Serdica J. Computing **1** (2007), 101–114

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

MIXED THEORIES

Slavian Radev

Abstract. In the present paper we investigate the life cycles of formalized theories that appear in decision making instruments and science. In few words mixed theories are build in the following steps: Initially a small collection of facts is the kernel of the theory. To express these facts we make a special formalized language. When the collection grows we add some inference rules and thus some axioms to compress the knowledge. The next step is to generalize these rules to all expressions in the formalized language. For these rules we introduce some conclusion procedure. In such a way we make small theories for restricted fields of the knowledge. The most important procedure is the mixing of these partial knowledge systems. In that step we glue the theories together and eliminate the contradictions. The last operation is the most complicated one and some simplifying procedures are proposed.

Scientific theories are investigated by many authors from different points of view [4], [5], [6], [8], [11]. There are some differences between authors' points of view as well as some differences between levels of the theory constructions. On the other hand, there are no good investigations of theories produced for decision-making. Most subjects before making a decision prepare (or use) a

ACM Computing Classification System (1998): F.4.1.
Key words: formal theories, expert systems, logical inference, argumentation systems

small theory that helps them in the decision calculation. The correctness of the decision depends on the postulates of the theory and on the method used in calculations. Sometimes these methods are logical or almost logical. That is why we introduce here a model of production of small theories as an introduction and base for the next chapters. Fortunately, this is not a universal scheme of decision-making and needs no more argumentation. Moreover, this is not a scheme for construction of scientific theories and needs no philosophical argumentation. Here we concentrate our investigations on argumentation methods, operations over arguments and their applications. The theories that appear are not so scientific and not always correct.

When we make a decision, we do not think about the argumentation. But later we have to prove that the decision is correct and prepare our arguments, proof methods and other argumentation (persuasion) attributes. In other words, we make theories. Simple decisions usually need small theories. When we want to generalize our decisions, to enlarge the scope of our theories, we have to be careful because of the conflicts that may appear. If we have only strict and complete information, or if the information is small and compact, then we apply neither theories nor proof mechanisms to use our knowledge. Only when the information, the knowledge, is too large we need some "mnemonic" instruments to order it and prepare it for use. In the changing world (as well as in the small mind) there is no sense in using a big general theory. More effective are small theories and changing inference mechanisms. Small theories are consistent or can easily made consistent. However, small theories are inconsistent together. Hence, the joint theories need more powerful mechanisms to be consistent. The practical theories may be useful independently of their consistency. Here we consider theories build for decisions, not for science, hence there is some difference between that schema and Popper's [11]. The steps we investigate may be applied dynamically together with the change of our knowledge hence the consistency may be guaranteed in practice only potentially, partially. The theory mixer, however, allows us to keep consistence in the parts in which we actually are.

1. How some theories came. We know a lot of facts about different objects. Naturally, a problem with our knowledge arises: How to organize all these facts in a system? Usually [12] different variations of the following procedure (sometimes not even mentioned) are proposed to systematize the knowledge:

- [**B**] Collect all we know in basic set(s) of our *beliefs*;
- [**L**] Make formal *language(s)* to express our knowledge;

- [**R**] Introduce *rules* of inference;
- [**G**] *Generalize* these rules;
- [**C**] Introduce a *consequence* mechanism(s) based on the generalized rules;
- [**M**] *Mix* the consequence mechanisms;
- [**E**] *Eliminate* the conflicts in the consequence procedure.

Consider how this procedure works (informally):

1.1. From B(ase) to L(anguage) $\mathbf{B} \to \mathbf{L}$. Let $\mathbf{B} = \{B_1, \dots, B_n\}$ be our knowledge base — all the facts (beliefs, observations, experiments) we know. The first step in the transformation of our knowledge into a theory is to construct a formal language $\mathbf{L}$ (Figure 2) and a translation $f : \mathbf{B} \to \mathbf{L}$ from our knowledge into language. In such a way we obtain the set $\mathbf{c} = \{C_1, \dots, C_n\}$ of formal expressions corresponding to $\mathbf{B}$ and possibly some new expressions that have to correspond to some potential facts. The language is a very strong abstraction because some information is lost: e. g., "When Charlie walks he does that and that, while when Flip walks he does that and that" – is a mimic expression but not linguistic expression. Without lost of information we can write (at least in the formalized language here) $\{B_1, \dots, B_n\}$ instead of $\{C_1, \dots, C_n\}$, but we have to remember that these are the *images* of the basic facts. At this step we obtain a *pretheory* $\langle \mathbf{B}, \mathbf{L}, f \rangle$ — a collection of facts with a formal language to describe them. This step corresponds to the data base level of our knowledge and of informatics practice.

1.2. From B(ase) and L(anguage) to R(ules) $\mathbf{B}, \mathbf{L} \to \mathbf{R}$. When we know nothing about the structure of the knowledge we are free to make hypotheses (speculations) and to construct elementary rules about the reasons in the world that we know about. For instance, we may suppose that from the first two C_1 and C_2 the third C_3 follows or, which is the same, "C_1 and C_2 must be true, because C_3 has to be true" (if C_1, C_2, C_3 are simultaneously true the reasoning is in both directions). And introduce a rule $\langle \{C_1, C_2\}, \{C_3\} \rangle$. The set $\mathbf{R} = \{r_1, \dots, r_k\}$ of such rules is the second stage of the (mnemonic) theory construction. At that stage we obtain an *elementary theory* $\mathbf{ET} = \langle \mathbf{B}, \mathbf{L}, \mathbf{R} \rangle$. The inference mechanism is simple — every $\mathbf{R}$-ordered subtree of $\mathbf{ET}$ is a proof. Elementary theories are basic objects in the Argumentation Systems, Information Systems, Expert Systems, Truth Maintenance Systems [1], [2], [5], [7], [9], [15], [16], [17].

1.3. To G(eneralization) $\mathbf{B}, \mathbf{L}, \mathbf{R} \to \mathbf{G}$. The set of elementary rules usually is quite large, hence we construct metarules $R_1, \dots, R_m$ *generalizing* some classes of the rules from **R** using the "typical example" techniques. For simplicity, we shall denote the set of metarules by the same letter **R**. It is clear that firstly (simultaneously) we construct the language **L** (introduce the linguistic instruments) in such a way as to prepare the ground for these *generalizations*.

Typical example. We shall consider theories based on generalizations of *typical examples*: Typical examples are the expressions of the language. From a typical example $A(a)$ we make an *abstraction* $A(x)$ changing the object's name a to the variable x and introducing the rule $\langle A(x), A[x/B]\rangle$ for every expression B admissible for x in the language. Typical examples of this style of generalization are: the textbook sentence "This is an apple." in which every new learned (shown) noun is admissible for substitution instead of "apple"; the logical construction $A \vee B$, in which every expression is admissible for substitution instead of A or B. The style of many articles is based on typical examples and seems fruitful. The basic languages also have these typical examples as: York — typical town, taler (dollar) — typical coin, Caesar — typical dictator (emperor), ...

Generalization. A *generalization* of an expression $A(a)$ is a triple $\langle A(a), x, L\rangle$ in which x is a variable and L is the language of the synonymous words *admissible* for a in A. It is easy to see in the previous examples that in the natural language the typical examples have different languages of words admissible for substitution. The generalization rules are the most popular inference instruments in logic, language [15], mathematics, informatics, ...

1.4. To C(onsequence) B, L, R, G → C. Now only a few expressions from a set **Ax** of typical examples and rules from **R** are sufficient to produce (prove) all the facts from **B** (at least all elements of **B** are typical examples). The proof mechanism **c** (see the procedure consequence-aim) is the classical one — the axioms are proved and if the elements of the input of a rule are proved then this is the proof of the output. When we enlarge the theory with these rules, we use the *induction* and *substitution* (Figure 1) principles (methods) to generate all the consequences of the theory. In such a way we produce a *classical theory* $\mathbf{CT} = \langle \mathbf{B}, \mathbf{L}, \mathbf{R}, \mathbf{C}\rangle$.

1.5. From theories $\mathbf{T}_1, \ldots, \mathbf{T}_n$ to M(etatheory) B, L, R, C → M. For every set $\mathbf{B}_i$ of facts we can construct in such manner a theory $\mathbf{t}_i$. Some of these theories may have different languages.

For instance the theories: TM of Turing machines, RF of recursive functions, FG of formal grammars, ... are theories of computations. Analogously the theories: FA of finite automata, RL of regular languages, KE of Kleene expressions are also about computations. These theories speak in different languages. For the general theory of computation, we need a general language and general inference method. There are two most popular methods to do that:

1. To find a general language (as in mathematics – set theory or abstract algebra) and to use the classical logic inference methods.

2. To find an inference method that controls the inferences of the partial theories and that allow jumping from one theory to another.

