Journal of Informatics and Mathematical Sciences
Volume 2 (2010), Numbers 2 & 3, pp. 109–119
© RGN Publications

http://www.rgnpublications.com

Frontiers of Topology

R.D. Sarma

Abstract. In the last half century, topology as a subject has undergone vast changes. From its status of "Rubber sheet geometry", topology has come a long way to be one of the most important subjects not only in Mathematics, but amongst various diverse systems and branches of modern sciences such as fuzzy logic, computing, artificial intelligence, pattern recognition, image–processing etc. In the process, the subject matter of topology has increased manifold. In this article, we discuss some of such new frontiers touched by topology. While interactions of logic programming and dislocated metric spaces are provided in detail, other areas are given only introductory mentions in the article.

1. Topology And Fuzzy Set Theory

L.A. Zadeh introduced the notion of a fuzzy set in 1965 [15]. Introduction of fuzzy sets was an answer to the problems posed by the two-valued logic system. Although there was initial skepticism, very soon the notion of fuzzy set gained a sound footing. Scientists all over the world switched over to this new system which they found more natural and nearer to the real life situations. In pure mathematics, topology was among the first few branches where fuzzy set theory was systematically applied. The first paper on fuzzy topology by C.L. Chang appeared in 1968 [2]. In fact, in the year 2008, fuzzy topology is completing 40 years of its existence. During this period, thousands of research papers have been published in this area.

As fuzzy sets generalize the classical set theory, fuzzy topology also generalizes the classical topology. In fact, TOP turns out to be a subcategory of FTOP, the category of fuzzy topological spaces, with continuous mappings as morphisms. Although there was criticism in the beginning, later on it was found that fuzzy topology has been able to remove two major deficiencies of TOP [5]:

2000 *Mathematics Subject Classification.* 50A05.
Key words and phrases. Topology, dislocated metric, fuzzy topology, logic program.

(i) There are many locales which can not be represented by any classical sober topological space. But every locale, indeed, every complete lattice has at least one representing sober fuzzy topological space.

(ii) Many localic products are not (isomorphic to) any product topology; but every localic product is a product fuzzy topology.

Apart from these,

(iii) Fuzzy topology gives a much richer representation of distributive lattices than classical topology alone can - for each complete distributive lattice L, there is a category of representing fuzzy topological spaces.

(iv) Classical digital topology fails in gray-area modeling; whereas fuzzy digital topology is found to be suitable for this purpose.

After fuzzy set theory, other newer theories are coming up dealing with the shortcomings of the classical set theory. L-fuzzy sets, intuitionistic fuzzy sets, rough sets etc. are amongst them. Studying topological structures in these new systems has been a thrust area of present day research in topology [5,7,8].

2. **Topology and logic programming**

Another important area where topology has recently found applications is the field of logic programming. Topology has entered here indirectly through a measure called 'dislocated metric'. Dislocated metrics were first studied by S.G. Methews in 1986 [4] in the domain theory; however it was Hitzler and Seda, who used it in logic programming and accordingly defined dislocated topology in 2000 [3]. In the following, we provide a step by step discussion about the interrelationship of dislocated metric and logic programming..

2.1. *Dislocated metrics*

Definition 2.1. Let X be any set and $\rho : X \times X \to \mathbb{R}^+ \cup \{0\}$ be a function. Then ρ is called a dislocated metric if

(i) $\rho(x, y) = 0 \Rightarrow x = y$;

(ii) $\rho(x, y) = \rho(y, x)$;

(iii) $\rho(x, y) \leq \rho(x, z) + \rho(z, y)$.

Thus we need not have, $\rho(x, x) = 0$.

Example 2.2. Let $X = \mathbb{Z}^+ \cup \{0\}$, $\rho(x, y) = (x + y)/2$. Then ρ is a dislocated metric on X. Some open balls of (X, ρ) are

$$B_1(1) = \{0\}, \quad B_2(1) = \{0, 1, 2\}, \quad B_{1/2} = \varnothing.$$

Some peculiarities of d-metric are

(i) $B_\varepsilon(x)$ need not contain x;

(ii) A constant net $\{x_n = x\}$ need not converge to x

Some results concerning d-metric spaces [3, 9]:

(i) Let (X, ρ) be a d-metric space and $f : X \to X$ be a contraction mapping. Then f is continuous.

(ii) Let (X, ρ) be a complete d-metric space and $f : X \to X$ be a contraction. Then f has a unique fixed point (Banach's Contraction mapping theorem).

(iii) Let (X, ρ) be a complete d-metric space and $\{F_n\}$ be a decreasing sequence of non-empty closed subsets of X such that diam $(F_n) \to 0$. Then $F = \bigcap_{n=1}^{\infty} F_n$ contains exactly one point (Cantor's Intersection theorem).

2.2. *Applying dislocated metrics in logic programming*

In this section, we first acquaint our reader with the basic concepts of logic programming. We heavily depend on Llyod [4] for the contents of this section.

Logic programming began in early 1970's as a direct outgrowth of the earlier work in automatic theorem proving and artificial intelligence. The credit for the introduction of logic programming goes mainly to Kowalaski and Colmerauer [4]. In 1972, Kowalaski and Colmerauer were led to the fundamental idea that " *Logic can be used as a programming language*".

The acronym PROLOG (PROgramming in LOGic) was conceived accordingly.

There are two major, rather different, classes of logic programming languages currently available. One may be called 'system' language and the other 'application' language. However, this division is not precise; it is just to give a flavour of the two classes of languages. PARLOG, PROLOG, GHC etc. fall in the first category. Whereas Quintus PROLOG, micro-PROLOG and NU-PROLOG fall in the second category. The system languages emphasize on AND-parallelism, definite progamme etc., while the application languages emphasize on OR-Parallelism, and unrestricted progams etc.

First order logic has two aspects: syntax and semantics. The syntactic aspect is concerned with well-formed formulas admitted by grammar of a formal language, as-well-as deeper proof-theoretic issues. The semantics is concerned with the meanings attached to the well-formed formulas and the symbols they contain. A first order theory consists of an alphabet, a first order language, a set of axioms and a set of inference rules. The axioms are a designated subset of well-formed formulas. The axioms and rules of inference are used to derive the theorems of the theory.

Definition 2.3. An alphabet consists of seven classes of symbols:

(a) variables denoted by $x, y, z, u, v, w, \ldots$

(b) constants denoted by $a, b, c, \ldots$

(c) function symbols denoted by $f, g, h, \ldots$

(d) predicate symbols denoted by $p, q, r, \ldots$

(e) connectives $\wedge, \vee, \sim, \to, \longleftrightarrow$ (conjunction, disjunction, negation, equivalence)

(f) quantifiers $\exists, \forall$ (existencial and universal)

(g) punctuation symbols such as , .

Definition 2.4. A term is defined inductively as follows:

(a) a variable is a term
(b) a constant is a term
(c) if f is any n-ary function symbol, $t_1, t_2, \dots, t_n$ are terms, then $f(t_1, t_2, \dots, t_n)$ is a term.

Definition 2.5. A (well-defined) formula is defined inductively as follows :

(a) if p is an n-ary predicate symbol and $t_1, t_2, \dots, t_n$ are terms, then $p(t_1, t_2, \dots, t_n)$ is a formula (called an atomic formula or an atom).
(b) if F and G are formulas, so are $\sim F, F \wedge G, F \vee G, F \longrightarrow G, F \longleftrightarrow G$.
(c) if F is a formula and x is a variable, then $(\forall x F)$ and $(\exists x F)$ are formulas.

For an 1-ary predicate symbol p and a term $t, p(t)$ may be treated as a property of t; whereas for any n-ary $p, p(t_1, t_2, \dots, t_n)$ may be treated as a relation amongst $t_1, t_2, \dots, t_n$. Some examples of formulas are

$$\forall\ x\ \exists\ y(p(x,y) \rightarrow q(x))$$
$$\sim \exists\ x(p(x,a) \wedge q(f(x)))$$
$$\forall\ x(p(x,g(x)) \rightarrow q(x) \wedge \sim r(x))$$

Definition 2.6. The first order language given by an alphabet consists of the set of all formulas constructed from the symbols of the alphabet.

Thus a language is a collection of all atoms based on the predicate symbols and terms obtained from the constants, variables, functional symbols, all formulas derived from the atoms by use of connectives $\wedge, \vee, \sim, \rightarrow, \longleftrightarrow$ and quantifiers $\exists, \forall$.

After defining a first order language, now we proceed to define a program:

Definition 2.7. (a) A literal is an atom or the negation of an atom. A positive literal is an atom; a negative literal is the negation of an atom.
(b) A clause is a formula of the form

$$\forall x_1 \forall x_2 \forall \dots \forall x_n (L_1 \vee L_2 \vee \dots \vee L_m),$$

where each L_i is a literal and $x_1, \dots, x_n$ are variables occurring in $L_1 \vee \dots \vee L_m$.
Some examples of clause are

$$\forall x\ \forall y\ \forall z(p(x,y,z) \vee \sim q(x,y) \vee \sim r(y,z))$$
$$\forall x\ \forall y(\sim p(x,y) \vee r(f(x,y),a))$$

The clause $\forall x_1 \forall x_2 \forall \dots \forall x_s (A_1 \vee A_2 \vee \dots \vee A_m \vee \sim B_1 \vee \dots \vee \sim B_n)$ is equivalent to $\forall x_1 \forall x_2 \forall \dots \forall x_s (A_1 \vee A_2 \vee \dots \vee A_m \leftarrow B_1 \wedge \dots \wedge B_n)$ which is also written as $A_1, A_2, \dots A_m \leftarrow B_1, \dots, B_n$, where all variables are universally quantified, antecedent is connected by conjunctions and consequents occurs with disjunction, A_i, B_j's are all atoms.

Definition 2.8. A definite program clause is a clause of the form $A \longleftrightarrow B_1, \ldots, B_n$, which contains precisely one atom in its consequent. A is called the head and $B_1, \ldots, B_n$ is called the body of the program clause.

In this case, we have, "for each assignment of each variable, if $B_1, \ldots, B_n$ are all true, then A is true".

Definition 2.9. A definite program is a finite set of definite program clauses.

The theory of definite programs is simpler in the sense that definite programs do not allow negations in the body of the clauses.

So far we have discussed syntax of a logic program, especially of definite program. The semantics of a program is concerned with the meanings attached to the well-formed formulas and symbols they contain. The declarative semantics of a logic program is given by the usual semantics of the formulas in the first order logic. In order to be able to discuss the truth or falsity of a formula, it is necessary to attach some meaning to each of the symbol in the formula first. The various quantifiers and connectives have fixed meanings, but the meanings attached to the constants, function symbols and predicate symbols may vary. An interpretation consists of some domain so that the variables may be assigned some range in the domain; along with it contains assignment to each constant element, also there is assignment to each function symbol and to each predicate symbol. An interpretation thus specifies a meaning for each symbol in the formula. An interpretation for which the formula expresses a true statement is called a model formula.

Formally we define pre-interpretation, interpretation and model in the following way:

Definition 2.10. (i) A *pre-interpretation* J of a first order language L consists of the following:
- (a) a non-empty set D, called the Domain of the pre-interpretation;
- (b) for each constant in L, the assignment of an element in D;
- (c) for each n-ary function symbol in L, the assignment of a mapping from D^n to D.

(ii) An *interpretation* I of a first order language L consists of a pre-interpretation J with domain D, together with the following:

For each n-ary predicate symbol in L, the assignment of a mapping from D^n to {true, false}.

Definition 2.11. (i) A *variable assignment* V with respect to J is an assignment to each variable in L of an element in the domain of J.

(ii) For L, J, D, V with their usual meanings, *term assignment* is defined as follows:
- (a) each constant is given assignment according to J;

(b) each variable is given assignment according to V;

(c) if $t'_1, \ldots, t'_n$ are the term assignment of $t_1, \ldots, t_n$ and f' is assignment of n-ary function symbol, then $f'(t'_1, \ldots, t'_n) \in D$ is the term assignment of $f(t_1, \ldots, t_n)$.

(iii) Let I be an interpretation of a first order language L, and let F be a closed formula of L. Then I is a *model* for F if F is true with respect to I. If S is a set of formulas, then I is a *model* for S if I is a *model* for each formula of S.

Example 2.12. Consider the interpretation I:

Domain is the non-negative integers;
a is assigned 0 ;
b is assigned 1;
f is assigned the successor function $x \to x+1$;
p is assigned the relation $\{(x,y) : x < y\}$;
q is assigned the relation $\{x : x > 0\}$;
r is assigned the relation $\{(x,y) : x \text{ divides } y\}$.

Then for the following formulas

(a) $\forall x\ \exists y p(x,y)$; *informal semantics*: for each x, there exists y that $p(x,y)$ is true;

meaning: for every non-negative integer x, there exists a non-negative integer y such that x is less than y.

I is model for this formula.

(b) $\exists x \forall y p(x,y)$ is false, hence I is not a model for this formula.

(c) $p(f(a),b)$ is false as $f(a) = f(0) = 1$ as $b = 1$. Therefore I is not a model for this formula.

In the above example(c), the formula $p(s(a),b)$ is a special one. Its terms $s(a), b$ do not involve variables. A term not containing variables is called a ground atom. The formula $p(s(a),b)$ is a ground atom for which interpretation I is not a model.

Definition 2.13. Let L be a first order language. The *Herbrand Universe* U_L for L is the set of ground terms, which can be formed out of the constants and function symbols appearing in L. The *Herbrand base* B_L for L is the set of all ground atoms which can be formed by predicate symbols of L with ground terms from the Herbrand universe as arguments. The Herbrand pre-interpretation for L is given by the following:

(a) the domain U_L;

(b) the constants in L are assigned themselves in U_L;

(c) if f is an n-ary function symbol in L, then the mapping from $(U_L)^n$ to U_L defined by $(t_1, \ldots, t_n) \to f(t_1, \ldots, t_n)$ is assigned to f.

A Herbrand interpretation for L is any interpretation based on the Herbrand preinterpretation for L.

In a Herbrand interpretation, the assignment to constants and function symbols is fixed, thus, it is possible to identify a Herbrand interpretation with a subset of the Herbrand base. In fact, for any Herbrand interpretation, the corresponding subset of the Herbrand base is the set of all ground atoms which are true with respect to the interpretation. Conversely, given any arbitrary subset of the Herbrand base, there is a corresponding Herbrand interpretation defined by specifying that the mapping assigned to a predicate symbol maps some arguments to 'true', precisely when the atom made up of the predicate symbol with the same arguments is in the given symbol. Thus one may identify a Herbrand interpretation as a subset of the Herbrand base, which is the collection of all ground atoms of L.

We explain the above concepts with the help of the following example:

Example 2.14. Consider the program P

$$p(x) \leftarrow q(f(x), g(x))$$
$$r(y) \leftarrow$$

which has an underlying first order language L based on the predicate symbols p, q and r and the function symbols f and g. Since L has no constant, we add some constant, say a, to form ground terms.

The Herbrand universe for L is

$$U_L = \{a, f(a), g(a), f(f(a)), f(g(a)), g(f(a)), g(g(a)), \ldots\}$$

The Herbrand base for L is

$$B_L = \{p(a), q(a,a), r(a), p(f(a)), p(g(a)), q(a, f(a)), q(f(a), a), \ldots\}$$

The Herbrand pre-interpretation of P will consist of

Domain $= U_L$;

Constant a is assigned a;

Function f is assigned $t \to f(t)$ where $t \in U_L$.

Any Herbrand interpretation of P will essentially contain the above three features of the Herbrand pre-interpretation; they will differ only in their assignments regarding the predicate symbols.

Definition 2.15. Let L be a first order language and S be a set of closed formulas of L. A *Herbrand model* for S is a Herbrand interpretation I for L which is model for S, that is, all formulas of S are true with respect to I.

For a program P, we may normally assume that the underlying language is defined by the constants, function symbols and predicate symbols appearing in P. With this understanding, we refer to the Herbrand universe as and Herbrand base as of P.

Since every definite program P has B_P as a herbrand model, the set of all Herbrand models for p is non-empty. Thus the intersection of all Herbrand models for P is again a model, called the Least Herbrand model for P, denoted by M_P.

Now we are coming towards combining our discussions on syntax and semantics of definite programs. We have defined various aspects of a definite program and looked for the best suitable interpretation or meaning of a program. With M_P, the least Herbrand model, we have reached the answer. There are very strong reasons for regarding M_P as the natural interpretation of a program. In fact the atoms in M_P are precisely those which are logical consequences of the program (An atom F is a logical consequence of a set of formula S, if whenever each formula of S is true in an interpretation I, F is also true).

Let P be a definite program. Then I_P, the set of all interpretations of P, is a complete lattice under the partial ordering of set inclusion. We define a mapping T_P, called single step operator, $T_P : I_P \rightarrow I_P$ as follows: Let I be an Herband interpretation (that is, a subset of B_P). Then

$$\begin{aligned} T_P(I) &= \{A \in B_P : A \longleftrightarrow A_1, \ldots, A_n \text{is a ground instance of a clause in} \\ &\qquad P \text{ and } \{A_1, \ldots, A_n\} \subseteq I\} \end{aligned}$$

(For the definition of ground instance, please refer to [4]).

The single step operator is continuous under Scott topology on I_p. Hence it is monotonic as a lattice mapping. So by Knaster-Tarski Theorem, a least fixed point exists for the single step operator. In fact, M_P turns out to be the least fixed point of T_P.

However, the above lattice-theoretic method is helpful so long as we deal with definite programs where bodily atoms of the clauses are positive literals. If we enhance the syntax, say, as in normal program, allowing negative literals to occur in the body of the clauses, the single step operator fails to be monotonic. Hence a fixed point is not obtained by the above approach.

To solve this problem, the possible approaches are

(i) To restrict the syntax;
(ii) To change the operator;
(iii) To shift to other fixed point theorems.

It is in the third approach, that dislocated metrics come to our help. For this, we proceed as follows:

Let P be a logic program, I be a model for $P, l : B_p \rightarrow \mathbb{N}$ be a level mapping, where each atom/predicate is given a level. If for each ground instance
$A \longleftrightarrow L_1, \ldots, L_n$ of a clause in P, we have if $L_i \in I$, $i = 1, \ldots, n$, then $l(L_i) \prec l(A)$ then P is called an *Acceptable logic program with respect to l and I.*

For $J, K \in I_P$, we define $d(K, K) = 0$ and $d(J, K) = 2^{-n}$, where J and K differ on some atom A of level n, but agree on all ground atoms of lower levels. Then (I_P, d) is a complete metric space (Notice, earlier we found that $(I_p, \subseteq)$ is a complete lattice!).

Define

$$f : I_P \to \mathbb{R}, \text{ by } f(K) = \begin{cases} 0, & \text{if } K \subseteq I \\ 2^n, & \text{if } K \not\subseteq I, \end{cases}$$

where n is the smallest positive integer such that there is an atom a of level n in K but not in I.

Define

$$U : I_P \to \mathbb{R} \text{ by } U(K) = \max\{f(K'), d(K', I)\},$$

where K' is K restricted to the predicate symbols which are not in Neg_P^* (For the definition of Neg_P^*, refer to [3]). Finally define

$$\rho : I_P \times I_P \to \mathbb{R} \text{ by } \rho(J,K) = \max\{d(J,K), U(J), U(K)\}.$$

ρ is a d-metric on I_P: We find that

(i) $\rho(K,K) = U(K) \geq 0$ ($\rho(K,K) > 0$ if $K|_{Neg_P^*} \subsetneq I$, thus ρ is not a metric)
(ii) $\rho(K,H) = \rho(H,K)$
(iii) $$\begin{aligned}\rho(H,K) + \rho(K,L) &= \max\{d(H,K), U(H), U(K)\} + \max\{d(K,L), U(K), U(L)\} \\ &\geq \max\{d(H,K) + d(K,L), U(H), U(L)\} \\ &\geq \max\{d(H,L), U(H), U(L)\} \\ &= \rho(H,L)\end{aligned}$$

It can be further proved that [3]

(i) *(I_p, ρ) is a complete metric space.*
(ii) *$T_P : I_P \to I_P$ is a contraction mapping.*

Hence Banach's fixed point theorem holds for T_P. Therefore T_P has a unique fixed point. Since a supported model for P is a fixed point for T_P, it follows that

Every acceptable logic program has a unique supported model.

Thus we have seen that dislocated metrics help us overcome the drawback of the single step operator that it can no longer provide a fixed point which could be a reasonably acceptable interpretation of non-positive logic programs. So it is desirable that this metric and its corresponding topological space are investigated. Since the set of open balls of a d-metric space does not yield a conventional topology, a dislocated topology has been defined with a different approach. It was studied by Hitzler and Seda in 2000 for the first time. Another approach has been provided in Sarma [9].

Definition 2.16. Let (X,ρ) be a d-metric space. Let for $x \in X$, $\mathfrak{A}_x = \{A \subseteq X\colon$ there exists $\varepsilon > 0$ such that $B_\varepsilon(x) \subseteq A\}$. Then $\mathfrak{A}_x$ is called a d-neighborhood system for x. Let $\mathfrak{A} = \{\mathfrak{A}_x : x \in X\}$. Then $(X, \mathfrak{A})$ is called a d-topological space.

Some properties of d-topological spaces are studied in [3,9]. However, it is still in embryonic stage.

2.3. *Query topology*

Another important topology used in logic programming is query topology introduced by Batarekh and Subrahmaniam in 1988 [1, 11]. A *query* is a typed first order formula of the type

$$\leftarrow W$$

Where W is a typed first order formula and any free variables of W are assumed to be universally quantified at the front of the query.

Let P be a normal logic program with its underlying language L. Let J be a pre-interpretation of L with domain $dom(J)$ and I be an interpretation based on J. Let $X = I_L^J$ denote the set of all interpretations of L based on J. Given a query E in L, we define

$$G(E) = \{I \in X : E \text{ is in } I\}$$

and form the class

$$\hat{G} = \{G(E) : E \text{ is a query }\}.$$

The *query topology* is the topology on X generated by $\hat{G}$ as a subbasis. Thus a typical open set in the query topology is a union of sets of the type $G(E_1) \cap G(E_2) \cap \ldots \cap G(E_n)$.

Some characteristics of a query topology are

(i) *it is always second countable, irrespective of the cardinality of the domain D of J;*

(ii) *it is not Hausdorff.*

To avoid these typical characteristics, *atomic topology Q* is defined with some refinement on query topology [1] which has the following features

(i) *Topology Q is finer than the query topology; every basic open set is closed in Q.*

(ii) *Q is regular and Hausdorff, totally disconnected and compact.*

(iii) *Q and the query topology coincide if J is Herbrand pre-interpretation.*

(iv) *Q is Cantor topology.*

References

[1] A. Batarekh and V.S. Subrahmanian, The query topology in logic programming, *Proc. 1989 Symp. on Theoretical Aspects of Computer Science, Lecture notes in Comp. Sc.*, Springer Verlag, 1989, 375–387

[2] C.L. Chang, Fuzzy topological spaces, *J. Math. Anal. Appl.* **24** (1968), 182–190.

[3] P Hitzler and A.K. Seda, Dislocated topologies, *J. Elec. Eng.* **51** (10)(2000), 3–7.

[4] J.W. Llyod, *Foundations of Logic Programming*, 2nd edition, Springer Verlag, 1988.

[5] S.E. Rodabaugh, Pointwise lattice-theoretic topology, *Fuzzy Sets and Systems* **40**(1991), 297–345.

[6] A. Rosenfield, Fuzzy digital topology, *Infor. And Control* **29**(1)(1979), 97–117.

[7] R.D. Sarma and N. Ajmal, Fuzzy nets and their applications, *Fuzzy Sets and Systems* **51**(1992), 41–51

[8] R.D. Sarma, The theory of localization for intuitionistic fuzzy topological spaces, *J. Fuzzy Math.* **23**(2007), 711–724.

[9] R.D. Sarma, Relational topology, *Preprint*.
[10] A.K. Seda, Some applications of general topology to the semantics of logic programs, *Bull. Europ. Ass. for Theoret. Comp. Sc.* **52**(1994), 279–292.
[11] A.K. Seda, Quasi-metric and semantics of logic programs, *Fund. Inform.* **29**(1)(1997), 97–117.
[12] A.K. Seda and P. Hitzler, Topology and iterates in computational logic, *Proc. of 12th Summer Conf. on Gen. Top. and its Applications* **22** (1999), 427–469.
[13] A.K. Seda, Topology and semantics of logic programs, *Fund. Inform.* **19**(1) (1997), 97–117.
[14] Stephen Willard, *General Topology*, Addision-Wesely, 1970.
[15] L.A. Zadeh, Fuzzy Sets, *Inform. and Control* **8**(1965), 338–353.
[16] L.A. Zadeh, Fuzzy algorithms, *Inf. and Control* **19** (1969), 94–102.
[17] L.A. Zadeh, A computational approach to fuzzy quantifiers in natural languages, *Comp. and Math. with Appl. 9*(1983), 149–184.

R.D. Sarma, *Department of Mathematics, Rajdhani College (University of Delhi), Delhi 110 015, India.*
E-mail: `ratna_sarma@yahoo.com`

Received July 2, 2009
Accepted May 22, 2010