q_{i_2} \in \overline{\mathcal{G}}_{i_2}$, ..., $q_{i_r} \in \overline{\mathcal{G}}_{i_r}$ or $q_{i_1} \in \overline{\mathcal{G}}^*_{i_1}$, $q_{i_2} \in \overline{\mathcal{G}}^*_{i_2}$, ..., $q_{i_r} \in \overline{\mathcal{G}}^*_{i_r}$ depending on situation. The function EXISTENCE(q_{i_1}, q_{i_2}, ..., q_{i_r}) returns value TRUE if there exists a choice function $h \in \{h\}_{\mathcal{G}_i}$ that $h(i_1) = q_{i_1}$, $h(i_2) = q_{i_2}$, ..., $h(i_r) = q_{i_r}$.

The function EXISTENCE(γ_{i_1}, γ_{i_2}, ..., γ_{i_r}) possess the fixed body and it consumes time $O(1)$ if processing CF DATABASES is performed in DSMs, while there is no one function EXISTENCE(q_{i_1}, q_{i_2}, ..., q_{i_r}), we have numerous algorithms for its evaluation dependent on the given model, instead. Asymptotic complexities of such algorithms vary from $O(1)$ up to $O(n.m)$. Nevertheless, for different models and their fundamental properties and for different detailed algorithms there is a correspondence between EXISTENCE(γ_{i_1}, γ_{i_2}, ..., γ_{i_r}) and EXISTENCE(q_{i_1}, q_{i_2}, ..., q_{i_r}) . We can examine this correspondence observing the following algorithms.

The algorithms MIN and *FIRST* expressed by mean of EXISTENCE functions.

Input: the values γ_1, γ_2, ..., γ_i denote the elements of Γ_1, Γ_2, ..., Γ_i, respectively, while q_1, q_2, ..., q_i denote the minimal elements of $\mathcal{G}_1, \mathcal{G}_2$, ..., $\mathcal{G}_i$, correspondingly that the requirement W holds.

Output: the minimum $o_j \in \mathcal{O}$ that $o_j =< \gamma_1, \gamma_2, ..., \gamma_m >$ or the minimum choice function $h =< h(1), h(2), ..., h(m) >$, i.e., rank $R(h)$ according to the lexical order is minimum.

Method: In brackets, we specify steps to be performed for the generation purpose, while denotations without brackets concern processing CF DATABASES.

1. for $i \longleftarrow 1$ to m do

1.1. if EXISTENCE(γ_1, γ_2, ..., γ_i) =TRUE
(EXISTENCE(q_1, q_2, ..., q_i) =TRUE) then assign the minimum of Γ_1 into g_i (minimum of $\mathcal{G}_i$ into $h(i)$).

2. return $< \gamma_1, \gamma_2, ..., \gamma_m >$ ($< h(1), h(2), ..., h(m) >$).

Structural similarities between MAX and *LAST* can also be specified, respectively. We will present now the structural similarities between the algorithms IDNEXTMIN and *NEXT* .

The algorithms IDNEXTMIN and $NEXT$ expressed by means of EXISTENCE functions

Input:(i) the string of values of the characteristic functions that $g_1(o_j) = \gamma_1$, $g_2(o_j) = \gamma_2$, ..., $g_m(o_j) = \gamma_m$,

(ii) the choice function h that $h(1) = q_1$, $h(2) = q_2$, ..., $h(m) = q_m$;

Output: (i) the string of values of the characteristic functions $g_1(o'_j) = g_1(o_j)$, $g_2(o'_j) = g_2(o_j)$, ..., $g_{i-1}(o'_j) = g_{i-1}(o_j)$, $g_i(o'_j) = \beta_i$, $g_{i+1}(o'_j) = \delta_{i+1}$, $g_{i+2}(o'_j) = \delta_{i+2}$, ..., $g_m(o'_j) = \delta_m$, or the next minimum choice function $h' =< h'(1) = h(1), h'(2) = h(2), ..., h'(i-1) = h(i-1), h'(i) = p_i, h'(i+1) = t_{i+1}, ..., h'(m) = t_m >$

Method: (i) with denotation β_i we mean the minimum value from the set Γ_i that $\beta_i > g_i(o_j)$, while δ_k denotes the minimum value from the set Γ_k

(ii) with denotation p_i we mean the minimum value from the set $\overline{\mathcal{G}_i^*}$ that $p_i > h(i)$, while t_k denotes the minimum value from the set $\overline{\mathcal{G}}_k$.

1. $i \longleftarrow m$;
2. **while** EXISTENCE(γ_1, γ_2, ..., γ_{i-1}, β_i) = FALSE (EXISTENCE(q_1, q_2, ..., q_{i-1}, p_i) =FALSE) **do**
2.1. $i \longleftarrow i - 1$;
3. $g_i(o'_j) \longleftarrow \beta_i$; $(h(i) \longleftarrow p_i;)$
4. **for** $k \longleftarrow i + 1$ **to** m **do**
4.1. $g_k(o'_j) \longleftarrow \delta_k$; $(h(k) \longleftarrow t_k;)$
5. **return** $< g_1(o'_j), g_2(o'_j), ..., g_m(o'_j) >$ $(< h'(1), h'(2), ..., h'(m) >)$;

For choice functions $h = < h(1), h(2), ..., h(m) >$ or for strings of values of characteristic functions $< \gamma_1, \gamma_2, ..., \gamma_m >$, we can use the common notion "string" in order to express the concepts of common head and tail.

Let $\{w_1, w_2, ..., w_z\}$ be a set of strings over a given alphabet that $w_i =< \alpha_1^i, \alpha_2^i, ..., \alpha_r^i >$. Suppose, w_i and w_j are two strings that $w_i =< \alpha_1^i, \alpha_2^i, ..., \alpha_r^i >$ and $w_j =< \alpha_1^j, \alpha_2^j, ..., \alpha_s^j >$ and $\alpha_1^i = \alpha_1^j$, $\alpha_2^i = \alpha_2^j$, ..., $\alpha_k^i = \alpha_k^j$, while $\alpha_{k+1}^i \neq \alpha_{k+1}^j$. The common prefix, i.e., $< \alpha_1^i = \alpha_1^j, \alpha_2^i = \alpha_2^j, ..., \alpha_k^i = \alpha_k^j >$ for w_i and w_j, we call the common head. We have the following observation:

Remark 8.1.1 *If w_j is the next greater than w_i, then w_j and w_i posses the longest common head out of all strings from the subset of $\{w_1, w_2, ..., w_z\}$ that contains the strings greater than w_i.*

Since, the above given observation concerns both the specified set of records and the set of choice functions of a given model, so it concerns both generation of choice functions and sorting records in DSMs.

The structure of the presented here algorithms is the foundation for development other generating algorithms even if they are to be performed for the models that the Q property does not hold.

8.2 $NEXT$ when the Q property does not hold

There are situations that making a model S_U possessing the Q property is really a difficult problem. [2] Then, we have to manage with models that the Q property does not hold, that happens mostly if we implement genetic algorithms. Such problems are extensively researched [48], [45]. For instance, we have shown the task of creation the model S_U in Example 5.2.5 that the Q property can not hold. We are concerned now with the general backtracking algorithm $NEXTWQ$ that is a generalization of the algorithm $NEXT$ and it is usable for any model including those models that the Q property does not hold.

Algorithm $NEXTWQ$ (general $NEXT$ for the models Without Q property)

Input: A given reduced model S_U that the Q property does not hold; the table IA represents current valid choice function h.

Output: The table $IA[i]$ representing the custom form of the next choice function.

Method:

Expression D : $q = \min\{\overline{\mathcal{G}_i^*}\} - \{IA[i]\}$, i.e., q is as small as possible and $q \in \mathcal{G}_i$ and $q > IA[i]$ and q satisfies $W \wedge W_1$, see Definition 3.2.3;

Expression U : $q = \min\{\mathcal{G}_i^*\}$, i.e., q is as small as possible and $q \in \mathcal{G}_i$ and q satisfies $W \wedge W_1$, see Definition 3.2.4;

Function $DOWN$: returns the element q from the set $\{\overline{\mathcal{G}_i^*} = \{IA[i]\}\}$ that satisfies D or it returns 0 if the element satisfying D does not exist;

if the element q satisfies D **then**

$DWON \longleftarrow q$ that q satisfies D;

else

$DOWN \longleftarrow 0$;

Function UP : returns an element of the set $\mathcal{G}_i$ that satisfies U or it returns 0 if the element satisfying U does not exist;

if the element q satisfies U **then**

$UP \longleftarrow q$ that q satisfies U;

else

$UP \longleftarrow 0$;

- -

1. $i \longleftarrow m$;

2. **while** $i \leq m$ or $Sem = FALSE$ **do**

2.1. **case** Sem **of**

$FALSE$: **if** $DOWN \neq 0$ **then** $IA[i] \longleftarrow DOWN$;

$TRUE$: **if** $UP \neq 0$ **then** $IA[i] \longleftarrow UP$;

---

[2] In fact, it is an open problem whether model that the Q property holds always exist If response is positive, then another problem is the size of input data we have to have in order to make assertion that the Q property always holds.

2.2. if $IA[i]$ has been updated at the step 2.1. **then** $i \longleftarrow i+1$; $Sem \longleftarrow TRUE$; **else** $i \longleftarrow i-1$; $Sem \longleftarrow FALSE$;
3. **return** IA.

