Multilevel decision logic: a formalism for rules mining

T. Y. Lin[1,2] *and* Xiaoling Zuo[3]
[1]*Mathematics and Computer Science Department,*
San Jose State University
San Jose, California 95120
408-924-512(voice), 408-924-5080 (fax), tylin@cs.sjsu.edu
[2]*Department of Electrical Engineering and Computer Science*
University of California
Berkeley, California 94720, USA
[3]*Department of Computer Science*
Shanghai Jiaotong University
Shanghai, P. R. China

Abstract

Decision logic is a logical formulation of rough set theory. It is an excellent formalism for expressing rules in relations. A multilevel decision logic is introduced to formalise the inferences and rules in relational databases.

Keywords

Decision logic, database security, database, decision table, information table, relation, multilevel security.

Database Security XI T.Y. Lin and Shelly Qian (Eds)

1 INTRODUCTION

Decision logic is a logical formulation of rough set theory; it is a convenient formalism for discussing rules and reasoning in relational databases. In this paper, we set up decision logic for multilevel (MLS) environment. Based on MLS decision logic, we may formalise and then discuss inferences in multilevel data. For single level decision logic, we refer readers to (Palwak, 1991).

Rough set theory (RS) is a formal theory derived from information tables (IT), also known as information systems or knowledge representation systems. . Loosely speaking IT's are relation or view instances in relational databases (RDB). RS is a theory on extensions of relational databases (ERDB) – snap shots of relational databases. However, unlike ERDB which focuses on storing and retrieving data from secondary storage, RS emphasizes discovering patterns, rules and knowledge in data - a sub-discipline of modern data mining theory. In this paper, we are adopting RS methodology to MLS ERDB.

2. INFORMATION TABLES & RELATION INSTANCES

The structure of IT is very similar to relations. Entities in IT are also represented by tuples of attribute values. However, an RS-representation may not be faithful, that is, the correspondence between entities and tuples may not be in one to one fashion.

Relation Instance

A relation instance(RI) consists of following items:

(1) implicitly a set of entities $U = \{u, v,..\}$ to be modelled

(2) a set of attributes, $T=\{A_1, A_2, .. A_n\}$

(3) a set of values, $Dom(A_i)$, for each attribute A_i, and their union denoted by

$$Dom(T) = dom(A_1) \cup dom(A_2) \cup .. \cup dom(A_n)$$

(4) a set of maps, where each map, called a tuple, represents an entity uniquely,

$$t : T \rightarrow Dom,$$

where $t(A) \varepsilon dom(A_i)$ for each $A_i \varepsilon T$.

Informally, one can view relation as a table consists of rows of elements. Each row represents an entity uniquely.

Information Table

An information table(IT), also known as an information system, a knowledge representation system, is a 2-tuple (U, T), by abuse of notation denoted by U again, such that it consists of

(1) explicitly a set of entities $U = \{u, v, ..\}$

(2) a set of attributes $T = \{A_1, A_2, .. A_n\}$

(3) a set of values, $dom(A_i)$, for each attribute A_i, and their union denoted by

$$Dom\ (T) = dom(A_1) \cup dom(A_2) \cup .. \cup dom(A_n)$$

(4) a map, $\rho: U \times T \rightarrow Dom$, called description function, such that

$$\rho(u, A_i) \in dom(A_i)$$

for all $u \in U$ and $A_i \in T$.

Note that ρ induces a set of maps

$$t = \rho(u, \bullet) : T \rightarrow Dom.$$

The image of each map is a n-tuple:

$$t = (\rho(u, A_1), \rho(u, A_2), ..., \rho(u, A_i), .. \rho(u, A_n))$$

Proposition For each relation instance, there is a naturally associated an information table.

Proof: To prove this proposition, it means to construct the following description function

$$\rho: U \times T \rightarrow Dom$$

from the given set of tuples

$$t : T \rightarrow Dom.$$

Let us name each tuple t by u(t). The collection of these names, each of which represents an "implicit" entity, forms a universe U. We will define the description function on such a universe U and the attribute set T as follows:

$$\rho(u(t), A_i) = t(A_i).$$

This completes the proof.

Note that in IT, a tuple t is not necessarily associated with entity u **uniquely;** two distinct entities could have the same tuple. However this is ***not permissible*** in relational databases.

