

Visual reasoning by generalized interval-values and interval temporal logic

Benedek Nagy^{1,2}

*Department of Computer Science
Faculty of Informatics and Information Sciences
University of Debrecen
Debrecen, Hungary*

Sándor Vályi³

*Department of Health Informatics
Faculty of Health
University of Debrecen
Nyíregyháza, Hungary*

Abstract

Interval-valued computation is an unconventional computing paradigm. It is an idealisation of classical 16-, 32-, 64- etc. bit based computations. It represents data as specific subsets of the unit interval – in this sense this paradigm is classified into the continuous space machine paradigm near to optical computing. In this paper we show the visual reasoning power of interval-valued computations, namely, we demonstrate that the decision process of quantified propositional formulae is fully representable in a natural visual form. Further, we give a temporal-logical interpretation of interval-valued computations.

Keywords: new computing paradigms, visual reasoning, interval temporal logic

1 Introduction

In the last fifteen years arised a new direction of computing which develops ideas for computing devices motivated by nature. It includes DNA computing, quantum computing and relativistic computers, among others.

In [9] and [11], another new computing paradigm were introduced, the so-called interval-valued computation system. In this paradigm, data is represented by spe-

¹ This work has been supported by the grant of the Hungarian National Foundation for Scientific Research OTKA T049409, by a grant of the Hungarian Ministry of Education and by the Öveges programme of the Agency for Research Fund Management and Research Exploitation (KPI) and National Office for Research

and Technology



² Email: nbenedek@inf.unideb.hu

³ Email: valysi@de-efk.hu

cific subsets of the unit interval, namely, by finite unions of disjoint subintervals. This data representation corresponds to the notion of generalized intervals ([6], [2]). In these papers some logics of temporal relations between such generalized intervals (interval-values, in this paper) are analyzed. In [9] some other operators were proposed to construct an interval-valued computing system and also the SAT problem was solved by a linear interval-valued computation. In [11] and [10] it was proved that a restricted class of interval-valued computations is adequate for PSPACE, that is, the class of languages decidable by this class of interval-valued computations coincides with PSPACE.

Reasoning by diagrams and intervals is a long-standing area of investigation into visual representations of mathematical reasoning. Very selectively, Venn- and Euler-diagrams are to mention here and also graphical versions of interval temporal logic. A basic method for visualizing Boolean algebraic calculations is the method of Venn diagrams. It is suitable to formulæ built from two or three propositional variables. There are good ideas to generalize Venn diagrams to a higher number of variables. ([4], [12], [1], [5], [3], [8]) Venn-diagrams are suitable to visually represent propositional logical laws. Of course, our interval-values is also able to represent propositional reasoning in a nice and natural visual form, because the interval-values form a Boolean algebra in which every finite Boolean algebra is visually representable. Moreover this visual representation is also suitable to help visually to follow the decision process of the validity of quantified propositional formulae. This problem is PSPACE-complete. This complexity class includes such typical problems that solution of two-player games like chess or go. In this paper we demonstrate the visual expressibility of the decision process of validity of quantified propositional formulae. We also formalise an interval temporal logic equipped by some modalities concerning the freshly introduced operators on interval-values. A decidability and a complexity result will also be given.

2 The idea of interval-valued computations

In [9], Nagy proposed a new discrete time / continuous space computational model, the so-called interval-valued computing. It involves another type of idealization than Turing machines – the density of the memory can be raised unlimitedly instead of its length. It is a natural model that can formulate computations of computers with higher and higher bit number in a byte in a unified framework.

As long as the paradigm keeps using only finite unions of intervals, the system fits within the bounds of classical Neumann-Church-Turing type computations.

The computation works on specific subsets of the interval $[0, 1)$, more specifically, on finite unions of $]$ -type subintervals. In a nutshell, interval-valued computations start with $[0, \frac{1}{2})$ and continue with a finite sequence of operator applications. It works sequentially in a deterministic manner.