Asymptotic complexity of the algorithm $NEXTWQ$ is $O(2^m)$, since performance of the step 2. takes at worst $O(2^m)$ steps. On the other hand performance of the algorithm can be completed in time $\Omega(1)$. For an average case, we can have complexity $O(2^m)$ or it can be $O(m^k)$, where k is a fixed integer dependent on the model S_U. Frankly speaking, we can have models S_U that the Q property does not hold only for very few cases on the other hand, we can have models S_U that the Q property does not hold nearly for all possible partial choice functions. Obviously, if the number of partial choice functions that the Q property does not hold is greater, then we have greater asymptotic complexity for an average case of the algorithm $NEXTWQ$.

If we apply the general algorithm $NEXTWQ$ for any special case, then the real difficulties come from huge number of possible partial choice mapping in comparing with the number of full choice functions of such models. That means, we have to perform many times backtracking and looking forward steps in comparing with the situation that the generation process is successfully completed without any backtracking. In order to diminish complexity of the generation of the next choice function, we have to examine the detailed properties of a given model with respect to possibility of making "shortcuts" that would reduce the number of backtracking and looking forward steps. Unfortunately, the asymptotic and time complexities of such dedicated algorithms are strongly effected by fitting a proper dedicated algorithm to a given model. Therefore, the number of such dedicated algorithms is much greater than the number of dedicated algorithms for the models possessing the Q property, so we are not able to consider here all the detailed cases and we can not design a proper variant of the $NEXTWQ$ algorithm for each case.

On the other hand the general structure of the algorithm $NEXTWQ$ describes the philosophy of the shortcuts and it is very helpful for designing a suitable variant of the general algorithm. So, we present now the general structure of the algorithm $NEXTWQ$.

The structure of the algorithm $NEXTWQ$.

Input: a choice function $h \in \{h\}_{\mathcal{G}_i}$

Output: the choice function h' that is the next greater than h (with respect to the lexical order)

Procedure SHORTCUT

can be applied only if EXISTENCE$(q_1, q_2, ..., q_i, p_{i+1}, ..., p_k)$ =FALSE, it takes the string $< q_1, q_2, ..., q_i, p_{i+1}, ..., p_k >$ and the model S_U as input and it returns the value z as output.

We have two cases:

(i) $z = s \leq i$ in the case of necessity of modification of the head,
(ii) $z = t \leq k$ and $z > i$ is returned if the tail needs modification.
{the body of *SHORCUT* depends on properties of the model S_U }.

---

1. $i \longleftarrow m-1; k \longleftarrow m$;
2. $p_k \longleftarrow UP$;
3.**while** $i < m$ or $k < m$ **do**
3.1. **if** EXISTENCE$(q_1, q_2, ..., q_i, p_{i+1}, ..., p_k)$ =FALSE **then**
{there is a new minimum prefix of the tail $< p'_{i+1}, ..., p'_s >$, $s \leq k$ or there is a new common head $< q_1, q_2, ..., q_t >$, $t \leq i$}
3.1.1.1. *SHORTCUT*;
3.1.1.2. **case** z **of**
3.1.1.2.1.1. $z < t : i \longleftarrow t$; $k \longleftarrow i+1$ $p_k \longleftarrow UP$; {if new common head is to be produced}
3.1.1.2.1.2. $k \longleftarrow s$; $p_k \longleftarrow DOWN$; {if only the tail is to be modified};
else
3.2.1. $k \longleftarrow k+1$;
3.2.1. $p_k \longleftarrow DOWN$;
4. **return** $h' =< q_1, q_2, ..., q_i, p_{i+1}, ..., p_m >$;

The given structure of the algorithm *NEXTWQ* and any its variant demonstrates the goal, we have for designing the most successful procedure *SHORTCUT* taking under account valid properties of a given model. That goal is to produce relatively smallest value z. We justify it observing that for the general algorithm *NEXTWQ* the procedure *SHORTCUT* is trivial since it returns always value $z = k - 1$. Moreover, we observe:

" If relatively smaller value z can be returned, then in fact greater reduction of backtracking and looking forward steps at the whole run of the algorithm. Consequently, complexity of the algorithm *NEXTWQ* would be most reduced."

We will give now the algorithm *NEXTWQB* matching the model $S_U =< < S_i >, 1 \leq i \leq m; W$ given in (1.19); $\{h\}_{S_i} >$, the Q property does not hold, see (1.21). We get such model by implementing genetic algorithms in order to find a permutation solving a given intractable task, see Example 5.2.5.

Algorithm NEXTWQB. (*NEXTWQ* for choice Bijections)
Input: a choice bijection $h \in \{h\}_{\mathcal{G}_i}$
Output: the choice function h' that is the next greater than h (with respect to the lexical order)
Method: We postpone the technical details concerning assignment of the value for bijections. That arrangement was discussed earlier for the algorithm *NEXT* and its versions. With CD we denote codomain of the current partial mapping. At the beginning CD equals codomain of the given h.
Procedure SHORTCUT

can be applied only if EXISTENCE(q_1, q_2, ..., $q_i, p_{i+1}, ..., p_k$) =FALSE, it takes the string $< q_1, q_2, ..., q_i, p_{i+1}, ..., p_k >$ and the model S_U as input and it returns the value z as output.

We have two cases:

(i) $z = s \leq i$ in the case of necessity of modification of the head,

(ii) $z = t \leq k$ and $z > i$ is returned if the tail needs modification.

1. $z \longleftarrow k - 1$;
2. **while** $(z > i + 1$ and $(p_z \notin \mathcal{S}_k$ or $(\mathcal{S}_z - p_z) \cap \overline{CD} = \emptyset)))$ or
   $(z \leq i$ and $(q_z \notin \mathcal{S}_k$ or $q_z = \max\{\mathcal{S}_z - \overline{CD}\}))$ **do**

2.1. **case** z **of**

$z > i+1$: $CD \longleftarrow CD - \{p_z\}$;

$z \leq i$: $CD \longleftarrow CD - \{q_z\}$;

2.2. $z \longleftarrow z - 1$;

3. **return** z

---

1. $i \longleftarrow m - 1$; $k \longleftarrow m$;
2. $p_k \longleftarrow UP$;
3. **while** $i < m$ or $k < m$ **do**

3.1. **if** EXISTENCE(q_1, q_2, ..., $q_i, p_{i+1}, ..., p_k$) =FALSE **then**

3.1.1.1. $SHORTCUT$;

3.1.1.2. *push* (k, p_z) *on stock*; {the value p_z is going to be assigned into current k }

3.1.1.3. $CD \longleftarrow CD \cup \{p_z\}$;

3.1.1.4. **case** z **of**

$z \leq i : i \longleftarrow z - 1;\ k \longleftarrow i + 1;\ p_k \longleftarrow UP;$

$z > i : k \longleftarrow z;\ p_k \longleftarrow DOWN;$

else {if}

3.2.1. $k \longleftarrow k + 1$;

3.2.2. **if** k is on the top of stock **then** *pop*(k, p_z) *from stock*; $p_k \longleftarrow p_z$; **else** $p_k \longleftarrow DOWN$;

3.2.3. $CD \longleftarrow CD \cup \{p_k\}$;

4. **return** $h' =< q_1, q_2, ..., q_i, p_{i+1}, ..., p_m >$;

Theorem 8.2.1 *The algorithm NEXTWQB is correct.*

Proof. If the Q property does not hold for a given generation model and if $< q_1, q_2, ..., q_i, p_{i+1}, ..., p_k >$ is not a partial choice mapping the model, then there is a value $z \leq k$ that $< q_1, q_2, ..., q_z >$ is not a common head for h and h' or $p_{i+1}, ..., p_z$ is not the prefix of the tail for h'. Since we have to produce a bijection and $\mathcal{S}_i \neq \emptyset$ for every i, $1 \leq i \leq m$, so no value p_k could be assigned to the index k only if for every $p_k \in \mathcal{S}_k$ there is an index $j < k$ that $p_k = p_j$ or $p_k = q_j$. Basing on the expression at **while** (the step 2. of the procedure $SHORTCUT$) we observe, z is the

maximum index that current value $p_z(q_z)$ could be assigned into current k. The steps 3.1.1.2. and 3.1.1.3. and 3.1.1.4. reserve current value p_z for making its assignment into current k and replace p_z with a new allowable value. Consequently, we cut off a number of backtracking and looking forward steps that would enable us to update the value p_z, if the algorithm $NEXTWQ$ were used. Assignment of the values p_k is the same as for the algorithm $NEXTWQ$, so $NEXTWQ$ gives the same extendable partial choice mapping $< q_1, q_2, ..., q_i, p_{i+1}, ..., p_k >$ after a number of backtracking and looking forward steps, as we get by applying the $SHORTCUT$ procedure and by modification of the values between $p_z(q_z)$ and p_k performed in the steps 3.2.1., 3.2.2. and 3.2.3. For observing correctness of the stock usage, we note that on the top there is always a pair (k, p_z) that the index k is minimum out of all the indexes currently put on the stock. That means nesting the backtracking and looking forward steps for proper assignment of the values into consecutive indexes is correct. That finishes the proof of correctness of the algorithm $NEXTWQB$.

□

8.3 Surrounded generation of choice functions

We develop the generation of a set of choice functions that is alternative for exhaustive generation using UNSEQ1.

Let $< \mathcal{G}_i^{\min} >$, $1 \leq i \leq m$, be the minimal non-deformed indexed family for the given model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; W_1; \{h\}_{\mathcal{G}_i} >$. Then, we have the corresponding model $S_U^{\min} =<< \mathcal{G}_i^{\min} >, 1 \leq i \leq m; W; W_1; \{h\}_{\mathcal{G}_i^{\min}} >$. Suppose, h_1 and h_2 belong to $\{h\}_{\mathcal{G}_i}$. Then, h_1 and h_2 belong to $\{h\}_{\mathcal{G}_i^{\min}}$ since $\{h\}_{\mathcal{G}_i} \subseteq \{h\}_{\mathcal{G}_i^{\min}}$. The choice functions are ranked according to the lexical order $<^h$.

Let $R^{\min}(h)$ denote the lexical rank of a choice function h that is evaluated for the model $S_U^{\min}$.

Definition 8.3.1 *For the given choice functions* $h_1 =< h_1(1), h_1(2), ..., h_1(m) >$ *and* $h_2 =< h_2(1), h_2(2), ..., h_2(m) >$ *the numbers* $R^{\min}(h_1)$ *and* $R^{\min}(h_2)$ *we call the absolute rank.*

Correspondingly, $R^{\min}(h_2) - R^{\min}(h_1)$ we call the absolute rank difference.Observe, the rank difference is an integer accompanied by a sign (+ or −).

Suppose, we have created the model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W, W_1; \{h\}_{\mathcal{G}_i} >$ using as a prerequisite the set of choice functions $\{h'\}$ that $\{h'\} \subset \{h\}_{\mathcal{G}_i}$, see Example 5.2.5. The Q property does not hold for S_U. We have, $\{h'\} = \{h'_1, h'_2, ..., h'_z\}$, so that $R^{\min}(h'_1) < R^{\min}(h'_2) < ... < R^{\min}(h'_z)$. If the set $\{h'\}$ is to be generated exhaustively using a counterpart

of the algorithm UNSEQ1, then a counterpart of the algorithm $FIRST$ must be used that produces a choice function from the set $\{h\}_{\mathcal{G}_i}$ which rank is minimum. Since the Q property does not hold, so running the modified $FIRST$ can require much time and after making all these evaluations we are not sure that the result would be produced. Moreover, using the algorithm $NEXTWQ$ the running time for producing each next choice function $h \in \{h\}_{\mathcal{G}_i}$ can very significantly since for this algorithm we have $O(2^m)$ and $\Omega(1)$. On the other hand the absolute ranks of the choice functions $\{h\}_{\mathcal{G}_i}$ are focused around the absolute ranks of the choice functions $\{h'_1, h'_2, ..., h'_z\}$. If $R^{\min}(h'_1) << R^{\min}(h'_2) << ... << R^{\min}(h'_z)$, then running time of the algorithm $NEXTWQ$ is greatest for producing the first choice function focused around each h'_j.

In order to reduce the mentioned problems, we investigate the following surrounded generation for producing the whole set $\{h\}_{\mathcal{G}_i}$. On the other hand exhaustive generation of the set $\{h\}_{\mathcal{G}_i}$ may not be needed, then the surrounded generation can be completed that produces a subset $\{h\}^* \subset \{h\}_{\mathcal{G}_i}$, where $|\{h\}^*| \cong |\{h\}_{\mathcal{G}_i}|$. Generation of the set $\{h\}^*$ can be quite satisfactory for a given application. That approach could also be treated as alternative for more restrictive defining the crossing operations, if the genetic algorithms are concerned.

The surrounded generation of the choice functions can be specified as follows:

Method suround_gen. (surrounded generation - generates most of choice functions from the set $\{h\}_{\mathcal{G}_i}$).

Input: S_U and the set of choice functions $\{h'\} \subset \{h\}_{\mathcal{G}_i}$ used for producing the model S_U.

Output: The set of choice functions $\{h\}^* \subset \{h\}_{\mathcal{G}_i}$, where $|\{h\}^* \cong |\{h\}_{\mathcal{G}_i}|$.

Method: With denotation $NEX(h', <^h)$ or $NEX(h', >^h)$, we mean any algorithm from the group $NEXT$ or $NEXTWQ$ applicable if the Q property does not hold. The generation is followed using the lexical or anti-lexical order, accordingly.

A variable *count* is a measurement concerning the generated surrounding subsets, i.e., *count* can represent the number of the generated choice functions or it represents time consumed for their generation or any other measurement that concerns each surrounding and produced set of choice functions.

1. order the choice functions $\{h'\}$ according to the lexical order producing the string $< h' > = < h'_1, h'_2, ..., h'_z >$.

2. **for** $i = 1$ **to** z **do**

2.1. $h^* \longleftarrow h'_i$;

2.2. $count \longleftarrow 1$;

2.3. **while** $h' \neq h'_{i+1}$ and $count <$ max **do**

2.3.1. $h^* \longleftarrow NEX(h^*, <^h)$;{generate the next choice function according to the lexical order}
2.3.2. update *count*;
2.3.3. **return** h^*;
2.4. $count \longleftarrow 1$; $h^* \longleftarrow h'_i$;
2.5. **while** $h^* \neq h'_{i-1}$ and $count <$ max **do**
2.5.1. $h^* \longleftarrow NEX(h^*, >^h)$;{generate the next choice function according to the anti-lexical order}
2.5.2. update *count*;
2.5.3. **return** h^*

The given method can be implemented both as a sequential algorithm or we can make implementation as a parallel or distributed algorithm. We can select a measurement *count*, so that too much costly choice functions would not be generated. That concerns especially the first and the last choice functions according to the lexical order.

8.4 Random generation when the Q property holds

Another case of non-exhaustive generation of choice functions is their random generation. The tasks of random generation of choice functions can be split in two classes. The first class contains reduced models that the Q property holds, the second class of tasks concerns the models that the Q property does not hold. In this section, we are concerned with the first class of tasks. Similarly, as it was for the deterministic search and generation, we observe structural similarities between the algorithms developed for searching a random object in CF DATABASES and the generation of a random choice function for a fixed S_U model. We have given algorithms IDRAN and IDSEMRAN in [29] for random search of a record (object) contained in a fixed CF DATABASE. The concept and the structure of these algorithms we are going to carry on the ground of random choice function generation.

8.4.1 The algorithms GERAND and GERANDSTE

The goal is to generate a random choice function of a given model S_U or S_R. The simplest way of solving this task is to use a given S_R model. Then, we generate a random number $R(h)$ from the set $1 \leq | \{h\}_{\mathcal{G}_i} |$, where $R(h)$ is treated as the rank. The last step is to use the algorithm $UNRANK$ producing the choice function h. Complexity of this sketched method is rather high since we known that the algorithm $UNRANK$ has high asymptotic and time complexity by itself. Therefore, we propose another method using the model S_U. Then, the points $h(i)$, $1 \leq i \leq m$, of the function h are

generated in a random way following the model properties. Formally, we have the following general algorithm $GERAND$ that is a counterpart of the algorithm IDRAN [29].

Algorithm $GERAND$ (general algorithm for GEneration of a RANDom choice function)

Input: A given reduced model S_U that the Q property holds.

Output: The table IA represents the custom form of a random choice function $h \in \{h\}_{\mathcal{G}_i}$.

Method: The steps 1.1. and 1.2.1. denote random generation of an element that belongs to $\mathcal{G}_i$, while EXISTENCE($IA(1), IA(2), ..., IA(i-1), q_i$) takes value TRUE if there exists partial choice mapping $< IA(1), IA(2), ..., IA(i-1), IA(i) >$ that the requirements W and W_1 are satisfied.

1. **for** $i \longleftarrow 1$ **to** m **do**
1.1. $q_i \longleftarrow$random($\mathcal{G}_i$) ;
1.2. **while** EXISTENCE($IA(1), IA(2), ..., IA(i-1), q_i$) = FALSE **do**
1.2.1. $q_i \longleftarrow$random($\mathcal{G}_i$);
1.3. $IA(i) \longleftarrow q_i$;
2. **return** IA.

The given algorithm $GERAND$ posses time complexity $\Omega(m)$, while for the worst case we could have $O(\infty)$ since the loop 1.2. could be performed any number of times. Moreover, the given algorithm does not enable us implementing any statistic for controlling the process of random generation of choice functions. In order to avoid these difficulties, we propose the following algorithm.

Algorithm $GERANDSTE$ (general algorithm for GEneration of a RANDom choice function Enabling STatistics)

Input: A given reduced model S_U that the Q property holds.

Output: The table IA represents the custom form of a random choice function $h \in \{h\}_{\mathcal{G}_i}$.

Method: For $i = 1$ the set $\mathcal{G}_1^*$ equals $\mathcal{G}_1$. For $i > 1$ the algorithm $SET\mathcal{G}_i^*$ produces a proper set $\mathcal{G}_i^*$ as it was specified earlier. The step 1.2. uses simplified notation that means a random element form the set $\mathcal{G}_i^*$ is assigned as the value $IA(i)$.

1. **for** $i \longleftarrow 1$ **to** m **do**
1.1. $SET\mathcal{G}_i^*$;
1.2. $IA(i) \longleftarrow$ random($\mathcal{G}_i^*$);
2. **return** IA.

Asymptotic complexity of the algorithm $GERANDSTE$ is $O(m.n)$ because of necessity of production of the set $\mathcal{G}_i^*$, while we have also $\Omega(m)$ for the best case. Since, we can implement statistics for performing the step 1.2., so the given algorithm enables us controlling probability of the gen-

eration of particular elements from the set $\mathcal{G}_i^*$, that determines a statistic ruling the generation of choice functions.

8.4.2 The algorithms GRERAN and RANSCAN

Given is a model $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; W_1; \{h\}_{\mathcal{G}_i} >$, a choice function $h \in \{h\}_{\mathcal{G}_i}$ and the lexical order $<^h$. The task is to generate a choice function $h' \in \{h\}_{\mathcal{G}_i}$ that the corresponding ranks satisfy the relation $R(h') > R(h)$. Similarly, as it was given in the previous section, we could solve this task by generating a random number p from the range $(R(h)+1, |\{h\}|]$. Then, treating p as the rank of a choice function h' and using the algorithm $UNRANK$, we could produce h'. We have for this method asymptotic complexity $O(n.m)$ and $\Omega(m)$, since it is effected by complexity of the algorithm $UNRANK$. Time complexity of such method is really huge and it makes the sketched approach impractical. Moreover, for making such generation we have to have assertion that the set is symmetric. We get better results using the following algorithm $GRERAN$.

Algorithm GRERAN (general algorithm for the generation of the choice functions whose rank is RANdomly GREater than the given one).

Input: A given reduced model S_U that the Q property holds, the table IA representing the given choice function h.

Output: The table IA represents the custom form of a choice function $h' \in \{h\}_{\mathcal{G}_i}$, that $R(h') > R(h)$.

Method: For $i = 1$ the set $\overline{\mathcal{G}}_1^* - \{IA[1]\}$ equals $\mathcal{G}_1 - \{IA[1]\}$. For $i > 1$ the algorithm $SET\overline{\mathcal{G}}_i^*$ produces a proper set $\overline{\mathcal{G}}_i^*$ using the value $IA(i-1)$ as it was specified earlier. The goal of the steps 3. and 4. is to find the value j that is nearest to i and such that increasing $IA(i)$ could be done. A simplified notation, we use at the step 6. similarly as it was given at the step 1.2. of the algorithm $GERANDSTE$. Observe, this notation differs from the notation used at the step 1.

1. $i \longleftarrow$ random$(1, m)$;
2. $j \longleftarrow i$;
3.1. **while** $(IA(j) = \max(\mathcal{G}_j))$ and $(j < m)$ **do**
3.1.1. $j \longleftarrow j + 1$;
4. **if** $((j = m)$ and $(j = \max(\mathcal{G}_j))$ **then**
4.1. $j \longleftarrow i - 1$;
4.2. **while** $(IA(j) = \max(\mathcal{G}_j))$ and $(j > 1)$ **do**
4.2.1. $j \longleftarrow j - 1$;
4.2.2. **if** $((j = 1)$ and $(j = \max(\mathcal{G}_j)))$ **then return** "rank $R(h)$ is maximum"; **stop;**
5. $SET\overline{\mathcal{G}}_j^*$;
6. $IA[j] \longleftarrow$ random $(\overline{\mathcal{G}}_j^* - \{IA[j]\})$;
7. **for** $i \longleftarrow j + 1$ **to** m **do**

7.1. $SET\mathcal{G}_j^*$;
7.2. $IA(i) \longleftarrow \min(\mathcal{G}_i^*)$;
8. **return** IA;

We have complexity $O(n.m)$ and $\Omega(1)$ for the algorithm *GRERAN*. If smaller value i is produced at the step 1., then there is greater rank difference $R(h') - R(h)$. So, the main control of the rank difference $R(h') - R(h)$ can be done using different statistics for random generation of value i. The range of control of the difference $R(h') - R(h)$ is much more narrow, if we use a statistic for producing greater or smaller values $IA[j]$ at the step 6. We have also a possibility of controlling the tail produced at the step 7. For that purpose, we replace the step 7.2. with the following step 7.2. $IA(i) \longleftarrow random(\mathcal{G}_i^*)$ and we can use statistics for making this random generation.

The algorithm *GRERAN* can be used as the basic component for scanning in a random manner a set of choice functions $\{h\}$ of a given model S_U. For this scanning we can use the following algorithm *RANSCAN*.

Algorithm RANSCAN (RANdom SCANner)
Input: model S_U.
Output: randomly generated choice functions whose ranks increase
1. $h \longleftarrow FIRST$
2. **while** $1 = 1$ **do**
2.1. $h' \longleftarrow GRERAN$;
2.2. $h \longleftarrow h'$.

The algorithm *GRERAN* can be widely used if non-exhaustive generation of choice functions for a model that posses the Q property.

8.5 Random generation when Q property does not hold

For the models that the Q property does not hold, we can not apply the algorithms for random generation of choice functions developed in the previous section. There is strong similarity between the algorithms for deterministic and random generation of choice functions using models that the Q property does not hold, instead. For the class of models discussed in this section a real random generation of choice functions using a counterpart of the algorithm *GRERANSTE* is theoretically possible. Nevertheless, very high asymptotic and time complexities of such algorithms, especially if the Q property does not hold for numerous instances of a given model, make such investigations impractical. Much easier task is to generate a semi random choice function h' that its rank is greater from a given choice function

h. For making such generation, we will develop a counterpart of the general algorithm $NEXTWQ$.

We present now the structure of the general $GENRANWQ$ algorithm for random generation of a choice function whose rank is greater than the rank of a given choice function h.

The structure of the algorithms $GRERANWQ$.

Input: a choice function $h \in \{h\}_{\mathcal{G}_i}$

Output: a random choice function h' whose rank is greater than h (with respect to the lexical order)

Method: We have two variants of such general algorithm. The instructions specified in parentheses are assigned to the second variant that enables us to implement numerous statistics concerning the generation of the choice function h'. For the first variant, we randomly select only the initial length i of the head. Then, the choice function h' possessing the minimum rank and the head of the length i or smaller is produced.

Functions UP and $DOWN$ are the same as those given for the algorithm $NEXTWQ$.

Procedure SHORTCUT

{similar as it was given for the algorithm $GRERUN$ used for the model S_U}.

- -

1. $i \longleftarrow \text{random}(1, m)$; $k \longleftarrow i + 1$;
2. $p_k \longleftarrow UP$; $(p_k \longleftarrow \text{random}(\overline{\mathcal{G}}_i^* - \{IA[i]\}))$;
3. **while** $i < m$ or $k < m$ **do**
3.1. **if** EXISTENCE$(q_1, q_2, ..., q_i, p_{i+1}, ..., p_k)$ =FALSE **then**
3.1.1.1. $SHORTCUT$;
3.1.1.2. **case** z **of**
3.1.1.2.1.1. $z < t : i \longleftarrow t$; $k \longleftarrow i + 1$; $p_k \longleftarrow UP$; $(p_k \longleftarrow \text{random}(\overline{\mathcal{G}}_i^* - \{IA[i]\}))$;
3.1.1.2.1.2. $k \longleftarrow s$; $p_k \longleftarrow DOWN$; $(p_k \longleftarrow \text{random}(\mathcal{G}_i^*))$;
else
3.2.1. $k \longleftarrow k + 1$;
3.2.1. $p_k \longleftarrow DOWN$; $(p_k \longleftarrow \text{random}(\mathcal{G}_i^*))$;
4. **return** $h' =< q_1, q_2, ..., q_i, p_{i+1}, ..., p_m >$;

Simple replacement of the instruction $p_k \longleftarrow UP$ with the instruction $p_k \longleftarrow \text{random}(\overline{\mathcal{G}}_i^* - \{IA[i]\})$ and replacement of the instruction $p_k \longleftarrow DOWN$ with the instruction $p_k \longleftarrow \text{random}(\mathcal{G}_i^*)$ essentially changes time complexity of evaluations especially if for numerous partial choice mapping, we can not get any full choice function. If we use the instructions $p_k \longleftarrow UP$ and $p_k \longleftarrow DOWN$, then the procedure $SHORTCUT$ can be used similarly as for the deterministic generation. If the instructions for random generation of p_k are used, then development of a proper $SHORTCUT$

procedure is much more difficult since there is a greater number of the possible tails for completing the generation. Therefore, we recommend possibility of usage of the instructions $p_k \longleftarrow \text{random}(\overline{\mathcal{G}}_i^* - \{IA[i]\})$ and $p_k \longleftarrow \text{random}(\mathcal{G}_i^*)$ only if the Q property does not hold for a few partial choice mappings, while usage of the instructions $p_k \longleftarrow UP$ and $p_k \longleftarrow DOWN$ should be considered as a valid principle if for numerous partial choice mappings the Q property does not hold. Nevertheless, we emphasis that applying instructions $p_k \longleftarrow \text{random}(\overline{\mathcal{G}}_i^* - \{IA[i]\})$ and $p_k \longleftarrow \text{random}(\mathcal{G}_i^*)$ enables us to use numerous statistics for the generation of a random choice function h' that $R(h') > R(h)$.

If we develop a proper variant of the algorithm $GRERANWQ$, then the algorithm $RANSCAN$ can be used directly.

8.6 Non-monotonic generation of choice functions

The goal is to develop deterministic ranking model that entropy of the absolute rank difference $R^{\min}(h) - R^{\min}(h')$ is as high as possible for given h, where h' is the next choice function generated for h, see (8.3.1). So, if one knows h, $R(h)$, $R(h')$ but does not know the model, then he is not able to evaluate h'. In other words, we can say that the goal is to make the order $\prec^h$ used as much hidden as possible. Of course, the requested order $\prec^h$ must be pretty far from the lexical order and any its variant including piecemeal lexical orders. The second restriction we put on $\prec^h$ is specification to be given for the proper user of the model. On one side the specification can not be defined with a simple formula since the requested entropy of h' could not be high. On the other side, we can not make a tabular specification of the order $\prec^h$, since the ranking model would be too much bulky. We conclude that solving the stated problem when staying on the classical ground would be really a difficult task.

If the developed till now representation models were used, then entropy of h' would be much greater since the number of the possible representation models depends exponentially on n and m. Nevertheless, if a sequence of consecutive choice functions were known, then evaluation of the next h' would be an easy task. Therefore, in order to preserve high entropy of h' for its given rank, a new approach for representing sets of choice functions is needed. A method for increasing the entropy of h' was investigated in [30] and its specific usage for development of cryptographic systems was given in [31]. We will recall and develop this methodology.

The block diagram of a hierarchical ranking model that possess seemingly random generation order $\prec^h$ is given in Fig.8.1.

The hierarchical model possess k levels. The order $\prec^h$ used for the model is nowhere specified but could be revealed by the sequence of choice functions h generated in accordance with the sequence of natural numbers

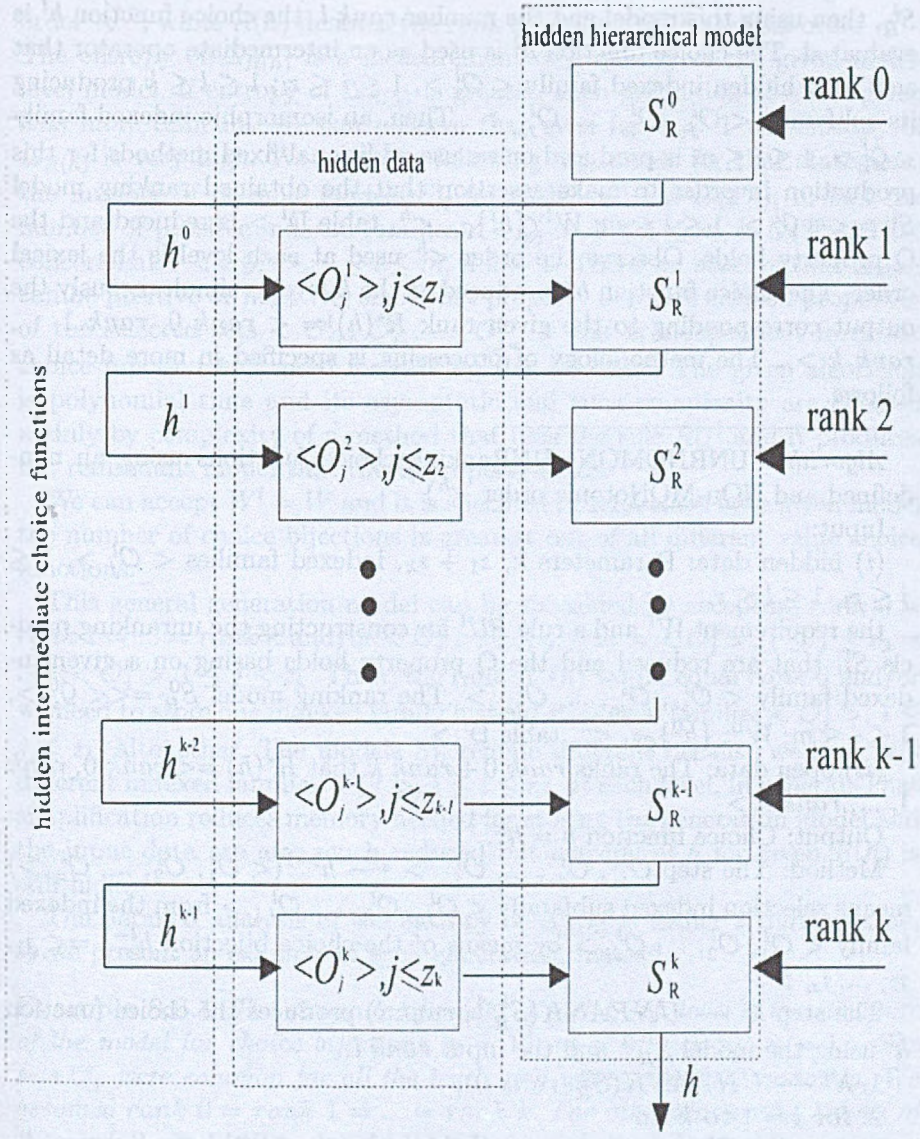

FIGURE 8.1. A ranking model for non-monotonic generation of choice functions
RYSUNEK 8.1. Model rankingowy dla niemonotonicznej generacji funkcji wyboru

$1, 2, ...$ representing the rank $R^*(h)$. In fact, the string of the ranking models $< S_R^0, S_R^1, ..., S_R^k >$ is made individually in the course of processing for each natural number representing a given rank $R^*(h)$. At each step of the hierarchical model evaluation, we construct the consecutive component S_R^l, then using this model and the number *rank* l the choice function h^l is evaluated. The choice function h^l is used as an intermediate operator that acts on a hidden indexed family $< \mathcal{O}_i^l >$, $1 \leq i \leq z_l$; $1 \leq l \leq k$ producing its subfamily $< \mathcal{O}_{j_1}^l, \mathcal{O}_{j_2}^l, ..., \mathcal{O}_{j_m}^l >$. Then, an isomorphic indexed family $< \mathcal{G}_i^l >$, $1 \leq i \leq m$ is produced or we use additional fixed methods for this production in order to make assertion that the obtained ranking model $S_R^l =<< \mathcal{G}_i^l >$, $1 \leq i \leq m$; W^l; $\{h^l\}_{\mathcal{G}_i}$, $<^h$, table $\mathrm{D}^l >$ is reduced and the Q property holds. Observe the order $<^h$ used at each level is the lexical order. The choice function h^k produced at the k level is simultaneously the output corresponding to the given rank $R^*(h) = <$ *rank* 0, *rank* 1, ..., *rank* $k >$. The methodology of processing is specified in more detail as follows.

Algorithm UNRNOMON(UNRanking choice functions using an non-defined and NOn-MONotonic order $\prec^h$)

Input:

(i) hidden data: Parameters k, $z_1 \div z_k$, indexed families $< \mathcal{O}_j^l >$, $1 \leq j \leq z_l$, $1 \leq l \leq z_l$,

the requirement W^l and a rule RU^l for constructing the unranking models S_U^l that are reduced and the Q property holds basing on a given indexed family $< \mathcal{O}_{j_1}^l, \mathcal{O}_{j_2}^l, ..., \mathcal{O}_{j_m}^l >$ .The ranking model $S_R^0 =<< \mathcal{G}_i^0 >$, $1 \leq i \leq m$; W^0; $\{h^0\}_{\mathcal{G}_i^0}$, $<^h$, table $\mathrm{D}^l >$.

(ii) open data: The ranks *rank* $0 \div$ *rank* k that $R^*(h) =<$ *rank* 0, *rank* 1, ..., *rank* $k >$.

Output: Choice function $h = h^k$.

Method: The step $\mathcal{O}_{j_1}^l, \mathcal{O}_{j_2}^l, ..., \mathcal{O}_{j_m}^l > \longleftarrow h^{l-1}(< \mathcal{O}_1^l, \mathcal{O}_2^l, ..., \mathcal{O}_{z_l}^l >)$ means selection indexed subfamily $< \mathcal{O}_{j_1}^l, \mathcal{O}_{j_2}^l, ..., \mathcal{O}_{j_m}^l >$ from the indexed family $< \mathcal{O}_1^l, \mathcal{O}_2^l, ..., \mathcal{O}_{z_l}^l >$ by means of the choice bijection $h^{l-1} =< j_1, j_2, ..., j_{z_l} >$.

The step $h^l \longleftarrow UNRANK(S_R^{l-1}, \textit{rank } l)$ produces the choice function h^l using the model S_R^{l-1} and the input *rank* l.

1. $h^0 \longleftarrow UNRANK(S_R^0, \textit{rank } 0)$;
2. **for** $l = 1$ **to** k **do**
2.1. $< \mathcal{O}_{j_1}^l, \mathcal{O}_{j_2}^l, ..., \mathcal{O}_{j_m}^l > \longleftarrow h^{l-1}(< \mathcal{O}_1^l, \mathcal{O}_2^l, ..., \mathcal{O}_{z_l}^l >)$;
2.2. make the indexed family $< \mathcal{G}_i^l >$, $1 \leq i \leq m$ applying RU^l into $< \mathcal{O}_{j_1}^l, \mathcal{O}_{j_2}^l, ..., \mathcal{O}_{j_m}^l >$;
2.3. evaluate table D^l,
2.4. $h^l \longleftarrow UNRANK(S_R^{l-1}, \textit{rank } l)$;
3. $h \longleftarrow h^k$;
4. **return** h;

The block diagram shows iterative evaluation of the choice function h for a given rank $R^*(h)$. One can show that if $R^*(h') = R^*(h) + 1$, then $R(h') = R(h) + \Delta_{R(h)}$, where $R^*(h)$ denotes the rank according to the order $\prec^{h^*}$, while $R(h)$ denotes the rank according to the lexical order $<^h$. The entropy of $\Delta_{R(h)}$ is a measurement of a quality of the given multi-level model. If entropy of $\Delta_{R(h)}$ is greater, then prediction of h' for given h is more difficult. We can observe that even for $k = 1$ the entropy of $\Delta_{R(h)}$ is very high. We have the following arguments for that statement: the number of possible models S_R^l for each level l is $O(2^m)$, in fact the number of possible indexed families $<< \mathcal{G}_i^l >>$, $1 \leq i \leq m$, is $O(2^m)$. That concerns also $<< \mathcal{G}_i^1 >>$, $1 \leq i \leq m$, if $k = 1$. Then, we observe that $\Delta_{R(h)}$ can be positive or negative, since that property depends only on properties of the selected sets $< \mathcal{O}_{j_1}^l, \mathcal{O}_{j_2}^l, ..., \mathcal{O}_{j_m}^l >$ but is independent from the choice function h^{l-1} used for making this selection. The given algorithm is polynomial time and its asymptotic and time complexity are effected mainly by complexity of a method that uses the rule RU^l and it produces not redundant model that the Q property holds.

We can accept $W^l = W$ and it is specified in 1.19, since for a given model the number of choice bijections is greatest out of all different value choice functions.

This general generation model can be simplified by accepting $rank\ 0 = rank\ 1 = ... = rank\ k$ and/or $< \mathcal{O}_j^1 >$, $1 \leq j \leq z_1 = < \mathcal{O}_j^2 >$, $1 \leq j \leq z_2 = ... = < \mathcal{O}_j^k >$, $1 \leq j \leq z_k$. Then, the rank $R^*(h)$ would equal $rank\ 0$ and/or we need to store one indexed family instead k indexed families $< \mathcal{O}_j^1 >$, $1 \leq j \leq z_1$, altogether. The models S_R^l remain different because we have still different indexed families $< \mathcal{G}_i^l >$, $1 \leq i \leq m$, at each level, in general. That simplification reduces memory needed for storing the generation model and the input data are also much reduced but entropy of h for given $R(h)$ is still high.

The detailed analysis of the entropy of $\Delta_{R(h)}$ is rather a difficult task, so we present an example of such generation, instead.

Example 8.6.1 *The structure of model at each $l-th$ level is the structure of the model for choice bijections, $n = 12$, $m = 8$, $k = 10$, $z_l = 12$. The sets $\mathcal{O}_j$ were common for all the levels and were generated randomly. We assumed $rank\ 0 = rank\ 1 = ... = rank\ k$. For nine consecutive values of the rank $0 =< 1 \div 9 >$, we get the following choice bijections h.*

$1- < 1,4,3,7,12,10,2,5 >$
$2- < 3,1,4,9,8,5,7,12 >$
$3- < 1,3,8,6,9,12,7,2 >$
$4- < 3,1,6,5,4,9,10,12 >$
$5- < 1,2,10,4,6,8,3,7 >$
$6- < 3,7,12,2,8,10,4,11 >$

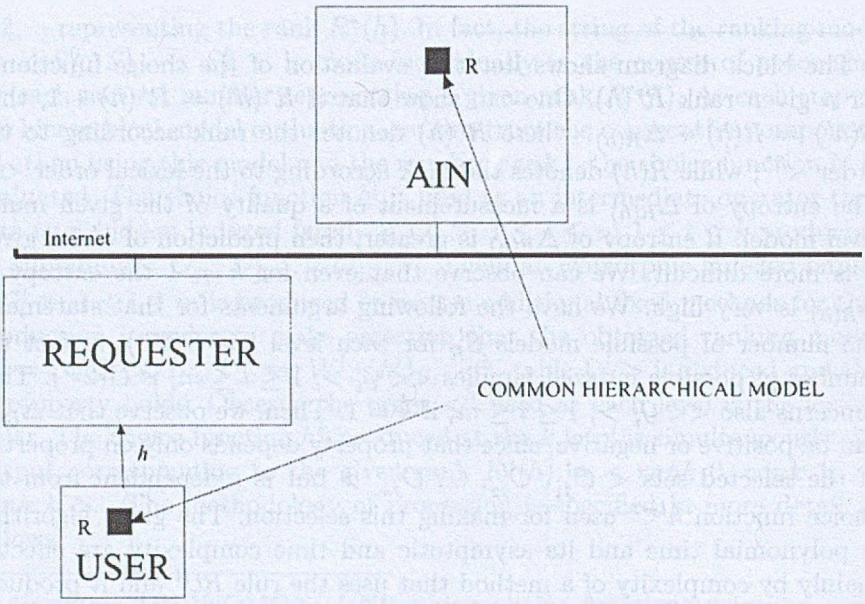

FIGURE 8.2. The block diagram of autentification
RYSUNEK 8.2. Schemat blokowy autentyfikacji

$7- < 1,4,11,8,6,10,7,2 >$
$8- < 7,1,4,9,2,11,5,8 >$
$9- < 4,2,10,8,9,5,6,12 >$

□

Consecutive choice functions produced can not be seen as random ones for an external observer but rather as semi random, since the numbers 1, 4 and 3 repeat very often as the values taken for the first three indexes. That property is very common for the considered here generation and it happens independently from values n, m and k, the starting rank $R^*(h)$ does not also have particular meaning for existence of this property. In order to reduce influence of this property on possibility of predicting the choice function h', we have to use possibly big values n and m but the value k is not especially significant for greater semi-randomization of the results.

The given multi-level models for non-monotonic ranking generation of the choice functions possess very wide and significant possibilities of application in cryptography. Simply the rank send by an open network can be treated as a secret cod of a key value represented by the corresponding choice function. Another valid applications concern individual digital & changeable signatures that correspond to consecutive choice functions of a given model, [30]. Usage of the authorization system is demonstrated in

Fig.3. For getting confirmation of the signature we can use the following two protocols.

The first protocol.

AIN (Authorization INstitution) and USER possess the same hierarchical model, both posses the same current value of the rank $R^*(h)$. The changeable digital signature is the pair $(R^*(h), h)$.

1. The value $R^*(h)$ USER increases by 1 and such updated $R^*(h)$ he stores and uses for evaluation corresponding h.

2. Then USER submits the pair $(R^*(h), h)$ to REQUESTOR.

3. Next REQUESTOR sends the name of USER to AIN asking for confirmation.

4. AIN checks the current value $R^*(h)$ contained in the model of USER and evaluates corresponding h.

5. AIN sends the pair $(R^*(h), h)$ to REQUESTOR.

6. REQUESTOR gets confirmation if the pairs $(R^*(h), h)$ obtained from REQUESTOR and USER match.

7. AIN increases $R^*(h)$ by 1 and writes this new value to the model of USER.

This first protocol enables not only confirmation of the changeable signature but lack of confirmation informs USER that somebody used his model for authorization or something wrong is going on with AIN. The disadvantage of the first protocol is possibility of making not wanted confirmation by REQUESTOR.

The second protocol.

AIN (Authorization INstitution) and USER possess the same hierarchical model, both posses the same current value of the rank $R^*(h)$. The changeable digital signature is the pair $(R^*(h), h)$.

1. The value $R^*(h)$ USER increases by 1 and such updated $R^*(h)$ he stores and uses for evaluation corresponding h.

2. Then USER submits the pair $(R^*(h), h)$ to REQUESTOR.

3. Next REQUESTOR sends the pair $(R^*(h), h)$ and the name of USER to AIN asking for confirmation.

4. AIN checks whether $R^*(h)$ obtained from REQUESTOR and stored in the model of USER match if no then REQUESTOR and next USER are informed that $R^*(h)$ is wrong, so confirmation is not possible.

5. If values $R^*(h)$ obtained form REQUESTOR and stored in the model match, then AIN evaluates h using model of USER.

6. If h evaluated by AIN and that obtained from REQUESTOR match, then AIN gets confirmation and informs REQUESTOR about

7. AIN increases $R^*(h)$ by 1 and writes this new value to the model of USER.

The second protocol gives greater safety both for USER and AIN. First of all the REQUESTOR can use only once the pair $(R^*(h), h)$ requiring confirmation. Moreover, if confirmation can not be completed, then both AIN and USER can come to interesting for them conclusions.

This simple method of usage of the hierarchical non-monotonic ranking model possess numerous practical applications that are pretty far from the subjective of this text, so we do not discuss them here.

8.7 Conclusion

Numerous problems concerning non-exhaustive generation of combinatorial objects have been considered in this chapter. We have developed methodology for the surrounded generation of the choice functions, for their random generation and also for non- monotonic generation of choice functions. The developed methodology concerns both the unranking models that the Q property holds or we use models that the Q property does not hold. For non-monotonic generation of choice functions the ranking models are used. The presented results prove versatility of the developed theory. We can easily apply it for discussing numerous generation problems included in this text. Since the developed theory concerns the fundamental properties of the models, so we can also expect its usefulness for studying new problems that were not mentioned in this text. The structural similarity between search problems performed in CF DATABASES and DSMs and the generation of choice functions shown in this chapter create new interesting perspectives of seeing search and generation as the similar problems. It enables us to imagine replacement of huge databases with compact generation models and a search replace with a generation. That perspective creates a new interesting range of future research.

The general results developed in this text show clearly that the problems of combinatorics possess strongly algebraic character, so algebra should be seen as the main supporter of the methodology when combinatorial problems are concerned.

References

[1] Akl, S. G., The Design and Analysis of Parallel Algorithms, Prentice–Hall, 1988.

[2] Akl, S. G., and Stojmenovic, I., Parallel Algorithms for Generating Integer Partitions and Decompositions, J. CMCC Vol.13, (1993), 107–120.

[3] Akl S.G., Stojmenovic, I., Generating Combinatorial Objects on a Linear Array of Processors, TR–93–10 University of Ottawa, April, 1993.

[4] Bisschop, J.J., Fourer, R., New constructs for the description of combinatorial optimisation problems in algebraic modelling languages, Computational Optimization and Applications, Vol.6, no.1, (1996), 83–116

[5] Brinkmann, G., McKay, B.D., Fast generation of non-isomorphic and cubic graphs of girth nine, Comb. Prob. Comp. Vol. 4, (1995), 317 – 330.

[6] Djokic, B., Miyakawa, M., Sekiguchi, S., Semba, I., and Stejmonowic, I., Parallel Algorithms for Generating Subsets and Set partitions, Proc. SIGAL Int. Symp. on Algorithms, Tokyo, Japan, Aug. 1990, Lectures note in computer science, Vol.450, (T.Asano, T.Ibaraki, H.Imai, T.Nishizeki, eds.), Springer–Verlag, 1990, pp.76 – 85.

[7] Dickson, L.E., History of the Theory of Numbers, Vol.II, Diophantine Analysis, Chelsea Publishing Co., New York, 1971.

[8] Er, M. C., A Fast Algorithm for Generating Set partitions The Computer J. , Vol. 31, No 3, (1988), 283 – 284.

[9] Er, M.C., Classes of Admissible Permutations that are Generatable by Depth–first Traversals of Ordered Trees. The Computer J., Vol.32, No 1, (1989), 76 – 85.

[10] Faradzev, I.A. Generation of nonisomorphic graphs with a given degree sequence, Algorithmic studies in Combinatorics, Nauka, Moscow, (1978), 11 – 19 (*in* $R^*(h)ussian$).

[11] Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[12] Graham, R.L., Knuth D., Patashnik O., Concrete Mathematics, Addison–Wesley, 1994.

[13] Gupta, Ph., Bhattacharjee G.P. Parallel Generation of Permutations. The Computer J., Vol.26, No 2, (1983), 97 – 105.

[14] Hall Ph., On representative of subsets, J London Math. Soc. 10 (1935), 26 – 30.

[15] Hall, M. Jr., Combinatorial Theory, New York, Chicheser, Brisbane, Toronto, Singapore, Wileys, (1986).

[16] Hindman, N., and Lefmann, H. partition Regularity of (m, P, C)–Systems, Journal of Combinatorial Theory, Series A, Vol.64, 1–9, (1993).

[17] Hoffmann, Ch.M., Group–Theoretic Algorithms and Graph isomorphism . Springer–Verlag, Berlin, Heidelberg, New York, (1982).

[18] Jian–Yi Shi, Skew Tableaux, Lattice Paths, and Bounded partitions Journal of Combinatorial Theory, Series A Vol. 63, (1993), 79 – 89

[19] Kapralski, A., New Methods for the Generation of Permutations, Combinations and Other Combinatorial Objects in Parallel, J. Parallel Distrib. Comp. Vol.17, (April, 1993), 315 – 326.

[20] Kapralski, A., Representation and Parallel Generation of Number and Set Partitions, Decompositions, Compositions and Related Combinatorial Objects. The University of Aizu, Tech. Rep. 94 – 1 – 38, Aizu–wakamatsu, July, 1994.

[21] Kapralski, A., Fast Massively Parallel Algorithms for Shortest Path within Plannar Figures Visual-Computer Vol 12. Springer-Verlag (1996), 484 – 502.

[22] Kapralski, A., Sequential and Parallel Processing in Depth Search Machines. World Scientific, Singapore, 1994.

[23] Kapralski, A., Macierze binarne ich zastosowania i przetwarzanie szeregowe i równoległe w dedykowanych procesorach, Monografia 95, Politechnika Krakowska, Krakow, 1989. (in Polish).

[24] Kapralski, A., "Supercomputing for solving a class of NP−complete and isomorphic−complete problems" Comp. Sys. Sci. and Eng. 7, 4 (1992) 218 – 228.

[25] Kapralski, A., "The Maximum and Minimum Selector SELRAM and its Application for Developing Fast Sorting Machines", IEEE Trans. on Comp. 38, 11 (1989) 1572 – 1576.

[26] Kapralski, A. and Skarbek, W.,"Problem of Searching Minimum Base in Boolean Tables", Podstawy Sterowania 16, z. 3 – 4 (1986) 257 – 265.

[27] Kapralski, A., "The Boolean Space Theory and its Application for Designing Digital Systems", Politechnika Krakowska, Zeszyty Naukowe Transport No. 2 (Krakow, 1979). (in Polish)

[28] Kapralski A. " Maszyna do głębokiego wyszukiwania jako maszyna kodująca i dekodująca" , Zeszyty Naukowe Pol. Śl., seria INFORMATYKA z.36 No 1414 pp. 263-274, 1999.(in Polish)

[29] Kapralski A. "Szyfrowanie i deszyfrowanie w maszynach do głębokiego wyszukiwania" w książce „Bezpieczestwo systemów komputerowych i Telekomunikacyjnych" redaktor Andrzej Grzywak, Wydawnictwo SOTEL , pp. 274-293, 1999.(in Polish)

[30] Kapralski A. "Niemonotoniczne modele rankingowe" STUDIA INFORMATICA, Vol. 21 No1 pp. 601-612, 2000.(in Polish)

[31] Kapralski A. "Systemy kryptograficzne na platformie maszyn do głębokiego wyszukiwania" rozdzial w książce

[32] Kaye, R. A., Gray Code for Set partitions IPL, 5, (1976),171 – 173.

[33] Kelsen, P., Ranking and unranking trees using regular reductions, in Proceedings STACS 96. 13th Annual Symposium on Theoretical Aspects of Computer Science p. xii + 690, 581 – 592 (1996)

[34] Kokosiński, Z. Mask and pattern generation for associative supercomputing, in Proceedings of the Twelfth IASTED International Conference Applied Informatics, p. 353, (1996) 324 – 6

[35] Kokosiński, Z., Algorithms for Unranking combinations and Their Applications in Proceedings of the Seventh IASTED/ISMM International Conference on Parallel and Distributed Computing and Systems, 216 – 224, (1995)

[36] Kokosiński, Z., On Parallel Generation of combinations in Associative Processor Architectures. The University of Aizu, Tech. Rep. 94 – 1 – 003, Aizu–wakamatsu, June, 1996.

[37] Kokosiński, Z., Unranking combinations in Parallel, in Proceedings of the PDPTA'96, Sunnyvale, CL, Aug. 9 – 11, USA, 79 – 82, (1996)

[38] Kokosiński, Z., Generation of Integer compositions on Linear Array of Processors, in Proceedings of the International Conference PDPTA'96, Sunnyvale, CL, Aug. 9 – 11, USA, 56 – 64, (1996)

[39] Kokosiński, Z., Układy generatorów obiektów kombinatorycznych dla systemów sekwencyjnych i równoległych., Politechnika Krakowska, Monografia 160, Kraków 1993.

[40] Kokosiński, Z., On Parallel Generation of Set Partitions in Associative Processor Architectures, Int. Conf. on Artificial Intelligence June 28 - July 1, 1999, Las Vegas, Nevada, CSREA Press, Vol III, pp.1257–1262.

[41] Knuth, D.E., The Sandwich Theorem, Electronic J. Combinatorics 1, A1, (1994), 48pp.

[42] Fischer, L.L., Krause, L.C., Lehrbuch der combinationslehre und der Arithmetik. Dresden, 1812.

[43] Lehmer, D.H., The machine tools of combinatorics. In Beckenbach E.F. (Ed.). Applied combinatorial mathematics, John Willey, New York, (1964), pp.5 – 31.

[44] Lipski W., Marek W., "Analiza kombinatoryczna" Biblioteka Matematyczna tom 59, PWN, Warszawa 1985 (in Polish).

[45] McKay, B.D., Isomorph-free Exhaustive Generation. Journal of Algorithms, Vol. 26, (1998), 306 – 324.

[46] McKay, B.D., Radziszowski, S.P., Subgraph counting identities and Ramsey numbers, J. Combinatorial Theorey ser.B. Vol. 69, (1997), 193 – 209.

[47] McKay, J. K. S. Partitions in Natural Order, Algorithm 371, Com of ACM, Vol. 13, (1971), 52.

[48] Michalewicz, Z. Algorytmy genetyczne+struktury danych = programy ewolucyjne, WNT, W-wa, 1999, (translated from: Genetic Algorithms + Data Structures = Evolutionary Programs, Springer-Verlag, Berlin, Heidelberg 1992).

[49] Mirsky, L., Transversal Theory Academic Press, New York, London, 1969.

[50] Page, E.S., and Wilson, L.B., An Itroduction to Computational Combinatorics, Cambridge University Press, Cambridge, London, New York, Melbourne, 1979.

[51] Nijenhius, A., and Wilf, H.S., A method and two algorithms on the theory of partitions, Journal of Combinatorial Theory, Vol.18, (1975), 219 – 222.

[52] Nijenhius, A., and Wilf, H.S. Combinatorial Algorithms, ACADEMIC PRESS, New York, San Francisco, London, 1975.

[53] Sedgewick, R., Permutation Generation Methods. Computing Surveys, Vol.9, No 2, (June 1977), 137 – 164.

[54] Stearns, R.E., Hunt, H.B. An algebraic model for combinatorial problems, SIAM Journal on Computing, vol.25, no.2, (1996), 448 – 476

[55] Tang, C.Y., Du, M.W., Lee, R.C.T., Parallel Generation of combinations. Proc.of International Computer Symposium, Taipei, Taiwan, (1984), pp.1006 – 1010.

[56] Wells, M. Elements of Combinatorial Computing Pergamon Press, Oxford, New York, Toronto, Sedney, Braunschweig, 1971.

[57] Williamson, S.G. Ranking Algorithms for Lists of partitions SIAM J. on Computing, Vol. 5, No 4, (1976), 602 – 617.

Index

Modelling Arbitrary Sets of Combinatorial Objects and their Sequential and Parallel Generation

Summary

This text shows the representations of sets of combinatorial objects by choice functions of indexed families. We investigate and study the unranking models those represent sets of choice functions. Our main concerns are sets of increasing choice functions, sets of monotonic choice functions and sets of choice bijections. We demonstrated usage of these models for representing the basic combinatorial objects, i.e., permutations, combinations, variations, partitions, decompositions and compositions. Then, the structural numbers and the tables D for representing the structure of sets of choice functions are studied. The tables D create new theoretical foundation for deeper understanding of the structure of the whole combinatorics in relation to Stirling's numbers and to Pascal's triangle. These tables are the main components of the ranking models, we investigate also transformations of the models basing on the tables D. Ranking Theorem presents the main existential result for our philosophy of getting the most suitable models. As far as the representation of sets of combinatorial objects is concerned the choice function approach gives benefits that could not be obtained using the classical methodology, we demonstrate these benefits. Then, the generation of combinatorial objects is performed by means of the generation of choice functions. We develop the system $< Methods >$ of the basic procedures for the generation of choice functions. We investigate the general algorithms $NEXT$, $FIRST$ and $LAST$ and their most efficient variants dedicated to unranking generation of choice functions of specific classes of the models. The investigated notion "the Q property" enables us to explain the model properties leading to existence of backtracking-less algorithm. Then, the algorithms $UNRANK$, $RANK$ and $RANGE$ for ranking generation of choice functions are devloped. The tables D are the essential components of the developed algorithms concerning the rank. The sequential, parallel or distributed systems for exhaustive generation of choice functions are given. We demonstrate also usefulness of the presented here approach into solving the general tasks for the models that the Q property does not hold. We have shown backtracking and looking forward algorithms showing simultaneously the structure of efforts undertaken in order to diminish as much as possible the number of backtracking and looking forward steps. We have also shown development of digital signature system basing on the given theory. The results concerning non exhaustive generation are for the first time so strongly unified supplying convenient tools for implementation and futher development of optimisation algorithms especially the genetic algorithms.

Modelowanie Arbitralnych Zbiorów Obiektów Kombinatorycznych oraz ich Sekwencyjna i Równoległa Generacja

Streszczenie

Obiektem kombinatorycznym w sensie niniejszej pracy jest każda struktura jak zbiór, graf, tablica, itp., skonstruowana na podstawie danych reprezentujących inną strukturę oraz o reguły tworzenia takiego obiektu kombinatorycznego. Przez podstawowe obiekty kombinatoryczne rozumiemy permutacje i wariacje, kombinacje, podziały liczb lub zbiorów, dystrybucje oraz kompozycje. Każdy zbiór obiektów kombinatorycznych może byc zdefiniowany w oparciu o odpowiadający zbiór podstawowych obiektów kombinatorycznych. W pracy przedstawiono metody modelowania oraz generacji zbiorów obiektów kombinatorycznych. Podstawową reprezentacją obiektów kombinatorycznych jest funkcja wyboru h rodziny indeksowej $< \mathcal{G}_i >$, $1 \leq i \leq m$, rozumiana jako odwzorowanie $i \rightarrow q_i$, gdzie $i \in I$, $q_i \in \mathcal{G}_i$, $I = \{1, 2, ..., m\}$. Dowolny zbiór obiektow kombinatorycznych odpowiada zbiorowi funkcji wyboru $\{h\}_{\mathcal{G}_i}$ jakiegoś modelu nierankingowego S_U, którego ogólna struktura wyrażana jest następująco: $S_U =<< \mathcal{G}_i >, 1 \leq i \leq m; W; W_1; \{h\}_{\mathcal{G}_i} >$, gdzie W oraz W_1 określają warunki, jakie dodatkowo musi spełniać każda funkcja wyboru $h \in \{h\}_{\mathcal{G}_i}$. Warunek W określa jedną z podstawowych klas funkcji wyboru, podczas gdy W_1 pozwala zdefiniować dowolną jej podklasę. W modelowaniu obiektów kombinatorycznych znajdują zastosowanie klasy rosnących funkcji wyboru, klasy funkcji monotonicznych oraz klasy bijekcji.

W pierwszych dwóch rozdziałach pracy przedstawiono podstawowe właściwości modeli reprezentacyjnych znajdujące zastosowanie w modelowaniu arbitralnych zbiorow obiektów kombinatorycznych, pokazano modele zbiorów podstawowych obiektow kombinatorycznych.

W dalszej cześci pracy przedmiotem rozważań jest struktura zbioru $\{h\}_{\mathcal{G}_i}$. Istotnym wyróżnikiem tej struktury jest odpowiednia tablica D, której elementy są ogólnie zdefiniowanymi liczbami strukturalnymi. W szczególnych przypadkach liczbami strukturalnymi są elementy trójkata Pascala lub liczby Stirlinga. W oparciu o tablice D rozważany jest izomorfizm oraz równoważność modeli reprezentacyjnych. W celu przedstawienia podstawowej filozofii transformowania modeli reprezentacyjnych istotne znaczenie ma twierdzenie rankingowe o charakterze egzystencjalnym. Przedstawione rozważania służą do rozwinięcia koncepcji reprezentacyjnego modelu rankingowego zdefiniowanego jako struktura, $S_{R(h)} =< S_U, <^h$, tablica D $>$, gdzie $<^h$ oznacza porządek na zbiorze $\{h\}_{\mathcal{G}_i}$, pochodną porzadku $<^h$ jest ranking $R(h)$ rozumiany jako odwzorowanie $N \rightarrow \{h\}_{\mathcal{G}_i}$, $N = \{1, 2, ..., | \{h\}_{\mathcal{G}_i} |\}$. Własności modeli rankingowych oraz nierankingowych wykorzystano następnie dla wprowadzenia i rozwinięcia modeli generacyjnych, których podstawowymi komponentami jest system $< Methods >$ zawiera-

jący istotne dla generacji procedury *FIRST*, *LAST*, *NEXT*, *UNRANK*, *RANK*, *RANGE* oraz procedury pomocnicze.

Rozdział 6 przedstawia podstawowe procedury z tego systemu oraz ich najważniejsze warianty w zależności od własności modeli reprezentacyjnych.

Rozdział 7 rozważa generację zupełną zbiorów funkcji wyboru w systemach sekwencyjnych, rozproszonych oraz równoległych przy wykorzystaniu przedstawionych powyżej procedur. Istotną własnością modeli reprezentacyjnych jest własność Q oznaczająca istnienie beznawrotowych algorytmów generacyjnych. Rozważania przedstawione w pierwszych siedmiu rozdziałach koncentrują się wokół modeli posiadających własność Q.

Rozdział 8 zajmuje się różnymi zagadnieniami dotyczącymi szerszych klas zadań i reprezentacji, przede wszystkim rozważane są warianty niezupełnej generacji zbiorów funkcji wyboru oraz generacji, w przypadku gdy własność Q nie jest spełniona. Pokazano strukturę wysiłków, które należy podjąć celem uzyskania algorytmów o możliwie najmniejszej liczbie nawrotów. Wynik ten wykorzystano dla rozwinięcia algorytmu generacji bijekcji wyboru. Ważnym wynikiem szczegółowym jest przedstawiony system podpisu elektronicznego rozwinięty w oparciu o wprowadzone modele hierarchiczne. Ciekawym prognostykiem dla przeszłych badań jest strukturalne podobieństwo wyszukiwania przeprowadzanego w maszynach do głębokiego wyszukiwania oraz generowania funkcji wyboru.