Decision Table

A decision table(DT) is an information table (U, T, V, ρ) in which the attribute set T = C ∪ D is a union of two non-empty sets, C and D, of attributes. The elements in C are called conditional attributes. The elements in D are called decision attributes

Example

Suppose we are given an RI. Each component of RI is described below:

(1) implicit set of entities, U={ID-1, ID-2,..., ID-9}

(2) T= {Location, TEST, NEW, CASE, RESULT}

(3) dom(Location) = {Houston, San Jose, Palto Alto, Berkeley, New York, Atlanta, Chicago, Baltimore, Seattle}

dom(TEST) = {10, 11, 20}

dom(NEW) = {92, 90, 91, 93}

dom(CASE) = {03, 02, 04, 70}

dom(RESULT) ={10, 50, 99}

(4) a set of maps:

t_1(Location) = Houston, t_1(TEST) = 10, t_1(NEW) = 92, t_1(CASE) = 03, t_1(RESULT) = 10;

t_2(Location) = San Jose, t_2(TEST) = 10, t_2(NEW) = 92, t_2(CASE) = 03, t_2(RESULT) = 10.

In stead of listing every map, we use the following table (boxed portion) to represent all maps; each row represent one map

U	Location	TEST	NEW	CASE	RESULT
ID-1	Houston	10	92	03	10
ID-2	San Jose	10	92	03	10
ID-3	Palto Alto	10	90	02	10
ID-4	Berkeley	11	91	04	50
ID-5	NewYork	11	91	04	50
ID-6	Atlanta	20	93	70	99
ID-7	Chicago	20	93	70	99
ID-8	Baltimore	20	93	70	99
ID-9	Seattle	20	93	70	99

Following the proof of the proposition, we give each tuple a name, namely, NAME(t_j) =ID-j, and the collection of names U. They are illustrated in the first column (unboxed portion).

(1) U={ID-1, ID-2, ID-3, ID-4, ID-5, ID-6, ID-7, ID-8, ID-9}

(2) T= {Location, TEST, NEW, CASE, RESULT}

(3) dom(Location) = {Houston, San Jose, Palto Alto, Berkeley, New York, Atlanta, Chicago, Baltimore, Seattle}
dom(TEST) = {10, 11, 20}
dom(NEW) = {92, 90, 91, 93}
dom(CASE) = {03, 02, 04, 70}
dom(RESULT) ={10, 50, 99}

(4) following the proof of the proposition, the description function can be described as follows:

ρ: U x T → Dom = dom(TEST) $\cup$ dom(NEW) $\cup$ dom(CASE)$\cup$dom(RESULT)

ρ(ID-1, Location) = t_1(Location)=Houston $\in$ dom(Location)
ρ(ID-1, TEST) = t_1(TEST)=10$\in$ dom(TEST)
ρ(ID-1, NEW) = t_1(NEW)=10$\in$ dom(NEW)
ρ(ID-1, CASE) = t_1(CASE)=10$\in$ dom(CASE)
ρ(ID-1, RESULT) = t_1(RESULT)=10$\in$ dom(RESULT)

We illustrate ρ for the "portion of first row," rest of ρ can easily be read out from the table (unboxed column and boxed portion).

3. SECURITY CLASSIFICATION

This section is not part of the formal theory. Issues on security classification are discussed informally so that one can derive a *formal requirement.* In this paper, information tables(IT) which we have discussed in last section are used as Tarskian semantic models of decision logic (see Section 4). In this respect, IT is better than RI In IT formulation, the real world is an integral part of the formalism; there are labels for the real world, which make the correspondence between real world and mathematical model explicit. An IT is a mathematical model of a "slice" of the real world that we are interested in. From the prospect of formal logic IT is identified with the "slice" (Frost, 1986, Section 5.3). IT is the semantic model of the decision logic.

In a secure world, an entity or a sub-universe of IT has a security classification. So each formula should also have the security classification, since a logical formula is meant to describe a "sub-slice" of the real world. In this section, we will examine the interactions of these security classifications.

3.1. Formulas and their meanings

Let SC be a lattice (Birkoff, 1967); its element is called a security class, security level or simply label. To each formula φ, we associate a variable to hold a security class. If φ is interpreted, then its label C(φ) reflects some security semantics of the *meaning* |φ|; see Section 4.2 for formal definition of the term "meaning." First, we will explore the semantics of C(φ) and C(|φ|). To this end, let us consider a tuple t, representing the entity u,

$$t = (\rho(u, A_1), \rho(u, A_2), \ldots, \rho(u, A_n))$$

For simplicity, we will write $\rho(u, A_i)$ by the attribute pair $<A_i, v_i(u)>$, or simply $<A_i, v_i>$ when u is understood. So the entity u can be written as a tuple

$$t = (< A_1, v_1>, < A_2, v_2>, \ldots, < A_n, v_n>).$$

Or as a formula

$$\varphi_n = < A_1, v_1> \wedge < A_2, v_2> \wedge \ldots \wedge < A_n, v_n>.$$

The meaning $| \varphi_n |$ is a singleton {u}, we will simply write $| \varphi_n | = u$.