It seems that the second method is more human and effective. Using some mixture procedures, we can work in "small" theories jumping from one theory to another. Such an instrument works well when the theories are equivalent, mutually consistent or ordered by inference power. The theories from the first (second) group of computation theories mentioned are equivalent and every element of the second group is a proper subtheory of the corresponding element of the first group. Hence, we can prove the theorems in these theories in which it is easy to do (hence easy to understand). Therefore, the proof of the existence of a fixed-point can be produced in λ-calculus and the proof of Cook's theorem – in Turing Machine theory.

In such a way we produce the mixed theory $\mathbf{t} = \langle \mathbf{B}, \mathbf{L}, \mathbf{R}, \mathbf{M}(C) \rangle$ where $\mathbf{B}$, $\mathbf{L}$, $\mathbf{R}$ are the unions of the corresponding objects $\mathbf{t}_1, \ldots, \mathbf{t}_n$ and $\mathbf{M}(C)$ is the union of all consequence procedures $\mathbf{c}_1, \ldots \mathbf{c}_n$.

The side effect of such generalizations of the language and the rules of the theory is that more sentences are expressible in $\mathbf{L}$ and more "theorems" are provable with the help of the consequence procedure $\mathbf{c}$, rules $\mathbf{R}$, respectively. Some of these expressions have no correspondence in the situations described and respectively in the knowledge base $\mathbf{B}$. Naturally some of the "theorems" will be speculations. Hence at that stage conflicts may appear. Moreover, the mixture of the theories may introduce a conflict. For instance in the theory of the concrete computer (with the help of which this article is written) we use an inconsistent family of "axioms" and theories.

Example: The general theories consist of the following axioms (theorems):

- Every computer is a finite automaton.

- Most programming languages are at least context-free.

- The operation system is a universal program.

These contradictions do not shame us. Nevertheless, we use these theories only in a “rough” sense. When we have to perform many iterations we remember that it is a finite automaton. Hence, we control the inference procedure.

Analogously we have more concrete rules:

- A parity error means that a piece of memory has a defect.
- If it cannot recognize the hard disk then either the disk is faulty or the battery is old.

But ERROR appears. From the tests of the memory it follows that it is OK. Moreover tests of all the elements tell us that they are OK. Hence we have to change the theory of that concrete computer. The result is a stronger complicated theory and a possible solution – shake the computer.

In such cases we have to eliminate the conflict. In that concrete case we return to the metatheory — format the disk and shake and clean the computer. In the overtheory of economics the problem costs more than the computer but uses the same methods.

1.6. To E(limination) B, L, R, C, M → E. When a theory is produced there are the following main possibilities:

- A. In this theory a conflict appears: Provable are C and $\neg C$.
- B. We do not observe a contradiction in the theory **t**.

In any one of these cases we usually proceed using the “pure” strategies “change the inference” or “change the language” or any mixture of simple strategies, for instance:

- [$\mathbf{a}'$] when contradictions between the theory and the knowledge base appear:
- [$\mathbf{a}'$.1] Change the rules (proof mechanism) to eliminate the appearance of the conflict.
- [$\mathbf{a}'$.2] Change the language to eliminate one of the conflict elements (this is possible because C is a translation of an element from **B** while its negation is not).
- [$\mathbf{B}'$] when we do not observe a contradiction: Claim that the theory **t** describes the Universe and its neighborhood.

For instance to eliminate the conflict mentioned in the example above we

- $\mathbf{a}'.1$ read only the instruction of the concrete computer;

or

- $\mathbf{a}'.2$ say that "computability" on that computer is different from "computability" in formal theories.

Once the contradictions are eliminated from the theory it is ready to work. The first (mnemonic) step is to find some minimal "basic" expressions with the help of which all the expressions from $\mathbf{B}$ are provable (note that this is another hard problem). Claim that these basic expressions are the *postulates* (axioms) $\mathbf{Ax}$ of the theory $\mathbf{t}$. Put these postulates instead of the basic facts $\mathbf{B}$.

Now the simplifications in points A' restrict the power of the theory $\mathbf{t} = \langle \mathbf{B}, \mathbf{L}, \mathbf{R}, \mathbf{c} \rangle$ (either we change the language — hence it is not so general; or we change the inference mechanism — hence it is not so simple).

2. Formal theories.

As we see the formal theories are built in at least two levels — elementary theories and generalized theories. In other words formal theories appear in two instances:

- as a collection of information (when they are based on elementary rules);
- as a generator of hypotheses (when they are based on more complicated proof mechanisms).