The allowed operations are motivated by the operations of the traditional computers on bit sequences: Boolean operations, shift operations and an extra operator, the product. The role of the introduced product is connecting interval-values on different 'resolution levels'. Essentially, it has the same function like magnification operators in optical computing ([13]) which is another continuous space computing

paradigm where data is represented by 2-dimensional complex-valued images.

In the interval-valued computing system, an important restriction is eliminated, i.e. there is no permanent limit on the number of bits in a data unit (byte); we have to suppose only that the number is always finite. Of course, in the case of a given computation an upper bound (the bit height of the computation sequence) always exists, and it gives the maximum number of bits the system needs for that computation process. Hence our model still fits into the framework of the Church-Turing paradigm, but it faces different complexity bounds than the classical Turing model. Although the computation in this model is sequential, the inner parallelism is extended. One can consider the system without restriction on the size of the information coded in an information unit (interval-value). It allows to increase the size of the alphabet unlimitedly in a computation. In this article we employ this inner parallelism to extend the visual expressiveness of calculations with interval-values. Complex manipulations on the interval-bytes can be shown, acting uniformly to the whole stored data – the interval-value. This makes possible, for instance, the visual representation of decision process of quantified propositional formulae.

3 Interval-values and operators

As we mentioned, interval-values are finite unions of disjoint left-closed, right-open subintervals of the unit interval $[0, 1]$.

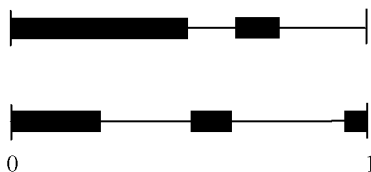


Fig. 1. Examples of interval-values

We recall the formal definitions from [11].

Definition 3.1 The set \mathbb{V} of *interval-values* coincides with the set of finite unions of $]$ -type subintervals of $[0, 1]$. The set \mathbb{V}_0 of *specific interval-values* coincides with $\left\{ \bigcup_{i=1}^k \left[\frac{l_i}{2^m}, \frac{1+l_i}{2^m} \right) : m \in \mathbb{N}, k \leq 2^m, 0 \leq l_1 < \dots < l_k < 2^m \right\}$.

We note that the set of finite unions includes the empty set ($k = 0$), that is, \emptyset is also an allowed interval-value.

Similarly to traditional computers working on bytes, we allow bitwise Boolean operations. If we consider interval-values as subsets of $[0,1]$ then the corresponding operations coincide with the set-theoretical operations of complementation (\overline{A}), union ($A \cup B$) and intersection ($A \cap B$). \mathbb{V} forms an infinite Boolean set algebra with these operations. \mathbb{V}_0 is an infinite subalgebra of the last algebra. We note that other usual Boolean operators, as *xor* or \rightarrow are definable in the usual way.

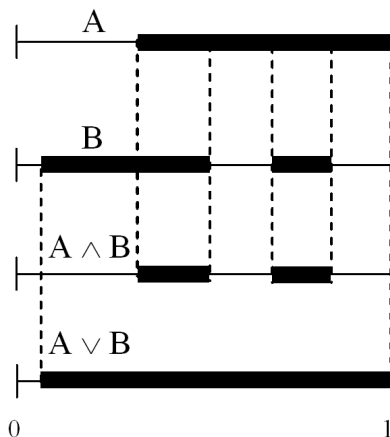
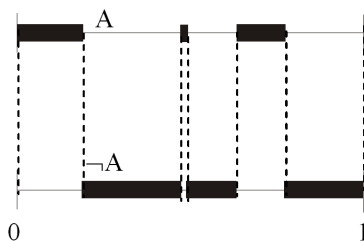

 Fig. 2. Examples for \cap and \cup


Fig. 3. Example for complement

Assisting formulation of the remaining operations, a function $Flength : \mathbb{V} \rightarrow \mathbb{R}$ is going to be defined. Intuitively, it determines the length of the first “component” of the input interval-value, that is, the first (from left) maximal subinterval of the unit interval included in the given interval-value.