The meanings $|\varphi_i|$ of sub-formulas

$$\varphi_i = <A_1, v_1> \wedge <A_2, v_2> \wedge \ldots \wedge <A_i, v_i>, i = 1, 2, \ldots, n$$

form a nested sequence of sets

$$|<A_1, v_1>| \supseteq |<A_1, v_1> \wedge <A_2, v_2>| \supseteq \ldots. \\ \supseteq |<A_1, v_1> \wedge <A_2, v_2> \wedge, \ldots. \wedge <A_n, v_n>| = \{u\}.$$

Informally, each formula φ_i can be expressed as a tuple, so we have the following "reverse nesting:"

$$(<A_1, v_1>) \subseteq (<A_1, v_1>, <A_2, v_2>) \subseteq \ldots. \\ \subseteq (<A_1, v_1>, <A_2, v_2>, \ldots, <A_n, v_n>) = u\,.$$

Note that a tuple can be interpreted as a set that selects one element from each active domain (Maier, 1983); this sequence is nesting in this sense. Intuitively we can regard φ as the "name" of the set $|\varphi|$; The former sequence is a sequence of sets and the latter is a sequence of "names." Note that two sequences can be juxtaposition together and form a longer sequence if we identify u with {u}. So these analysis leads us to conclude that we need the following

$$C(\varphi) \leq C(u), u \in |\varphi|. \qquad \text{(Req 1)}$$

$$C(|\varphi|) \geq C(u), u \in |\varphi|. \qquad \text{(Req 2)}$$

(Req 1) *The security class of the name of a set is dominated by the security classes of its elements.*

(Req 2) *The security class of the set dominates the security classes of its elements.*

By similar reasons, we require,

$$C(|\varphi|) \geq C(|\eta|), \text{ if } |\varphi| \supseteq |\eta| \qquad \text{(Req 3)}$$

(Req 3) *The security class of a meaning dominates the security classes of its sub-meanings.*

So we have established three requirements for security classification.

Example Consider the following SQL

```
Select *
From IT
where (A1 =v1 ∧ A2= v2 ∧ ....∧ Ai,= vi)
```

The output is the set $|\varphi_i|$, the conditions in "where clause" is the formula φ_i. So the security classes of all output tuples dominate the security class of the condition. The condition uniquely determine the output, so it is the unique "name" of the output

3.2. Access control, lattice model and information granulation

In last sub-section, we have clarify the relationships between C(A) and C(name(A)), where A is the meaning of a formula. Now we would like to examine their access controls. In general,

$$C(A) \geq C(u) \geq C(name(A)),\ u \in A$$

C(A) and C(name(A) may not be equal. If C(A) strictly dominates C(u) $\forall$ $u \in A$ (and A is minimal), then A is called an *aggregate* (Lunt, 1989). The existence of aggregates makes the access control complicated. It has been discussed extensively by many authors (Hinke, 1988, Lunt, 1989, Lin, 1989, 1990, 1991) just to name a few. In this paper, we will assume such aggregates do not exit; such security model is called lattice model (Denning, 1976, Lin et al, 1990, Lin, 1991).

Lattice model
A security model is called a lattice model, if $C(A) = \mathrm{Sup}\ \{C(u): \forall\ u \in A\}$. If a person whose clearance dominates C(name(A)), he can examine the set A, however, he can examine only those members whose security classes are dominated by his clearance.

For example, Hughes Aircraft Co (HAC) is a defence contractor. It handles projects ranging from top secret to unclassified. Let A be the set of data (or documents) in HAC, then name(A) = HAC. In this case C(HAC), the security class of name(A), is unclassified according to our interpretation. So an unclassified person, say John, can work at Hughes Aircraft. However John may not access to all of A. On the other hand, a top secret person, say Peter, may also work at HAC. In this case, Peter is permitted to access all of A (provided that he also meets all the need-to-know conditions).

Information granulation
It is clear there are two primitive objects in MLS environment, namely, individual elements and aggregates; a user either can access it or not. In other words, the

universe U is decomposed into elements and aggregates by its security semantics. Such models are studied in (Lin, 1991). For logic formulation, we will come back in another paper.

