▶ TOMÁŠ KROUPA, TOMÁŠ VALLA, *Constructing many-valued logical functions with small influence of their variables.* [1]
Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, Prague, Czech Republic.
E-mail: kroupa@utia.cas.cz.
Faculty of Information Technology, Czech Technical University in Prague, Czech Republic.
E-mail: tomas.valla@fit.cvut.cz.

§1. Introduction. The Boolean functions with small influence of their inputs are used in the collective coin flipping algorithms [2]. In this contribution we replace the random bit generator with a random generator over a finite set and we show the existence of finitely-valued Łukasiewicz formulas with small influence of their variables.

§2. Łukasiewicz logic. We repeat basic definitions and results concerning finite-valued Łukasiewicz logic and its Lindenbaum algebra [3, 1]. We consider only finitely-many propositional variables $A_1, \dots, A_n$. Formulas $\varphi, \psi, \dots$ are then constructed from these variables and the truth-constant $\bar{0}$ using the following basic connectives: negation $\neg$ and strong disjunction $\oplus$. For any $k \in \mathbb{N}$ the semantics for connectives of $(k+1)$-valued Łukasiewicz logic is given by the corresponding operations of the *finite MV-chain,* which is just the set of rational numbers $Ł_k = \left\{0, \frac{1}{k}, \dots, \frac{k-1}{k}, 1\right\}$ endowed with constant zero 0 and the operations of negation $\neg$ and strong disjunction $\oplus$ defined as $\neg a = 1 - a$ and $a \oplus b = \min\{1, a+b\}$, respectively, for each $a, b \in Ł_k$. The structure $\langle Ł_k, \oplus, \neg, 0\rangle$ then becomes an *MV-algebra.* The operations $\odot$, $\wedge$, $\vee$, $\rightarrow$ are introduced in the standard way.

In the sequel we consider the expansion of the $(k+1)$-valued Łukasiewicz logic with the truth constants from the chain $Ł_k$. The language of $(k+1)$*-valued Łukasiewicz logic with truth constants* results from the language of $(k+1)$-valued Łukasiewicz logic by adding the truth constant $\bar{r}$ for each $r \in Ł_k$. Every truth constant is a formula. Formulas are built from propositional variables $A_1, \dots, A_n$ and truth constants using the connectives $\oplus$ and $\neg$ as well as other defined connectives of Łukasiewicz logic. Let F_n^k be the Lindenbaum algebra of $(k+1)$-valued Łukasiewicz logic with truth constants over n variables. By [1] we may identify F_n^k with the product $Ł_k^{(Ł_k^n)}$ whose elements are all the functions $f\colon Ł_k^n \to Ł_k$.

§3. Influence of Boolean variables. Boolean functions have a natural interpretation in game theory. A Boolean function is called a *simple game*. Each variable is controlled by a unique *player* and setting this variable to 1 or 0 expresses the yes/no voting scheme. The value of the Boolean function then represents an overall outcome of the voting. Observe that in our notation an n-variable Boolean function $f : Ł_1^n \to Ł_1$ is an element of F_n^1. The problem of measuring influence of a given propositional variable on the values of $f \in F_n^1$ was studied in coalitional game theory [6] and in the field of fault-tolerant computations [2].

The following notations will be used throughout the paper. Let $f \in F_n^k$ and $i \in \{1, \dots, n\}$. For every $y = (y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n) \in Ł_k^{n-1}$, we denote by f_y^{-i} the function $Ł_k \to Ł_k$ defined by $f_y^{-i}(x) = f(y_1, \dots, y_{i-1}, x, y_{i+1}, \dots, y_n)$, where $x \in Ł_k$.

The influence $\beta_i(f)$ of variable x_i on a monotone Boolean function $f \in F_n^1$ is defined

[1]Tomáš Kroupa was supported by the grant GAČR n. 13-20012S. The work of Tomáš Valla was supported by the grant GA P402/12/1309 of the Czech Science Foundation.

as the probability that f_y^{-i} remains non-constant when $y \in \{0,1\}^{n-1}$ is selected at random: $\beta_i(f) = \sum_{y\in\{0,1\}^{n-1}} \frac{f_y^{-i}(1)-f_y^{-i}(0)}{2^{n-1}}$. The number $\beta_i(f)$ is also called the *Banzhaf index* of player i in a coalition game f. We also define the Banzhaf index $\beta(f)$ of f as $\beta(f) = \max\{\beta_1(f), \ldots, \beta_n(f)\}$. The Banzhaf index measures the influence of players.

A natural motivation for investigating the players' influence comes from the collective random bit generators. Suppose there are n computers equipped with random generators. The task is to generate one random bit identical for all the machines. Simultaneously, each machine produces a uniform random bit and announces it to other machines. Each of them then has to perform a computation based on these inputs to produce the identical uniform random bit. The question is to which extent is a given random generator resistant towards possible third party attacks and corruption of one machine. The goal of the design of Boolean functions with low variables' influence is to minimize the chance of the attacker to manipulate the result.

Consider the Boolean function $f(x_1, \ldots, x_n) = x_k$ called the *dictatorship* of player k. The influence of dictator k is 1 and 0 for other players: $\beta(f) = 1$. The Boolean *majority function* $m \in F_n^1$ is defined as follows. Let n be odd and let $m(x_1, \ldots, x_n) = 1$ if there is a set $S \subseteq \{1, \ldots . n\}$, $|S| > n/2$, such that $x_i = 1$ for $i \in S$, and 0 otherwise. It follows that $\beta(m) = \Theta(1/\sqrt{n})$. On the other hand, it was shown [5, 4] that for any Boolean function the average influence of a variable is at least $\Omega(1/n)$. Surprisingly, there exists a Boolean function L performing better than the majority functions. In the next theorem we identify the vertices of $\{0,1\}^n$ with the subsets of $\{1, \ldots, n\}$.

THEOREM 1 (Ben-Or and Linial [2]). *There exists a construction of the function $L \in F_n^1$ such that $|L^{-1}(0)| = |L^{-1}(1)| = 2^{n-1}$ and $\beta(L) = O\left(\frac{\log n}{n}\right)$.*

The rough idea how the function L is constructed is as follows. Let b be the unique solution of the equation $(2^b - 1)^{1/b} = 2^{1-1/n}$. Decompose the set $\{1, \ldots, n\}$ into n/b blocks of size b and consider the set J of those subsets of $\{1, \ldots, n\}$ which contain no block. Let L be defined such that $L(A) = 0$ if $A \in J$ and $L(A) = 1$ otherwise.

§4. Influence of variables in many-valued logics. We will propose a natural generalization of Boolean Banzhaf index. Let $f \in F_n^k$, $k \geq 1$, and $i \in \{1, \ldots, n\}$. The *influence of variable x_i on f* is $\gamma_i(f) = (k+1)^{1-n} \cdot \sum\limits_{y\in Ł_k^{n-1}} \left(\max\limits_{x\in Ł_k} f_y^{-i}(x) - \min\limits_{x\in Ł_k} f_y^{-i}(x) \right)$. It can be shown that $\gamma_i(f)$ is a faithful generalization of the Banzhaf index. A natural next step is to design functions with low variable influence in the setting of the finitely-valued Łukasiewicz logic with truth constants. Our setting is the case of random generators producing a number from a finite set. Note that this setting naturally allows designs that may ask the input generators repeatedly.

4.1. 2^k-valued logic. Let us consider the set $Ł_{h-1}$ for $h = 2^k$ for some nonnegative integer k. In the sequel, we will naturally identify the elements $S_h = \{0, 1, \ldots, h-1\}$ with those in $Ł_{h-1}$. Note that we may encode each $x \in \{0, 1, \ldots, h-1\}$ by a k-element Boolean vector $(x^1, x^2, \ldots, x^k)$ representing the binary number x, with x^1 being the highest bit and x^k the lowest. Under the identification of S_h with $Ł_{h-1}$, we may analogously use the Banzhaf index $\gamma_i'(f) = h^{1-n} \sum_{y\in S_h^{n-1}} (\max_{x\in S_h} f_y^{-i}(x) - \min_{x\in S_h} f_y^{-i}(x))$. Let us define the function $f\colon S_h^n \to S_h$ as

$$f(x_1, \ldots, x_n) = \left(L(x_1^1, x_2^1, \ldots, x_n^1), L(x_1^2, \ldots, x_n^2), \ldots, L(x_1^k, \ldots, x_n^k) \right), \tag{1}$$

where the value $f(x_1, \ldots, x_n)$ is the binary representation of a number in S_h.