Definition 3.2 Let the function $Flength : \mathbb{V} \rightarrow \mathbb{R}$ be defined as follows. If there exist $a, b \in [0, 1]$ satisfying $[a, b] \subseteq A$, $[0, a] \cap A = \emptyset$ and $[a, b'] \not\subseteq A$ for all $b' \in (b, 1]$, then $Flength(A) = b - a$, otherwise $Flength(A) = 0$.

$Flength$ helps us to define the binary shift operators on \mathbb{V} . The *left-shift* operator will shift the first interval-value to the left by the first-length of the second operand and remove the part which is shifted out of the interval $[0, 1)$. As opposed to this, the *right-shift* operator is defined in a circular way, i.e. the parts shifted above 1 will appear at the lower end of $[0, 1)$. In this definition we write interval-values in their “characteristic function” notation instead of subset notation.

Definition 3.3 The binary operators $Lshift$ and $Rshift$ on \mathbb{V} are defined in the following way. If $x \in [0, 1]$ and $A, B \in \mathbb{V}$ then

$$Lshift(A, B)(x) = \begin{cases} A(x + Flength(B)), & \text{if } 0 \leq x + Flength(B) \leq 1, \\ 0 & \text{in other cases.} \end{cases}$$

$$Rshift(A, B)(x) = \begin{cases} A(\text{frac}(x - \text{Flength}(B))), & \text{if } x < 1, \\ 0 & \text{if } x = 1. \end{cases}$$

Here the function frac gives the fractional part of a real number, i.e., $\text{frac}(x) = x - \lfloor x \rfloor$, where $\lfloor x \rfloor$ is the greatest integer which is not greater than x .

In Figure 4 some examples can be seen for both operations $Rshift$ and $Lshift$. The second operands are shown in grey but they are not the real parts of the resulted interval-values. Now we define the so-called *fractalian product* on interval-values.

Definition 3.4 Let A and B be interval-values and $x \in [0, 1)$. Then the fractalian product $A * B$ includes x if and only if $B(x) = 1$ and $A\left(\frac{x - \underline{B}_x}{\overline{B}_x - \underline{B}_x}\right) = 1$, where \underline{B}_x denotes the lower end-point of the B -component including x and \overline{B}_x denotes the upper end-point of this component, that is, $[\underline{B}_x, \overline{B}_x)$ is the maximal subinterval of B containing x .

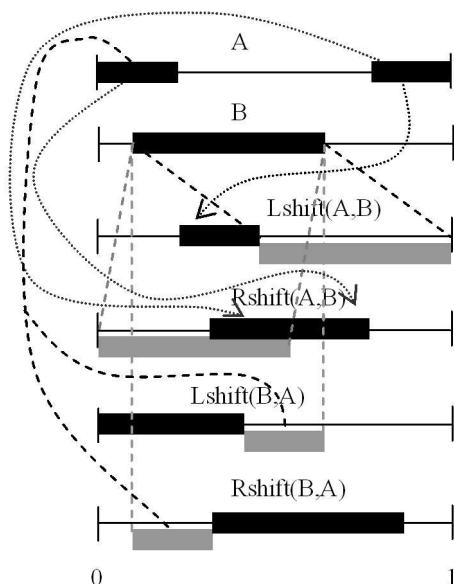


Fig. 4. Examples of shift operators with interval-values

We can explain this in a more descriptive manner. If A contains exactly k interval components with ends $a_{i,1}, a_{i,2}$ ($1 \leq i \leq k$) and B contains exactly l components with ends $b_{i,1}, b_{i,2}$ ($1 \leq i \leq l$), then we determine the value of $C = A * B$ as follows: we set the number of components of C to be $k \cdot l$. For this process we can use double indices for the components of C . The starting- and end points of the ij -th component are $a_{i1} + b_{j1}(a_{i2} - a_{i1})$ and $a_{i1} + b_{j2}(a_{i2} - a_{i1})$, respectively.

The idea and the role of this operation is similar to that of unlimited shrinking of 2-dimensional images in [13]. It will be used to connect interval-values of different resolution. As we can observe in Figure 5, as well, the fractalian product of two

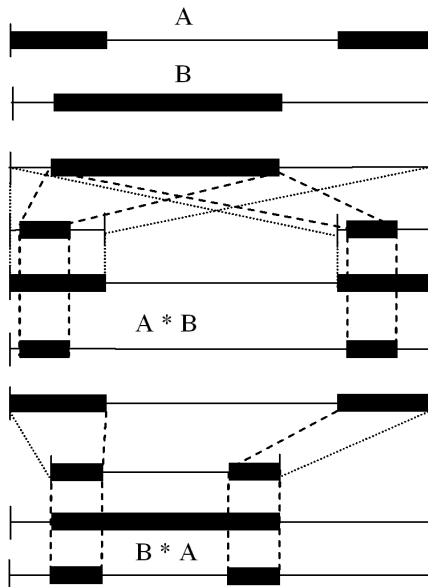


Fig. 5. Examples for product of interval-values

interval-values is the result of shrinking the first operand to each component of the second one.

4 A representation of n independent truth values

In this section we give a natural interval-valued computational representation of all variations of n independent truth values which is only a visual rephrase of the well-known truth tables and will be useful not only in checking whether a given propositional formula is a logical law or not but also in the interval-valued computations deciding whether a given quantified propositional formula is true or not.

For the lack of space, we do not define formally the *interval-valued computations*, consult [10] or [11] for the formal details. For the aims of the present paper it is enough to know that it is a sequence of interval-values where each new member of the sequence is an operator application of one or two precedents in the same sequence and which is starting with $[0, \frac{1}{2})$. In this manner, *deciding a language L by an interval-valued computation* means constructing an algorithm that for any input problem instance responds an interval-valued computation sequence with the following property: the result of the interval-valued computation sequence created by the algorithm to an input word w is equal to the unit interval $[0, 1)$ if and only if w is in L .

We give a computation which constructs a quite natural interval-valued representation of n independent truth values. Let K_1 be $[0, \frac{1}{2})$. For all non-negative integers k , we define $K_{3k+2} = K_1 * K_{3k+1}$, $K_{3k+3} = RShift(K_{3k+2}, K_{3k+1})$ and $K_{3k+4} = K_{3k+2} \cup K_{3k+3}$.

Fact 4.1 *By an induction on k one can establish that*

$$K_{3k+1} = \bigcup_{l=0}^{2^{k-1}-1} \left[\frac{2l}{2^k}, \frac{2l+1}{2^k} \right).$$

By the previous fact this computation sequence produces suitable interval-values since it satisfies the following.

Fact 4.2 *For any $(x_1, \dots, x_n) \in \{0, 1\}^n$ there exists $r \in [0, 1)$ satisfying that $(x_1, \dots, x_n) = (r \in K_1, r \in K_4, \dots, r \in K_{3n-1})$.*

All of our interval-valued computations (at least the ones deciding validity of quantified propositional formulae) will start with the construction of K_1, \dots, K_{3n-1} , if n is the number of propositional variables of the input formula. The first 4 interval-values in Figure 6 and 7 are K_1, K_4, K_7, K_{11} , they represent 4 independent truth values of x_1, x_2, x_3, x_4 .

Of course, using these interval-values representing all possible variations of the truth/falsity of the propositional variables x_1, \dots, x_n , one can easily decide the validity of propositional formulae, by executing the Boolean operations on the interval-values on the desired order. In this way any propositional formula $\varphi(x_1, \dots, x_n)$ gets its interval-truth-value by an appropriate interval-valued computation $C(\varphi)$. Not specifying $C(\varphi)$ more formally, we can observe that for any propositional formula ϕ built from propositional variables x_1, \dots, x_n and for any $r \in [0, 1)$ the following holds: $r \in C(\varphi) \Leftrightarrow \varphi$ is satisfied by the truth valuation $(x_1 : (r \in K_1), x_2 : (r \in K_4), \dots, x_n : (r \in K_{3n-1}))$. One may check this simple-to-prove statement formally in [10] or in [11]. We do not analyse this part in detail. The fifth lines of Figure 6 and 7 represent $C(\varphi)$ for the two given formulae, respectively.

5 Visual solution of a PSPACE-complete problem

We employ the visual reasoning power of interval-valued computations for a more complex task. In [10] and [11] we proved that a PSPACE-complete problem, namely, the problem whether a quantified propositional formula is true, is also decidable by a linear interval-valued computation. Also for this case, we show that the sequence of interval-values produced by this computation represents visually the full information needed to understand the solution of the given case of the problem.

We do not specify formally the needed computations. We state only that the presented interval-values can be computed by interval-valued computations. For the formal details the reader may consult the above mentioned papers. Instead of this, we concentrate on that how these interval-values visually encode the information needed to follow the decision process for validity of quantified propositional formulae.

A quantified propositional formula –without loss of generality – is of form $\forall t_1 \exists t_2 \dots Q_n \varphi$ where Q_i is \forall for odd i and \exists for even i . It is called true or valid iff $\forall t_1 \in \{0, 1\} \exists t_2 \in \{0, 1\} \dots Q_n t_n \in \{0, 1\} \varphi(x_1 : t_1, \dots, x_n : t_n)$ holds.

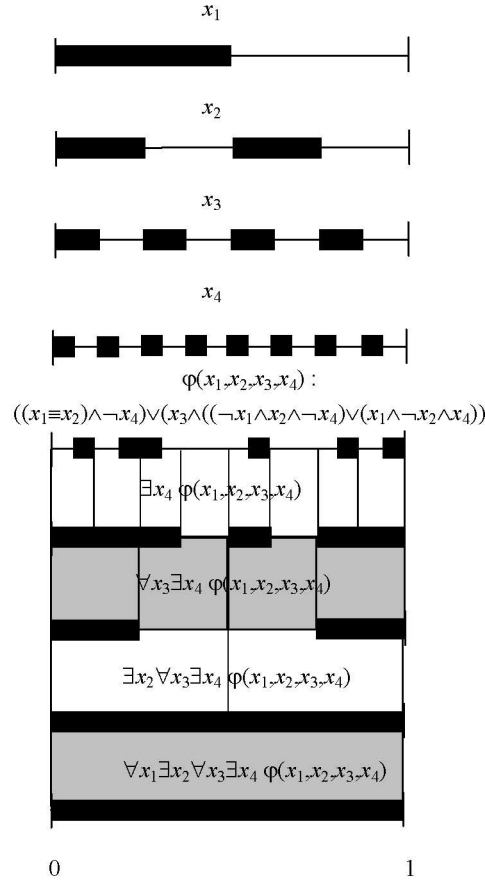


Fig. 6. This quantified formula is true

In [10] and [11], the interval-valued decision algorithm constructs interval-values $C_0(\varphi)(= C(\varphi)), C_1(\varphi), \dots, C_n(\varphi)$ with the following properties.

- $C_i(\varphi) = \{r \in [0, 1] :$
 $Q_{n-i+1}t_{n-i+1} \dots Q_n t_n$
 $\varphi(x_1 : (r \in K_1), \dots, x_{n-i} : (r \in K_{3(n-i)-1}), x_{n-i+1} : t_{n-i+1}, \dots, x_n : t_n)\},$
- $C_{i+1}(\varphi)$ can be constructed from $C_i(\varphi)$ and the interval-value corresponding to x_{n+1-i} , that is, from $K_{3(n+1-i)-1}$.

The steps to determine $C_{i+1}(\varphi)$ needs alternating \forall - and \exists -transformations. A \forall -step means checking the interval AND its corresponding neighbor in $C_i(\varphi)$, while an \exists -step amounts to checking the interval OR its neighbour. \forall -step visually means simply a check if the appropriate neighbour of the examined subinterval is also in $C_i(\varphi)$ while an \exists -step checks if the appropriate neighbour of the examined subinterval OR the subinterval itself is in $C_i(\varphi)$.

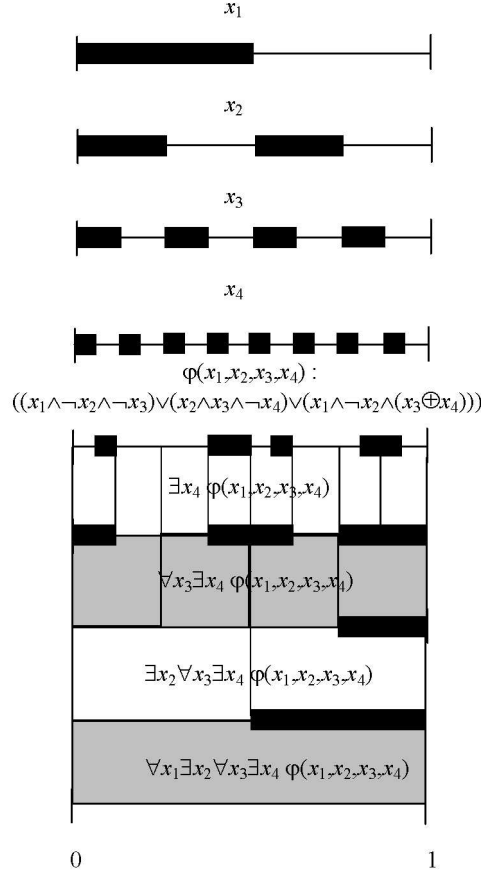


Fig. 7. This quantified formula isn't true

In Figure 6 and 7 one can follow two interval-valued computations deciding whether a given quantified propositional formula is true. The 1st-4th lines show the 4 independent truth values, the 5th line shows the result of the evaluation of the Boolean operators and 6th-9th lines include the result of adding one quantifier per line to the formulae.

6 Interval-valued computations and interval temporal logic

If we think of $[0,1)$ as a time flow then we may investigate its temporal logic. It trivially coincides with the temporal logic of $(\mathbb{R}^{+0}, <)$ where \mathbb{R}^{+0} is the set of nonnegative reals. It may be an interesting question, what happens if we add our operators to the temporal logic over $[0,1)$ as binary modal operators. For an interesting class of such modal-temporal formulae we can provide a decidability theorem.

Definition 6.1 Let us call the members of the following set of formulae interval-valued modal-temporal formula. It is the minimal set of strings satisfying the following:

- a, b, \dots are (atomic) formulae,
- $FirstHalf$ is a formulae,
- if φ, θ are formulae, then $(\varphi \wedge \theta)$, $(\varphi \vee \theta)$ and $\neg\varphi$ are formulae, too,
- if φ, θ are formulae, $\Box\rightarrow\varphi$ and $\leftarrow\Box\varphi$ are formulae, too,
- if φ, θ are formulae then $R(\varphi, \theta)$, $L(\varphi, \theta)$ and $P(\varphi, \theta)$ are formulae, too.

Definition 6.2 An interval-valuation v is no other than a function assigning to each member of $\{a, b, \dots\}$ an interval-value. Then for any interval-valued modal-temporal formula $\|\varphi\|_v$ is an interval-value of the interval-valued modal-temporal formula φ . The definition of this notion is the expected one. We just write three clauses of this definition.

- $\|FirstHalf\|_v = [0, \frac{1}{2})$,
- $\|\Box\rightarrow\varphi\|_v$ is $\{t \in [0, 1) : (t, 1) \subseteq \|\varphi\|_v\}$,
- $\|P(\varphi, \theta)\|_v = \|\varphi\|_v * \|\theta\|_v$.

A modal-temporal formula is said to be modal-temporal logical law if with every valuation v its interval-value is $[0,1)$.

Problem 6.3 *How to axiomatize this kind of modal-temporal logic? Is it decidable? If yes, what is its complexity?*

We have a partial answer to this question.

Claim 6.4 *The problem if a modal-temporal formula built up only from $FirstHalf$ but without other propositional variables is decidable by exponential time. If the usage of product operator is restricted to always take product with $FirstHalf$ then the arising problem is solvable in polynomial space, moreover, is PSPACE-complete.*

This answer can be extracted by Theorem 21 of [11]. To any formula one can find an appropriate interval-valued computation whose result have to be $[0,1)$ if and only if the formula is a law.

7 Conclusion

We have demonstrated the visual reasoning power of a freshly introduced unconventional computing system. Its expressiveness depends on data representation by interval-values which makes it possible by its topological properties.

It is worth thinking over what further problems can be naturally represented by generalized intervals. Possible candidates are problems about occurring events in temporal logic with a notion of compositionality. The product operator would provide transfer between different compositional levels - , embeddability of macro- and micro scales can be conceptualized. In this way, also visual analysis and visual representation of re-occurring, periodic hierarchical events – e.g. in biostatistics and health insurance – would be available.

Further generalization is possible to regions in higher dimensional spaces, mainly to \mathbb{R}^2 . In this way one should work out the connections of interval-valued computing to so-called optical computing where objects of computing are 2-dimensional images ([13]) through their visual applications.

References

- [1] Anderson, D., E., and F. L. Cleaver, *Venn-type diagrams for arguments of n terms*, J. Symb. Logic **30** (1965), 113–118.
- [2] Balbiani, P., et al., *Reasoning about generalized intervals*, in: F. Giunchiglia (ed), Artificial Intelligence: Methodology, Systems and Applications, Lecture Notes in Artificial Intelligence **1480** (1998), 50–61, Springer.
- [3] Chilakamarri, K., B., and R. E. Pippert, *Venn diagrams and planar graphs*, Geometriae Dedicata **62** (1996), 73–91.
- [4] Edwards, A., W., F., “Cogwheels of the Mind: The Story of Venn Diagrams,” John Hopkins University Press, 2004.
- [5] Henderson, D., W., *Venn diagrams for more than four classes*, Amer. Math. Monthly **70**(1963), 424–426.
- [6] Ligozat, G., *On generalized interval calculi*, AAAI-91 Proc. of the 9th National Conf. on Artif. Intelligence **1** (1991), 234–240.
- [7] Nagy, B., and G. Allwein, *Diagrams and Non-monotonicity in Puzzles*, in: Diagrammatic Representation and Inference, Proc. of Diagrams’2004, 3rd Int. Conf. on Theory and Application of Diagrams, Cambridge, England, (eds.:A. Blackwell, K. Marriott, A. Shimojima) Lect. Notes in Comp. Sci. LNAI **2980** (2004), 82–96.
- [8] Nagy, B., *Reasoning by intervals*, Proc. 4th Int. Conf. on Theory and Applications of Diagrams, Stanford(CA), USA, Lect. Notes in Comp. Sci. LNAI **4045** (2006), 145–147.
- [9] Nagy, B., *An Interval-valued Computing Device*, in: Computability in Europe 2005: New Computational Paradigms, (eds. S. B. Cooper, B. Löwe, L. Torenvliet), ILLC Publications X-2005-01, Amsterdam, 166–177.
- [10] Nagy, B., and S. Vályi, *Solving a PSPACE-complete problem by a linear interval-valued computation*, in: Proc. of Conf. Computability in Europe 2006: Logical Approaches to Computational Barriers, (eds. A. Beckmann, U. Berger, B. Loewe), Uni. of Swansea Report no. CSR-7-2006, Swansea, 216–225. <http://www.cs.swansea.ac.uk/reports/yr2006/CSR7-2006.pdf>
- [11] Nagy, B., and S. Vályi, *Interval-valued computations. Introduction and connections with PSPACE*, accepted for publication in **Theoretical Computer Science** in 2007.
- [12] Ruskey, F., and M. Weston, *A survey of Venn diagrams*, Electronic Journal of Combinatorics **12** 2005.
- [13] Woods, D., and T. Naughton, *An optical model of computation*, Theoretical Computer Science **334** (2005), 227–258.