4. MLS information tables and relation instances

Let SC be a partial ordered set of security classes. In an MLS environment, each object, such as an attribute, attribute value and entity, has been assigned a security class. If we replace objects in IT or RI by object-class pairs, the notions of IT and RI in Section 2 may be transformed into MLS IT and RI respectively. Note that we need to require the constraints, (Req 1), (Req 2) and (Req 3) stated in Section 3. We would like to remind the following implications:

(1) The security label of an attribute is dominated by the security label of its values:

$$C(A_i) \leq C(v) \text{ for all } v \in dom(A_i)$$

(2) The security label of an attribute-value pair is dominated by the security label of its entities:

$$C(v_i) \leq C(u), \text{ where } u \in U \text{ and } v_i = \rho(u, A_i) \in dom(A_i)$$

Let us re-iterate the principle behind these constraints. The security label of a "name" of a collection of objects is dominated by the security labels of these objects. Attribute A_i is the "name" of dom(A_i), so it is dominated by $C(v_i)$. v_i is the "name" of all those entity u whose A_i-component has common value $v_i = \rho(u, A_i)$ (or common A_i-properties), so $C(v_i) \leq C(u)$ for those u whose A_i-component is v_i.

5. MLS DECISION LOGIC

In this section, we will set up MLS decision logic. For details on single level version we refer readers to (Palwak, 1991).

5.1. The Syntax of a MLS Decision Logic

Alphabet

a) *SC– The security lattice; its element is called a security class, security level or simply label.*

b) T – The set of attribute names

c) V= $\cup$ dom (A) – The set of attribute values of A $\in$ T, called active domain of attribute A.(Maier, 1983)

d) Ξ={~, $\wedge$, $\vee$, $\rightarrow$, $\equiv$ }–The set of connectives (negation, and, or, implication, equivalence)

Formulas Ω

The smallest set satisfies the following:

a) Expressions of the form, attribute value pair < A, v> with label C (<A, v>) called atomic formulas, are formula of MLS DL-language for any A $\in$ T and v $\in$ dom(A).

b) To each formula φ in DL-language, *we associate a label, denoted by* C(φ), *to hold the security class of* φ.

c) If φ and η are formulas, so are ~φ, ($\varphi\wedge$ η), ($\varphi\vee$ η), (φ $\rightarrow$ η) and their labels are C(~φ)=C(φ), C(φ)$\wedge$C(η), C(φ)$\vee$C(η), ~C(φ)$\vee$C(η), where $\vee$ and $\wedge$ are lattice operations.

5.2. The Semantics of a MLS Decision Logic

A model of MLS decision logic is an MLS IT.

Interpretations at level C

Let U^C or simply U (when C is understood) be an *MLS IT at level C. It consists of all entities that are dominated by the security class C.* As usual at each level, we will denote u $\models_U$ φ or u $\models$ φ when U is understood, if an object u $\in$ U satisfies a formula φ in U. So we will say u $\models$ φ, iff

u $\models$<A, v> iff ρ(u, A) = v
u $\models$ ~φ iff non u $\models$ φ
u $\models$ ($\varphi\wedge$ η) iff u $\models$ φ and u $\models$ η
u $\models$ (φ $\vee$ η) iff u $\models$ φ or u $\models$ η

We have many usual formulas, such as
u $\models$ (φ $\rightarrow$ η) iff u $\models$ ~φ $\vee$ η

We associate the formula φ, the following set

$$|\varphi|_U = \{u : u \in U \text{ and } u \models_U \varphi\}.$$

It will be called the *meaning* of φ at level C. A formula is said to be *true* if $|\varphi|_U$ = U; φ is *logically equivalent* to η iff their meanings are the same, i.e., $|\varphi|_U = |\eta|_U$.

All formula and their meanings are properly classified. Note all U in this paragraph is actually U^C

Monotonic assumption
For simplicity, we will assume SC consists of two elements, L and H, read as low and high respectively. We will ignore polyinstantiation and assume $U^L \subseteq U^H$, where $L \leq H$.

5.3. The Deductive System of a MLS Decision Logic

Recall that at level C means all objects which are dominated by C.

Inference rules at Level C
Modus ponens is the only rule.

Axioms at level C
(1) The set of propositional tautologies
(2) Specific axioms:

(a) $<A, v> \wedge <A, u> \equiv 0$ for any $A \in T$ and $v, u \in V$ and $v \neq u$

(b)$\bigvee\{ <A, v> :$ for every $v \in \mathrm{dom}(A)$ and for every $A \in T\} \equiv 1$

(c)$\sim <A, v> \equiv \bigvee\{ < A, u> :$ for every $u \in \mathrm{dom}(A)$ and every $A \in T, \ v \neq u \}$

We need few auxiliary notations and results: Let 0 and 1 denote falsity and truth at security level C. From the monotonicity assumption, these 0 and 1 will behave consistently from level to level.