The first step in theorizing any knowledge is to collect facts and to find some connections between these facts. The collections of facts (with the elementary rules) are elementary theories Later we generalize these theories to produce theorems and prove new facts. Hence we introduce generalized theories. In these theories undecidability appears, whence all the problems of the classical first-order calculi. In practice the generalized theories are extended with some heuristic mechanisms hence they usually are based on more complicated consequence procedures. To express some of these procedures we introduce a small PASCAL-like programming language.

2.1. A programming language.

The programming language we use is an extension of basic PASCAL (or any other popular programming language) with a class of choice functions **choose**, and mixture (parallel) operations ρ

and σ. The result of the operation ρ is the simultaneous execution of both procedures the (intersection) accumulation; the operation σ compares the results of both procedures and if they are equal adds the result in the (intersection) accumulation. The other instruments (functions and procedures) are programmable in the following manner:

```
input: $\mathbf{x}$, $y$, $\mathbf{z}$
if $\mathbf{x} = \lambda$ then $\mathbf{x}[y/\mathbf{z}] := \lambda$
if $\mathbf{x} = y$ then $\mathbf{x}[y/\mathbf{z}] := \mathbf{z}$
if $\mathbf{x} = a_n\mathbf{t}$ and $y \neq a_n$ then $\mathbf{x}[y/\mathbf{z}] := a_n\mathbf{t}[y/\mathbf{z}]$
output: $\mathbf{x}[y/\mathbf{z}]$
```

Fig. 1. The substitution $\mathbf{x}[y/\mathbf{z}]$

The programming language we use is an extension of basic PASCAL (or any other popular programming language – we have a simple translation procedure of that language) with an operation choose, and (parallel) operations ρ and σ. The operation ρ realizes dynamically the union and the operation σ – the intersection of sets of expressions.

The other instruments (functions and procedures) are programmable in the following manner:

The result of the substitution we shall denote by $\mathbf{x}[y/\mathbf{z}]$. When we make more substitutions we write $\mathbf{x}[y_1/\mathbf{z}_1, \ldots, y_n/\mathbf{z}_n]$ instead of $\mathbf{x}[y_1/\mathbf{z}_1], \ldots, [y_n/\mathbf{z}_n]$, and assume that the substitutions are made sequentially.

With the help of the substitution we define the formal language of all logical expressions (Figure 2):

```
input: B(ase), O(perations)
$\mathbf{E} := \mathbf{B}$
while false do
$\otimes := \text{choose}(O)$;
$A_1 := \text{choose}(E); \ldots A_{n(\otimes)} := \text{choose}(E)$;
$E := E \cup \{\otimes (A_1, \ldots, A_{n(\otimes)})\}$
output: E(xpressions)
```

Fig. 2. The language “-all”

where $\otimes$ is an operation and E is the set of all expressions. This procedure produces all the expressions of the logical language but does not halt. With the help of the operations ρ and α we make the aim-form of that procedure – instead of **false** we use some A that verifies the fact that the expressions we need are obtained. For instance if we want to verify that A_1 is of the intersection in two theories we generate it in both theories. The logic speaks about the set of all expressions while we need only some concrete expressions. But it is easy to transform every "-all" procedure to an "-aim" procedure.

The aim-form of the language generator looks as in Figure 3.

Note that the aim may not be the expression but only some properties of the expression we want to obtain. For instance — to have at least two conjunctions or disjunctions (depth 2 of the formula). It is clear that the choice operations can help us to reach the aim faster.

```
input: B(ase), O(perations), A(im)
E := B
repeat until A
    $\otimes$ := choose(O);
    for $i \le n(\otimes)$ do $A_1$ := choose(E);
    E := E $\cup \{\otimes(A_1, \dots, A_{n(\otimes)})\}$
output: E(xpressions)
```

Fig. 3. The language "-aim"

The other aim-type procedures are produced in the same manner.

2.2. Elementary theories. *An elementary rule* is a pair $r = \langle I_r, O_r \rangle$ of sets of expressions where I_r is the *input* (premises) and O_r is the *output* (conclusions) of the rule r. *An elementary formal theory* is a pair $\mathbf{t} = \langle \mathbf{L}, \mathbf{R} \rangle$ where $\mathbf{L}$ is a formal language; $\mathbf{R}$ is the set of elementary rules.

In other words an elementary theory is a directed graph. A *proof* of an expression C is any successful execution of the consequence procedure (Figure 4) with aim C.

These procedures are *context-free* — the proof doesn't depend on the aim of the proof. In practice we use *context-sensitive* proof procedures in which the choice functions depend on the aim of the proof and the corresponding strategies used in the general procedure.

```
input: B(ase), R(ules), A(im)
C := B
repeat until A
    for $i \leq n$ do
    $A_1$ := choose($\mathbf{C}$);
    $r$ := choose($\mathbf{R}$);
     if $\langle A_1 \dots A_n \rangle = I_r$ then $C := C \cup O_r$
output: C(onsequence)
```

Fig. 4. Procedure consequence "-aim"

2.3. Generalized theories. The generalized theories are based on generalization rules. A form of the procedure which can realize a generalized rule is given in Figure 5.

```
input: B(ase), A $\mathbf{x}_1 \dots \mathbf{x}_n$ $\mathbf{L}_1 \dots \mathbf{L}_n$ B
    for $i \leq n$ do $D_i$ := choose($L_i$);
    $\mathbf{B} := \mathbf{A}[\mathbf{x}_1/D_1 \dots \mathbf{x}_n/D_n]$
output: $\mathbf{A}[\mathbf{x}_1/D_1 \dots \mathbf{x}_n/D_n]$
```

Fig. 5. Generalized rule "-step"

The generalized rule-step procedures are the basic inference objects in the generalized theories. Iteration of the rule-step procedure gives a global inference.

Note that there are interesting implication theories [15] based on a single generalized rule.

A *proof* of an expression C is any successful execution of the iteration of the consequence-aim procedure.

If we have more rules we use the parallel Δ execution of these rule-steps.

The Tarski relation $\mathbf{Cn}(\mathbf{t})$ of all the consequences of the theory $\mathbf{t}$ is obtained when we put false instead of test. Hence the consequence relation is the ω-result of the loop.

In this definition the language is arbitrary and not formalized. If $\mathbf{R}$ and $\mathbf{B}$ are finite, then the set $\mathbf{c}$ is also finite even when the procedure has a loop. The infiniteness comes from the connection between the language-generator and the proof-generator.

Both definitions of Expressions and Proof are inductive and we may connect them.

2.4. Semantics. A theory is *consistent* if no contradiction can be proved in it. But a contradiction may appear in nontrivial theories while some trivial ones may have no contradictions. Hence it is more effective to say that a theory is consistent if it has a model (Consistency is a property of the deduction system. Model here means 'has a semantic function'. Formal proofs of completeness theorems have the antinomial effect of describing one formal proof by another. At the end the pure arithmetics of natural numbers is outweighed by thousands of mathematical theories). From the logical point of view the semantics is any total mapping of the formulas in the set of truth values. But when we know only a few facts it is natural to consider partial mappings. Hence we define:

Definition. *A partial semantical function* (psf) *is any mapping* s : $\mathbf{L} \to \{0,1\}$ *such that:*

$s(A \wedge B) = 1$	implies	$s(A) = 1$ and $s(B) = 1$
$s(A \wedge B) = 0$	implies	$s(A) = 0$ or $s(B) = 0$
$s(A \vee B) = 1$	implies	$s(A) = 1$ or $s(B) = 1$
$s(A \vee B) = 0$	implies	$s(A) = 0$ and $s(B) = 0$
$s(\neg A) = 1$	implies	$s(A) = 0$
$s(\neg A) = 0$	implies	$s(A) = 1$

Table 6. A partial semantical function

A psf s *satisfies* the set $\mathbf{B}$ of expressions ($s \Rightarrow \mathbf{B}$) iff $s(B) = 1$ for all $B \in \mathbf{B}$. If s satisfies $\mathbf{B}$ then we say that s is a *partial model* of $\mathbf{B}$.

Let R be a rule of inference. It is *admissible* for the set $\mathbf{B}$ if there exists a psf s such that $s/_B \sim 1$ and $s(I_R) = 1$ implies $s(O_R) = 1$.

Lemma. *Obviously we have the following trivial fact:*

Every partial semantic function may be extended to a total one. Hence if a set has a psf then it is consistent.

Moreover we have the following:

Proposition. *For every set* $\mathbf{B}$ *and every rule* R *there is a language* $\mathbf{L}$ *and a nontrivial plf* s *such that* R *is admissible for* $\mathbf{B}$.

Proof. If the input and output of R contain a contradiction then we define the negation to be nonclassical. Hence it is allowed that $s(A) = 1$ and $s(\sim A) = 1$. And we define it so. If we have more rules then we divide their languages as in the procedure **c**-control. $\square$

3. Conclusions. This paper is a continuation of [7], [13], [15] where some relations and some operations over theories are investigated. Relations and operations over theories as well as changing theories can be found in many papers by different authors (compare with [4]), [10], [15], [19]). We think that the theory change is a form of elimination of the conflicts which appear in the inference procedures. Hence instead of universal control of the connections between theories we introduce moving control inside the inference procedures.

Suppose that all the theories are built in such manner. From the mnemonic principle it follows that we have to minimize the language and the set of axioms. Hence we minimize the field of applications. Simultaneously we solve the next problem: how to combine theories to produce new (or faster) theorems?

In the procedures mentioned in section 2 a nontrivial element is the operation 'choose'. Here is not the place to speak about it. In articles about automatic proof one may find many choice strategies but no language generation strategy.

Sometimes the problem is how to express the theorems and theories for another (intelligent) agent. One way is to use a family of pseudotheories and pseudoproofs to convince less exactly agents of the thesis and simultaneously to "save face" — to continue the proof if it is needed.

There are some interesting directions for investigation:

How are the "magic" proofs produced – the proofs in which the rabbit is firstly put in the tophat to be later effectively taken out?

How long is the circle proof in dependency of the depth of the control of the listener?

Acknowledgments. Prof. Jan Żytkow read the previous version of that paper and in light of his notes it has to be a joint publication, but he is not among us now. Hence I shall never have his agreement to use his name in the authors of this article.

REFERENCES

[1] Allen J. Natural Language Understanding. Benjamin and Cummings, Menlo Park, CA, 1987.

[2] Barwise J., J. Perry. Situations and Attitudes. MIT Press, Cambridge, Mass., 1983.

[3] Benoit W., D. Hample, P. Benoit, eds. Readings in Argumentation. Foris Publications, Berlin, 1992.

[4] Brusilovskii B. System theory and system of theories. Vyshcha Shkola, Kyiv, 1977 (in Russian).

[5] Davidson D. Essays on Actions and Events. Oxford University Press, 1990.

[6] Dewey J. How we think. A restatement of the relation of reflective thinking to the education process (Revised edn), Boston: D. C. Heath, 1933.

[7] Gargov G., S. Radev. Expert logics. In: Artificial Intelligence: Methodology, Systems, Applications (Eds P. Jorrand, V. Sgurev), AIMSA'86, North Holland, 1986, 181–188.

[8] Hintikka J. Eseje logiczno-filozoficzne. PWN, Warszawa, 1992.

[9] Ivanov V. Even and Odd: Asymmetry of the Brain and Symbol Systems. Sovetskoe Radio, Moskva, 1978 (in Russian).

[10] Muskens R. Anaphora and the logic of change. In: Logics in AI—JELIA'90 (Ed. J. van Eijck) Springer, 1991, 413–427.

[11] Popper K. Philosophy of Science: a Personal Report, 1957.

[12] Rich E., K. Knight. Artificial Intelligence. McGraw-Hill, Inc., New ork, 1991.

[13] Radev S. Extensions of PDL and consequence relations. In: Computation Theory (Ed. Andrzej Skowron) Fifth Symposium, Zaborów, Polska, vol. **108** of LNCS, Springer, 1985, 251–264.

[14] Radev S. Some Logics for Argumentation Systems. XV Warsztaty Informatyki Teoretycznej, 08–09 grudnia, 2001, Białystok.

[15] Radev S. Argumentation Systems. Białystok, 2006 (in print).

[16] Vreeswijk G. Studies in Defeasible Argumentation. PhD, Vrije Universiteit te Amsterdam, Amsterdam, 1993.

[17] Winston P. Artificial Intelligence. Addison-Wesley, Reading, Massachusetts, 1992.

[18] Witteveen C. A sceptical semantics for truth maintenance. In: Truth maintenance systems. Springer, 1991, 136–154.

[19] Żytkow J. Intertheory relations on the formal and semantical level. In: Formal Methods in the Methodology of Empirical Science. (Eds M. Przecki, K. Szaniawski, R. Wócicki, G. Malinowski) Wrocław, Ossolineum, 1976, 450–457.

Bulgarian Academy of Science
Institute of Mathematics and Informatics
Department of Mathematical Linguistics
Acad. G. Bontchev Str., Bl. 8
1113 Sofia, Bulgaria

Technical University of Bialystok
Faculty of Computer Science
15, Wiejska Str.
Bialystok, Poland
e-mail: sradev@go.com, sradev@wp.pl

Received September 29, 2006
Final Accepted March 12, 2007