We shall prove that f has a small variable influence.

Theorem 2. *For $i = 1, \dots, n$, $\gamma'_i(f) = O\left(h\frac{\log n}{n}\right)$.*

Proof. The Banzhaf index of the function $L' = L(x_1^1, \dots, x_n^1)$ is $\beta_i(L') = O(\log n/n)$ by [2]. Observe that in the resulting vector of f the value of the highest bit $L(x_1^1, x_2^1, \dots, x_n^1)$ has the same effect on the size of the output value as the sum of all other lower bit orders, and the same holds for the influence of each lower bit. We may thus bound

$$\gamma'_i(f) \leq (2^{k-1} + 2^{k-2} + \dots + 1) \cdot \beta(L') \leq 2^k \cdot \beta(L') = O\left(h\frac{\log n}{n}\right).$$

$\dashv$

4.2. General many-valued logic. Let us now consider the set S_h with $h > 2$ and let ℓ be the smallest integer such that $2^\ell \geq h$. Let us denote by G_2 a random generator producing one uniform random bit. We construct the generator $G_{2\to h}$ that uses G_2 as the input and produces a value from S_h with uniform distribution:

1. Produce a number N by reading random bits $b_1, \dots, b_\ell$ from the G_2.
2. If $N < h$ then return the number N.
3. Otherwise, repeat the whole process.

Lemma 3. *The generator $G_{2\to h}$ produces a result with a uniform distribution over S_h. The expected number of random bits read from G_2 is $\ell 2^\ell/h = \Theta(\log h)$.*

Proof. W.l.o.g., let p denote the probability that 0 is on the output of $G_{2\to h}$. Then

$$p = \frac{1}{2^\ell} + \frac{2^\ell - h}{2^\ell}p$$

since with probability $1/2^\ell$ the result is produced immediately and with probability $(2^\ell - h)/2^\ell$ the process is repeated independently on the previous round. Solving the equation yields $p = 1/h$.

Denote by E the expected number of random bits needed to produce the result. Similar equation $E = \ell + \frac{2^\ell - h}{2^\ell}E$ holds as ℓ bits are used always and with probability $(2^\ell - h)/2^\ell$ the whole memoryless process repeats. The solution gives $E = \ell 2^\ell/h$. Finally, note that $\ell \approx \log_2 h$ and $h \leq 2^\ell < 2h$. $\dashv$

Let us denote G_h a random generator producing uniform S_h-valued output. Using analogous technique, we design a generator $G_{h\to 2}$ which reads uniformly distributed random S_h-valued input and produces a uniformly distributed random bit. The process is as follows. If h is even, one random input is read and its parity is returned. If h is odd, the following procedure is used.

1. Read one random number $N \in S_h$ from G_h.
2. If $N \neq h - 1$, return the parity of N.
3. Otherwise, repeat the whole process.

Lemma 4. *The generator $G_{h\to 2}$ produces a uniform random bit. If h is even, $G_{h\to 2}$ reads 1 random input value. If h is odd, the expected number of random values read is $h/(h-1)$.*

Proof. Let p denote the probability that $G_{h\to 2}$ produces, w.l.o.g., 0. Then $p = \frac{(h-1)/2}{h} + \frac{1}{h}p$ as with probability $(h-1)/(2h)$ the result is produced immediately and with probability $1/h$ the process is repeated independently on the previous round. We get $p = 1/2$.

Denote by E the expected number of random S_h-values needed to produce the result. We have that $E = 1 + \frac{1}{h}E$ holds as 1 value is used always and with probability $1/h$ the memoryless process repeats, which gives the solution $E = h/(h-1)$. $\dashv$

4.3. Function with low influence of variables. We describe the generator G which is given n uniform S_h-valued random generators and produces the S_h-valued output with a low influence of the input generators. Let us denote the input random S_h-value generators by $g_1, \ldots, g_n$.

The generator is constructed as $G = G_{2\to h}\Big(L\big(G_{h\to 2}(g_1), G_{h\to 2}(g_2), \ldots, G_{h\to 2}(g_n)\big)\Big)$, where L is the function from Theorem 1. In another words, the generator $G_{2\to h}$ repeatedly asks for Boolean bits from the function L, which in turn asks the n generators $G_{h\to 2}$ connected to the inputs of L, which in turn ask the input generators $g_1, \ldots, g_n$.

Lemma 5. *The probability that G produces $v \in S_h$ is $1/h$.*

Proof. By Lemma 4, each generator $G_{h\to 2}(g_i)$ produces uniform random bit. By Theorem 1, the function L then produces a uniform random bit. Finally, the generator $G_{2\to h}$ produces uniform S_h-valued output. $\dashv$

Let ℓ be the smallest integer such that $2^\ell \geq h$.

Lemma 6. *For the generator G, the expectation of the total number of random values produced by the generators $g_1, \ldots, g_n$ in order to obtain one output of G is $n{\cdot}\ell{\cdot}2^\ell/(h-1)$.*

Proof. Recall that by Lemma 4 the expectation of random values needed by the generator $G_{h\to 2}$ to produce single output is $h/(h-1)$. As the generators $g_1, \ldots, g_n$ are independent, linearity of expectation yields that to produce one output bit of the function L, $nh/(h-1)$ input values are needed in the expectation. Since each execution of L is independent on the previous runs, by Lemma 3 we obtain the total expectation $(\ell 2^\ell/h)nh/(h-1) = n\ell 2^\ell/(h-1)$. $\dashv$

Lemma 7. *For each $i = 1, \ldots, n$, we have $\gamma'_i(G) = O((2^\ell \log n)/n)$.*

Proof. Observe that the function $L' = L(G_{h\to 2}(g_1), \ldots, G_{h\to 2}(g_n))$ behaves exactly as the function f defined by (1). Theorem 2 yields $\gamma'_i(L') = O((2^\ell \log n)/n$. During the step 1 of the generator $G_{2\to h}$, the function L is called ℓ times, which produces a number N in the range $0, \ldots, 2^\ell - 1$. Less than one half of the possible values of N is rejected and step 1 is repeated independently on the result of the previous iteration. $\dashv$

We may identify G with a unique function $\widehat{G} : \text{Ł}^n_{h-1} \to \text{Ł}_{h-1}$. Note that $\gamma_i(\widehat{G}) = \gamma'_i(G)/(h-1)$. Since $\ell \approx \log_2 h$, using Lemma 6 and Lemma 7 we may conclude with the following corollary.

Corollary 8. *The generator $\widehat{G}$ needs in total $\Theta(n \log h)$ input random values in expectation, and $\max\limits_{i=1,\ldots,n} \gamma_i(\widehat{G}) = O(\log n/n)$.*

[1] S. Aguzzoli, S. Bova, and B. Gerla, *Free algebras and functional representation for fuzzy logics*, ***Handbook of Mathematical Fuzzy Logic - Volume 2***, (P. Cintula, P. Hájek, and C. Noguera, editors), College Publications, London, 2011, pp. 713–791.

[2] M. Ben-Or and N. Linial, *Collective coin flipping*, ***Randomness and Computation*** (S. Micali, editor), Academic Press Inc., 1989, pp. 91–115.

[3] P. Hájek, ***Metamathematics of fuzzy logic***, volume 4 of Trends in Logic—Studia Logica Library, Kluwer Academic Publishers, 1998.

[4] L. Harper, *Optimal Numberings and Isoperimetric Problems of Graphs*, ***Journal of Combinatorial Theory*** ser. A (1966), pp. 385–393.

[5] S. Hart, *A note on the edges of the n-cube*, ***Discrete Mathematics*** 14 (1976), pp. 157–163.

[6] G. Owen, ***Game theory***, Academic Press Inc., 1995.