Formula of the form

$$< A_1, v_1> \wedge < A_2, v_2> \wedge, \ldots < A_n, v_n>$$

is called P-basic formula or P-formula, where $v_I \in$ dom (A_i), and P= $\{A_1, A_2, .. A_n \}$. For P = T, P-basic formulas will be called *basic formulas*. The set of all basic formulas satisfiable in U is called *basic knowledge* in U. The specific Axiom (a) follows from the assumption that each entity can have exact one value in each attribute. The Axiom (b) implies that each value of its domain must be taken once. This is saying that dom(A) is the active domain of attribute A. The Axiom (c) allows us to get rid of the negation in such a way that instead of saying that an object does not possesses a given property we can say that it has one of the remaining properties. It implies the closed word assumption. Let Σ_U (P), or simply

Σ (P) denote the disjunction of all P-basic formulas satisfied in U. The closed word assumption can be express in the following (Pawlak, 1991):

Proposition $\models_U \Sigma_U (P) \equiv 1$. For any $P \subseteq T$.

Note that all commercial DBMS have this assumption. For example, the output of not red colour consists of all non-red colours. A formula φ is a theorem, denoted by $\vdash \varphi$, if it is derivable from the axioms. At level C, the set of theorems of MLS DL-logic is *identical with the set of theorems of propositional calculus with specific axioms (a)- (c).*

6. INFERENCE AND DECISION RULES

In this section, we will discuss potential applications of MLS decision logic. Recall that "objects at level C" means "objects whose security classes are dominated by C."

Decision rules at level C

We will use U^C or simply U to denote the universe at level C (dominated by C). Any implication $(\varphi \rightarrow \eta)$ is called a decision rule. A decision rule is consistent in U if it is true in U (that is, $\models_U (\varphi \rightarrow \eta)$; otherwise, the decision rule is inconsistent.

Decision rule is a term used by the community of decision support systems. In database security community a decision rule is often referred to as an *inference rule* (Lunt, 1989, Lin, *1989, 1993).* We will use it, when there is no danger of confusing. It should be clear that such a inference rule is a formula, not the inference rules of a deductive system. A decision rule $(\varphi \rightarrow \eta)$ is called a PQ-basic rule, if φ and η are P-basic and Q-basic formulas respectively. A decision algorithm is defined to be a set of decision rules (Pawlak, 1991). If all decision rules are PQ-basic rules, then the algorithm is said to be PQ-algorithm. Note that common definition of an algorithm is a sequence, not a set. Let $P \subseteq A$ be a subset of attributes.

Proposition Let P and Q be two subsets of attributes in U. Then PQ-algorithm determine a decision table and vice versa.

Inference problems

By employing IT processing (rough set methodology), we can find a minimal decision PQ-algorithm at each level C. From such a minimal decision algorithm, we can detect all the potential formal inference channels (channels of formal reasoning). However, there are informal channels, such as plausible reasoning, human reasoning and etc.

Let us consider three formulas φ, η, and $\varphi \rightarrow \eta$. We would like to find the constraints among their security classes so that inference attacks can be avoided. Let p be a person who may access φ but *not* η. Then p should not be permitted to access $\varphi \rightarrow \eta$. The security clearance C(p) of should obey the following constraints:

$$C(p) \geq C(\varphi),\ \ C(p) < C(\varphi \rightarrow \eta), \text{and } C(p) < C(\eta).$$

In other words, if p is permitted to access some information about φ, then p should not be permitted to know anything about $\varphi \rightarrow \eta$. Partial knowledge about $|\varphi|$ and $|\varphi \rightarrow \eta|$ may result in some partial knowledge about $|\eta|$ which is not desirable. Any subset of $|\varphi \rightarrow \eta| = \sim|\varphi| \cup |\eta|$, whose security classification is dominated by C(p), should be empty. We will defer such analysis in future papers.

Robust rules and soft rules

The previous analysis is based on precise and exact analysis that are useful for small to median size of data. If the data is huge, we may want to do some filtering. One possible choice is to look only at rules that have appeared repeatedly (Agrawal et al, 1993, Lin, 1996, Lin and Chen, 1996, 1997). These are very robust rules. We should caution security officers that some of these robust rules are very often well known facts in the users' community. We may also want to mine approximate rules (Lin & Yao, 1996).

7. CONCLUSION

An information table that include a snap shot of a relational database can be viewed as a logic system. