Jean Dezert[1], Florentin Smarandache[2]

[1]ONERA, 29 Av. de la Division Leclerc 92320, Chatillon, France

[2]Department of Mathematics University of New Mexico Gallup, NM 8730, U.S.A.

The generation of hyper-power sets

Published in:
Florentin Smarandache & Jean Dezert (Editors)
Advances and Applications of DSmT for Information Fusion
(Collected works), Vol. I
American Research Press, Rehoboth, 2004
ISBN: 1-931233-82-9
Chapter II, pp. 37 - 48

Abstract: *The development of DSmT is based on the notion of Dedekind's lattice, called also hyper-power set in the DSmT framework, on which is defined the general basic belief assignments to be combined. In this chapter, we explain the structure of the hyper-power set, give some examples of hyper-power sets and show how they can be generated from isotone Boolean functions. We also show the interest to work with the hyper-power set rather than the power set of the refined frame of discernment in terms of complexity.*

2.1 Introduction

One of the cornerstones of the DSmT is the notion of Dedekind's lattice, coined as hyper-power set by the authors in literature, which will be defined in next section. The starting point is to consider $\Theta = \{\theta_1, \dots, \theta_n\}$ as a set of n elements which cannot be precisely defined and separated so that no refinement of Θ in a new larger set Θ_{ref} of disjoint elementary hypotheses is possible. This corresponds to the free DSm model. This model is justified by the fact that in some fusion problems (mainly those manipulating vague or continuous concepts), the refinement of the frame is just impossible to obtain; nevertheless the fusion still applies when working on Dedekind's lattice and based on the DSm rule of

This chapter is based on a paper [6] presented during the International Conference on Information Fusion, Fusion 2003, Cairns, Australia, in July 2003 and is reproduced here with permission of the International Society of Information Fusion.

combination. With the DSmT approach, the refinement of the frame is not prerequisite for managing properly the combination of evidences and one can abandon Shafer's model in general. Even if Shafer's model is justified and adopted in some cases, the hybrid DSm rule of combination appears to be a new interesting and preferred alternative for managing high conflicting sources of evidence. Our approach actually follows the footprints of our predecessors like Yager [23] and Dubois and Prade [7] to circumvent the problem of the applicability of Dempster's rule face to high conflicting sources of evidence but with a new mathematical framework. The major reason for attacking the problem directly from the bottom level, i.e. the free DSm model comes from the fact that in some real-world applications observations/concepts are not unambiguous. The ambiguity of observations is explained by Goodman, Mahler and Nguyen in [9] pp. 43-44. Moreover, the ambiguity can also come from the granularity of knowledge, known as Pawlak's indiscernability or roughness [15].

2.2 Definition of hyper-power set D^Θ

The *hyper-power set* D^Θ is defined as the set of all composite propositions built from elements of Θ with $\cup$ and $\cap$ (Θ generates D^Θ under operators $\cup$ and $\cap$) operators such that

1. $\emptyset, \theta_1, \ldots, \theta_n \in D^\Theta$.
2. If $A, B \in D^\Theta$, then $A \cap B \in D^\Theta$ and $A \cup B \in D^\Theta$.
3. No other elements belong to D^Θ, except those obtained by using rules 1 or 2.

The dual (obtained by switching $\cup$ and $\cap$ in expressions) of D^Θ is itself. There are elements in D^Θ which are self-dual (dual to themselves), for example α_8 for the case when $n = 3$ in the example given in the next section. The cardinality of D^Θ is majored by 2^{2^n} when $\text{Card}(\Theta) = |\Theta| = n$. The generation of hyper-power set D^Θ is closely related with the famous Dedekind problem [4, 3] on enumerating the set of monotone Boolean functions as it will be presented in the sequel with the generation of the elements of D^Θ.

2.3 Example of the first hyper-power sets

- In the degenerate case ($n = 0$) where $\Theta = \{\}$, one has $D^\Theta = \{\alpha_0 \triangleq \emptyset\}$ and $|D^\Theta| = 1$.
- When $\Theta = \{\theta_1\}$, one has $D^\Theta = \{\alpha_0 \triangleq \emptyset, \alpha_1 \triangleq \theta_1\}$ and $|D^\Theta| = 2$.
- When $\Theta = \{\theta_1, \theta_2\}$, one has $D^\Theta = \{\alpha_0, \alpha_1, \ldots, \alpha_4\}$ and $|D^\Theta| = 5$ with $\alpha_0 \triangleq \emptyset$, $\alpha_1 \triangleq \theta_1 \cap \theta_2$, $\alpha_2 \triangleq \theta_1$, $\alpha_3 \triangleq \theta_2$ and $\alpha_4 \triangleq \theta_1 \cup \theta_2$.
- When $\Theta = \{\theta_1, \theta_2, \theta_3\}$, the elements of $D^\Theta = \{\alpha_0, \alpha_1, \ldots, \alpha_{18}\}$ and $|D^\Theta| = 19$ (see [5] for details) are now given by (following the informational strength indexation explained in the next chapter):

Elements of $D^{\Theta=\{\theta_1,\theta_2,\theta_3\}}$

$\alpha_0 \triangleq \emptyset$	
$\alpha_1 \triangleq \theta_1 \cap \theta_2 \cap \theta_3$	$\alpha_{10} \triangleq \theta_2$
$\alpha_2 \triangleq \theta_1 \cap \theta_2$	$\alpha_{11} \triangleq \theta_3$
$\alpha_3 \triangleq \theta_1 \cap \theta_3$	$\alpha_{12} \triangleq (\theta_1 \cap \theta_2) \cup \theta_3$
$\alpha_4 \triangleq \theta_2 \cap \theta_3$	$\alpha_{13} \triangleq (\theta_1 \cap \theta_3) \cup \theta_2$
$\alpha_5 \triangleq (\theta_1 \cup \theta_2) \cap \theta_3$	$\alpha_{14} \triangleq (\theta_2 \cap \theta_3) \cup \theta_1$
$\alpha_6 \triangleq (\theta_1 \cup \theta_3) \cap \theta_2$	$\alpha_{15} \triangleq \theta_1 \cup \theta_2$
$\alpha_7 \triangleq (\theta_2 \cup \theta_3) \cap \theta_1$	$\alpha_{16} \triangleq \theta_1 \cup \theta_3$
$\alpha_8 \triangleq (\theta_1 \cap \theta_2) \cup (\theta_1 \cap \theta_3) \cup (\theta_2 \cap \theta_3)$	$\alpha_{17} \triangleq \theta_2 \cup \theta_3$
$\alpha_9 \triangleq \theta_1$	$\alpha_{18} \triangleq \theta_1 \cup \theta_2 \cup \theta_3$

Note that the *classical* complementary $\bar{A}$ of any proposition A (except for $\emptyset$ and Θ), is not involved within the free DSm model because of the refutation of the third excluded middle; it can however be introduced if necessary when dealing with hybrid models as it will be shown in chapter 4 if we introduce explicitly some exclusivity constraints into the free DSm model when one has no doubt on the exclusivity between given elements of Θ depending on the nature of the fusion problem. $|D^\Theta|$ for $n \geq 1$ follows the sequence of Dedekind's numbers[1] 1, 2, 5, 19, 167, 7580, 7828353, 56130437228687557907787... [17]. Note also that this huge number of elements of hyper-power set is comparatively far less than the total number of elements of the power set of the refined frame Θ_{ref} if one would to work on $2^{\Theta_{ref}}$ and if we admit the possibility that such refinement exists as it will be seen in section 2.4.1.

2.4 The generation of D^Θ

2.4.1 Memory size requirements and complexity

Before going further on the generation of D^Θ, it is important to estimate the memory size for storing the elements of D^Θ for $|\Theta| = n$. Since each element of D^Θ can be stored as a $2^n - 1$-binary string, the memory size for D^Θ is given by the right column of the following table (we do not count the size for $\emptyset$ which is 0 and the minimum length is considered here as the byte (8 bits)):

[1]Actually this sequence corresponds to the sequence of Dedekind minus one since we don't count the last degenerate isotone function $f_{2^{2^n}-1}(.)$ as element of D^Θ (see section 2.4).

$\|\Theta\| = n$	size/elem.	# of elem.	Size of D^Θ
2	1 byte	4	4 bytes
3	1 byte	18	18 bytes
4	2 bytes	166	0.32 Kb
5	4 bytes	7579	30 Kb
6	8 bytes	7828352	59 Mb
7	16 bytes	$\approx 2.4 \cdot 10^{12}$	$3.6 \cdot 10^4$ Gb
8	32 bytes	$\approx 5.6 \cdot 10^{22}$	$1.7 \cdot 10^{15}$ Gb

This table shows the extreme difficulties for our computers to store all the elements of D^Θ when $|\Theta| > 6$. This complexity remains however smaller than the number of all Boolean functions built from the ultimate refinement (if accessible) $2^{\Theta_{ref}}$ of same initial frame Θ for applying DST. The comparison of $|D^\Theta|$ with respect to $|2^{\Theta_{ref}}|$ is given in the following table

$\|\Theta\| = n$	$\|D^\Theta\|$	$\|2^{\Theta_{ref}}\| = 2^{2^n - 1}$
2	5	$2^3 = 8$
3	19	$2^7 = 128$
4	167	$2^{15} = 32768$
5	7580	$2^{31} = 2147483648$

Fortunately, in most fusion applications only a small subset of elements of D^Θ have a non null basic belief mass because all the commitments are just usually impossible to assess precisely when the dimension of the problem increases. Thus, it is not necessary to generate and keep in memory all elements of D^Θ or $2^{\Theta_{ref}}$ but only those which have a positive belief mass. However there is a real technical challenge on how to manage efficiently all elements of the hyper-power set. This problem is obviously more difficult when working on $2^{\Theta_{ref}}$. Further investigations and research have to be carried out to develop implementable engineering solutions for managing high dimensional problems when the basic belief functions are not degenerated (i.e. all $m(A) > 0$, $A \in D^\Theta$ or $A \in 2^{\Theta_{ref}}$).

2.4.2 Monotone Boolean functions

A simple *Boolean function* $f(.)$ maps n-binary inputs $(x_1, \ldots, x_n) \in \{0,1\}^n \triangleq \{0,1\} \times \ldots \times \{0,1\}$ to a single binary output $y = f(x_1, \ldots, x_n) \in \{0,1\}$. Since there are 2^n possible input states which can map to either 0 or 1 at the output y, the number of possible Boolean functions is 2^{2^n}. Each of these functions can be realized by the logic operations $\wedge$ (and), $\vee$ (or) and $\neg$ (not) [3, 21]. As a simple example, let's consider only a 2-binary input variable $(x_1, x_2) \in \{0,1\} \times \{0,1\}$ then all the $2^{2^2} = 16$ possible Boolean functions $f_i(x_1, x_2)$ built from (x_1, x_2) are summarized in the following tables:

(x_1,x_2)	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
$(0,0)$	0	0	0	0	0	0	0	0
$(0,1)$	0	0	0	0	1	1	1	1
$(1,0)$	0	0	1	1	0	0	1	1
$(1,1)$	0	1	0	1	0	1	0	1
Notation	False	$x_1 \wedge x_2$	$x_1 \wedge \bar{x}_2$	x_1	$\bar{x}_1 \wedge x_2$	x_2	$x_1 \veebar x_2$	$x_1 \vee x_2$

(x_1,x_2)	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
$(0,0)$	1	1	1	1	1	1	1	1
$(0,1)$	0	0	0	0	1	1	1	1
$(1,0)$	0	0	1	1	0	0	1	1
$(1,1)$	0	1	0	1	0	1	0	1
Notation	$x_1 \bar{\nabla} x_2$	$x_1 \triangle x_2$	$\bar{x}_2$	$x_1 \vee \bar{x}_2$	$\bar{x}_1$	$\bar{x}_1 \vee x_2$	$x_1 \bar{\wedge} x_2$	True

with the notation $\bar{x} \triangleq \neg x$, $x_1 \veebar x_2 \triangleq (x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$ (xor), $x_1 \bar{\nabla} x_2 \triangleq \neg(x_1 \vee x_2)$ (nor), $x_1 \triangle x_2 \triangleq (x_1 \wedge x_2) \vee (\bar{x}_1 \wedge \bar{x}_2)$ (xnor) and $x_1 \bar{\wedge} x_2 \triangleq \neg(x_1 \wedge x_2)$ (nand).

We denote by $\mathcal{F}_n(\wedge,\vee,\neg) = \{f_0(x_1,\ldots,x_n),\ldots,f_{2^{2^n}-1}(x_1,\ldots,x_n)\}$ the set of all possible Boolean functions built from n-binary inputs. Let $\mathbf{x} \triangleq (x_1,\ldots,x_n)$ and $\mathbf{x}' \triangleq (x'_1,\ldots,x'_n)$ be two vectors in $\{0,1\}^n$. Then $\mathbf{x}$ precedes $\mathbf{x}'$ and we denote $\mathbf{x} \preceq \mathbf{x}'$ if and only if $x_i \leq x'_i$ for $1 \leq i \leq n$ ($\leq$ is applied componentwise). If $x_i < x'_i$ for $1 \leq i \leq n$ then $\mathbf{x}$ strictly precedes $\mathbf{x}'$ which will be denoted as $\mathbf{x} \prec \mathbf{x}'$.

A Boolean function f is said to be a *non-decreasing monotone (or isotone) Boolean function* (or just *monotone Boolean function* for short) if and only if $\forall \mathbf{x},\mathbf{x}' \in \{0,1\}^n$ such that $\mathbf{x} \preceq \mathbf{x}'$, then $f(\mathbf{x}) \preceq f(\mathbf{x}')$ [19]. Since any isotone Boolean function involves only $\wedge$ and $\vee$ operators (no $\neg$ operations) [21] and there exists a parallel between $(\vee,\wedge)$ operators in logics with $(+,\cdot)$ in algebra of numbers and $(\cup,\cap)$ in algebra of sets, the generation of all elements of D^Θ built from Θ with $\cup$ and $\cap$ operator is equivalent to the problem of generating isotone Boolean functions over the vertices of the unit n-cube. We denote by $\mathcal{M}_n(\wedge,\vee)$ the set of all possible monotone Boolean functions built from n-binary inputs. $\mathcal{M}_n(\wedge,\vee)$ is a subset of $\mathcal{F}_n(\wedge,\vee,\neg)$. In the previous example, $f_1(x_1,x_2)$, $f_3(x_1,x_2)$, $f_5(x_1,x_2)$, $f_7(x_1,x_2)$ are isotone Boolean functions but special functions $f_0(x_1,x_2)$ and $f_{2^{2^n}-1}(x_1,\ldots,x_n)$ must also be considered as monotone functions too. All the other functions belonging to $\mathcal{F}_2(\wedge,\vee,\neg)$ do not belong to $\mathcal{M}_2(\wedge,\vee)$ because they require the $\neg$ operator in their expressions and we can check easily that the monotonicity property $\mathbf{x} \preceq \mathbf{x}' \Rightarrow f(\mathbf{x}) \preceq f(\mathbf{x}')$ does not hold for these functions.

The Dedekind's problem [4] is to determine the number $d(n)$ of *distinct* monotone Boolean functions of n-binary variables. Dedekind [4] computed $d(0) = 2$, $d(1) = 3$, $d(2) = 6$, $d(3) = 20$ and $d(4) = 168$. Church [1] computed $d(5) = 7581$ in 1940. Ward [20] computed $d(6) = 7828354$ in 1946. Church [2] then computed $d(7) = 2414682040998$ in 1965. Between sixties and eighties, important advances have been done to obtain upper and lower bounds for $d(n)$ [10, 12, 14]. In 1991, Wiedemann [22] computed $d(8) = 56130437228687557907788$ (200 hours of computing time with a Cray-2 processor) which has recently been validated by Fidytek and al. in [8]. Until now the computation of $d(n)$ for $n > 8$ is still a challenge for mathematicians even if the following direct exact explicit formula for $d(n)$ has been obtained by Kisielewicz and Tombak (see [11, 18] for proof) :

$$d(n) = \sum_{k=1}^{2^{2^n}} \prod_{j=1}^{2^n-1} \prod_{i=0}^{j-1} (1 - b_i^k(1 - b_j^k)) \prod_{m=0}^{l(i)} (1 - b_m^i(1 - b_m^j))) \tag{2.1}$$

where $l(0) = 0$ and $l(i) = [\log_2 i]$ for $i > 0$, $b_i^k \triangleq [k/2^i] - 2[k/2^{i+1}]$ and $[x]$ denotes the floor function (i.e. the nearest integer less or equal to x). The difficulty arises from the huge number of terms involved in the formula, the memory size and the high speed computation requirements. The last advances and state of art in counting algorithms of Dedekind's numbers can be found in [18, 8, 19].

2.4.3 Generation of MBF

Before describing the general algorithm for generating the monotone Boolean functions (MBF), let examine deeper the example of section 2.4.2. From the previous tables, one can easily find the set of (restricted) MBF $\mathcal{M}_2^\star(\wedge, \vee) = \{f_0(x_1, x_2) = \text{False}, f_1(x_1, x_2) = x_1 \wedge x_2, f_5(x_1, x_2) = x_2, f_7(x_1, x_2) = x_1 \vee x_2\}$ which is equivalent, using algebra of sets, to hyper-power set $D^X = \{\emptyset, x_1 \cap x_2, x_1, x_2, x_1 \cup x_2\}$ associated with frame of discernment $X = \{x_1, x_2\}$. Since the tautology $f_{15}(x_1, x_2)$ is not involved within DSmT, we do not include it as a proper element of D^X and we consider only $\mathcal{M}_2^\star(\wedge, \vee) \triangleq \mathcal{M}_2(\wedge, \vee) \setminus \{f_{15}\}$ rather than $\mathcal{M}_2(\wedge, \vee)$ itself.

Let's now introduce Smarandache's codification for the enumeration of distinct parts of a Venn diagram X with n partially overlapping elements x_i,$i = 1, 2, \ldots, n$. Such a diagram has $2^n - 1$ disjoint parts. One denotes with only one digit (or symbol) those parts which belong to only one of the elements x_i (one denotes by $< i >$ the part which belongs to x_i only, for $1 \leq i \leq n$), with only two digits (or symbols) those parts which belong to exactly two elements (one denotes by $< ij >$, with $i < j$, the part which belongs to x_i and x_j only, for $1 \leq i < j \leq n$), then with only three digits (or symbols) those parts which belong to exactly three elements (one denotes by $< ijk >$ concatenated numbers, with $i < j < k$, the part which belongs to x_i, x_j, and x_k only, for $1 \leq i < j < k \leq n$), and so on up to $< 12 \ldots n >$ which represents the last part that belongs to all elements x_i. For $1 \leq n \leq 9$, Smarandache's encoding works normally as in base 10. But, for $n \geq 10$, because there occur two (or more) digits/symbols in notation of the elements starting from 10 on, one considers this codification in base $n+1$, i.e. using one symbol to represent two (or more) digits, for example: $A = 10$, $B = 11$, $C = 12$, etc.

- For $n = 1$ one has only one part, coded $< 1 >$.
- For $n = 2$ one has three parts, coded $< 1 >$, $< 2 >$, $< 12 >$. Generally, $< ijk >$ does not represent $x_i \cap x_j \cap x_k$ but only a part of it, the only exception is for $< 12 \ldots n >$.
- For $n = 3$ one has $2^3 - 1 = 7$ disjoint parts, coded $< 1 >$, $< 2 >$, $< 3 >$, $< 12 >$, $< 13 >$, $< 23 >$, $< 123 >$. $< 23 >$ means the part which belongs to x_2 and x_3 only, but $< 23 > \neq x_2 \cap x_3$ because $x_2 \cap x_3 = \{< 23 >, < 123 >\}$ in the Venn diagram of 3 elements x_1, x_2, and x_3 (see next chapter).
- The generalization for $n > 3$ is straightforward. Smarandache's codification can be organized in a numerical increasing order, in lexicographic order or any other orders.

A useful order for organizing Smarandache's codification for the generation of D^X is the *DSm-order* $\mathbf{u}_n = [u_1, \ldots, u_{2^n-1}]'$ based on a recursive construction starting with $\mathbf{u}_1 \triangleq [< 1 >]$. Having constructed $\mathbf{u}_{n-1}$, then we can construct $\mathbf{u}_n$ for $n > 1$ recursively as follows:

- include all elements of $\mathbf{u}_{n-1}$ into $\mathbf{u}_n$;
- afterwards, include element $< n >$ as well in $\mathbf{u}_n$;
- then at the end of each element of $\mathbf{u}_{n-1}$ concatenate the element $< n >$ and get a new set $\mathbf{u'}_{n-1}$ which then is also included in $\mathbf{u}_n$.

This is $\mathbf{u}_n$, which has $(2^{n-1} - 1) + 1 + (2^{n-1} - 1) = 2^n - 1$ components.

For $n = 3$, as example, one gets $\mathbf{u}_3 \triangleq [< 1 > < 2 > < 12 > < 3 > < 13 > < 23 > < 123 >]'$. Because all elements in $\mathbf{u}_n$ are disjoint, we are able to write each element d_i of D^X in a unique way as a linear combination of $\mathbf{u}_n$ elements, i.e.

$$\mathbf{d}_n = [d_1, \ldots, d_{2^n-1}]' = \mathbf{D}_n \cdot \mathbf{u}_n \tag{2.2}$$

Thus $\mathbf{u}_n$ constitutes a basis for generating the elements of D^X. Each row in the matrix $\mathbf{D}_n$ represents the coefficients of an element of D^X with respect to the basis $\mathbf{u}_n$. The rows of $\mathbf{D}_n$ may also be regarded as binary numbers in an increasing order.

Example: For $n = 2$, one has:

$$\underbrace{\begin{bmatrix} d_1 = x_1 \cap x_2 \\ d_2 = x_2 \\ d_3 = x_1 \\ d_4 = x_1 \cup x_2 \end{bmatrix}}_{\mathbf{d}_2} = \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}}_{\mathbf{D}_2} \cdot \underbrace{\begin{bmatrix} <1> \\ <2> \\ <12> \end{bmatrix}}_{\mathbf{u}_2} \tag{2.3}$$

where the "matrix product" is done after identifying $(+, \cdot)$ with $(\cup, \cap)$, $0 \cdot <x>$ with $\emptyset$ and $1 \cdot <x>$ with $<x>$.

The generation of D^X is then strictly equivalent to generate $\mathbf{u}_n$ and matrix $\mathbf{D}_n$ which can be easily obtained by the following recursive procedure:

- start with $\mathbf{D}_0^c = [0\,1]'$ corresponding to all Boolean functions with no input variable ($n = 0$).
- build the $\mathbf{D}_1^c$ matrix from each row $\mathbf{r}_i$ of $\mathbf{D}_0^c$ by adjoining it to any other row $\mathbf{r}_j$ of $\mathbf{D}_0^c$ such that $\mathbf{r}_i \cup \mathbf{r}_j = \mathbf{r}_j$. This is equivalent here to add either 0 or 1 in front (i.e. left side) of $\mathbf{r}_1 \equiv 0$ but only 1 in front of $\mathbf{r}_2 \equiv 1$. Since the tautology is not involved in the hyper-power set, then one has to remove the first column and the last line of

$$\mathbf{D}_1^c = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \quad \text{to obtain finally} \quad \mathbf{D}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- build $\mathbf{D}_2^c$ from $\mathbf{D}_1^c$ by adjoining to each row $\mathbf{r}_i$ of $\mathbf{D}_1^c$, any row $\mathbf{r}_j$ of $\mathbf{D}_1^c$ such that $\mathbf{r}_i \cup \mathbf{r}_j = \mathbf{r}_j$ and then remove the first column and the last line of $\mathbf{D}_2^c$ to get $\mathbf{D}_2$ as in (2.3).
- build $\mathbf{D}_3^c$ from $\mathbf{D}_2^c$ by adjoining to each row $\mathbf{r}_i$ of $\mathbf{D}_2^c$ any row $\mathbf{r}_j$ of $\mathbf{D}_2^c$ such that $\mathbf{r}_i \cup \mathbf{r}_j = \mathbf{r}_j$ and then remove the first column and the last line of $\mathbf{D}_3^c$ to get $\mathbf{D}_3$ given by (where $\mathbf{D}'$ denotes here the transposed of the matrix $\mathbf{D}$)

$$\mathbf{D}_3' = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&1 \\ 0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&1&0&1 \\ 0&0&0&1&1&1&0&0&1&1&1&0&1&1&1&1&1&1&1 \\ 0&0&1&0&1&1&0&1&0&1&1&1&1&1&0&1&1&1&1 \\ 0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \end{bmatrix}$$

- Likewise, $\mathbf{D}_n^c$ is built from $\mathbf{D}_{n-1}^c$ by adjoining to each row $\mathbf{r}_i$ of $\mathbf{D}_{n-1}^c$ any row $\mathbf{r}_j$ of $\mathbf{D}_{n-1}^c$ such that $\mathbf{r}_i \cup \mathbf{r}_j = \mathbf{r}_j$. Then $\mathbf{D}_n$ is obtained by removing the first column and the last line of $\mathbf{D}_n^c$.

Example for $\Theta = \{\theta_1, \theta_2, \theta_3\}$: Note that the new indexation of elements of D^Θ now follows the MBF generation algorithm.

$$\underbrace{\begin{bmatrix} \alpha_0 \triangleq \emptyset \\ \alpha_1 \triangleq \theta_1 \cap \theta_2 \cap \theta_3 \\ \alpha_2 \triangleq \theta_2 \cap \theta_3 \\ \alpha_3 \triangleq \theta_1 \cap \theta_3 \\ \alpha_4 \triangleq (\theta_1 \cup \theta_2) \cap \theta_3 \\ \alpha_5 \triangleq \theta_3 \\ \alpha_6 \triangleq \theta_1 \cap \theta_2 \\ \alpha_7 \triangleq (\theta_1 \cup \theta_3) \cap \theta_2 \\ \alpha_8 \triangleq (\theta_2 \cup \theta_3) \cap \theta_1 \\ \alpha_9 \triangleq (\theta_1 \cap \theta_2) \cup (\theta_1 \cap \theta_3) \cup (\theta_2 \cap \theta_3) \\ \alpha_{10} \triangleq (\theta_1 \cap \theta_2) \cup \theta_3 \\ \alpha_{11} \triangleq \theta_2 \\ \alpha_{12} \triangleq (\theta_1 \cap \theta_3) \cup \theta_2 \\ \alpha_{13} \triangleq (\theta_2 \cup \theta_3) \\ \alpha_{14} \triangleq \theta_1 \\ \alpha_{15} \triangleq (\theta_2 \cap \theta_3) \cup \theta_1 \\ \alpha_{16} \triangleq (\theta_1 \cup \theta_3) \\ \alpha_{17} \triangleq (\theta_1 \cup \theta_2) \\ \alpha_{18} \triangleq (\theta_1 \cup \theta_2 \cup \theta_3) \end{bmatrix}}_{\mathbf{d}_3} = \underbrace{\begin{bmatrix} 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&1 \\ 0&0&0&0&1&0&1 \\ 0&0&0&0&1&1&1 \\ 0&0&0&1&1&1&1 \\ 0&0&1&0&0&0&1 \\ 0&0&1&0&0&1&1 \\ 0&0&1&0&1&0&1 \\ 0&0&1&0&1&1&1 \\ 0&0&1&1&1&1&1 \\ 0&1&1&0&0&1&1 \\ 0&1&1&0&1&1&1 \\ 0&1&1&1&1&1&1 \\ 1&0&1&0&1&0&1 \\ 1&0&1&0&1&1&1 \\ 1&0&1&1&1&1&1 \\ 1&1&1&0&1&1&1 \\ 1&1&1&1&1&1&1 \end{bmatrix}}_{\mathbf{D}_3} \cdot \underbrace{\begin{bmatrix} <1> \\ <2> \\ <12> \\ <3> \\ <13> \\ <23> \\ <123> \end{bmatrix}}_{\mathbf{u}_3}$$

For convenience, we provide in appendix the source code in Matlab[2] language to generate D^Θ. This code includes the identification of elements of D^Θ corresponding to each monotone Boolean function according to Smarandache's codification.

2.5 Conclusion

In this chapter, one has introduced the notion of Dedekind's lattice D^Θ (hyper-power set) on which are defined basic belief functions in the framework of DSmT and the acceptance of the free DSm model. The justification of the free DSm model as a starting point (ground level) for the development of our new theory of plausible and paradoxical reasoning for information fusion has been also given and arises from the necessity to deal with possibly ambiguous concepts which can appear in real-world applications. The lower complexity of the hyper-power set with respect to the complexity of the classical refined power set

[2]Matlab is a trademark of The MathWorks, Inc.

$2^{\Theta_{ref}}$ has been clearly demonstrated here. We have proven the theoretical link between the generation of hyper-power set with Dedekind's problem on counting isotone Boolean functions. A theoretical solution for generating lattices D^{Θ} has been presented and a MatLab source code has been provided for users convenience.

2.6 References

[1] Church R., *Numerical analysis of certain free distributive structures*, Duke Math. J., Vol. 6, no. 3, pp. 732-734, 1940.

[2] Church R., *Enumeration by rank of the elements of the free distributive lattice with seven generators*, Not. Amer. Math. Soc., Vol. 12, p. 724, 1965.

[3] Comtet L., *Sperner Systems*, sec.7.2 in Advanced Combinatorics: The Art of Finite and Infinite Expansions, D. Reidel Publ. Co., pp. 271-273, 1974.

[4] Dedekind R. *Über Zerlegungen von Zahlen durch ihre grössten gemeinsammen Teiler*, In Gesammelte Werke, Bd. 1. pp. 103-148, 1897.

[5] Dezert J., *Fondations pour une nouvelle théorie du raisonnement plausible et paradoxal*, ONERA Tech. Rep. RT 1/06769/DTIM, Jan. 2003.

[6] Dezert J., Smarandache F., *On the generation of hyper-power sets*, Proc. of Fusion 2003, Cairns, Australia, July 8-11, 2003.

[7] Dubois D., Prade H., *Representation and combination of uncertainy with belief functions and possibility measures*, Computational Intelligence, 4, pp. 244-264, 1988.

[8] Fidytek R., Mostowski A.W., Somla R., Szepietowski A., *Algorithms counting monotone Boolean functions*, Inform. Proc. Letters, Vol. 79, pp. 203-209, 2001.

[9] Goodman I.R., Mahler R.P.S., Nguyen H.T., *Mathematics of Data Fusion*, Kluwer Academic Press, Boston, 1997.

[10] Hansel G., *Sur le nombre des fonctions booléennes monotones de n variables*, C.R. Acad. Sci. Paris, Serie A-B, p. 262, 1966.

[11] Kisielewicz A., *A solution of Dedekind's problem on the number of isotone Boolean functions*, J. reine angew. math., Vol. 386, pp. 139-144, 1988.

[12] Kleitman D., *On Dedekind's problem: The number of Boolean functions*, Proc. of the Amer. Math Society, Vol. 21, pp. 677-682, 1969.

[13] Kleitman D., Markowsky G., *On Dedekind's problem: The number of isotone Boolean functions. II*, Trans. of the Amer. Math. Soc., Vol. 213, 1976.

[14] Korshunov A., *On the number of Boolean functions*, Prob. kibernetiki, Vol. 38, pp. 5-108, 1981.

[15] Pawlak Z., *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, Dortrecht, 1991.

[16] Shapiro H.N. *On the counting problem for monotone Boolean functions*, Comm. on pure and applied math., Vol. XXIII, pp. 299-312, 1970.

[17] Sloane N.J.A., *The On-line Encyclopedia of Integer Sequences 2003*, (Sequence No. A014466), `http://www.research.att.com/~njas/sequences/`.

[18] Tombak M., Isotamm A., Tamme T., *On logical method for counting Dedekind numbers*, Lect. Notes on Comp.Sci., 2138, p. 424-427, Springer-Verlag, 2001. `www.cs.ut.ee/people/m_tombak/publ.html`.

[19] Triantaphyllou, E., Torvik V.I., *Inference of Monotone Boolean Functions*, Encyclopedia of Optimization, (P.M. Pardalos and C. Floudas, Eds.), Kluwer Acad. Publi., Boston, Vol. 2, pp. 472-480., 2001.

[20] Ward M., *Note on the order of free distributive lattices* , Bull. Amer. Math. Soc., Vol. 52, no. 5, p. 423, 1946.

[21] Weisstein E. W., *CRC Concise Encyclopedia of Mathematics* , CRC Press; 2nd edition, 3252 pages, Dec. 2002. `http://mathworld.wolfram.com/`.

[22] Wiedemann D., *A computation of the eighth Dedekind number*, Order, no. 8, pp. 5-6, 1991.

[23] Yager R.R., *On the Dempster-Shafer Framework and New Combination Rules*, Information Sciences, Vol. 41, pp.93-137, 1987.

Appendix: MatLab code for generating hyper-power sets

```
%***********************************************
% Copyright (c)  2003 J.Dezert and F.Smarandache
%
% Purpose: Generation of D^Theta for the DSmT for
% Theta={theta 1,..,Theta n}. Due to the huge
% # of elements of D^Theta. only cases up to n<7
% are usually tractable on computers.
%***********************************************
n=input('Enter cardinality for Theta (0<n<6) ?');
% Generation of the Smarandache codification
% Note: this should be implemented using
% character strings for n>9
u n=[1];
for nn=2:n
u n=[u n nn (u n*10+nn*ones(1,size(u n*10,2)))];
end
% Generation of D n (isotone boolean functions)
D n1=[0;1];
for nn=1:n, D n=[];
    for i=1:size(D n1,1),Li=D n1(i,:);
        for j=i:size(D n1,1)
         Lj=D n1(j,:);Li inter Lj=and(Li,Lj);
         Li union Lj=or(Li,Lj);
         if((Li inter Lj==Li)&(Li union Lj==Lj))
                D n=[D n; Li Lj];
        end
       end
    end
    D n1=D n;
end
DD=D n;DD(:,1)=[];DD(size(DD,1),:)=[]; D n=DD;
% Result display
disp(['|Theta|=n=',num2str(n)])
disp(['|D^Theta|=',num2str(size(D n,1))])
disp('Elem. of D^Theta are obtained by D n*u n')
disp(['with u n=[',num2str(u n),']'' and'])
D n=D n
```

Matlab source code for generating D^Θ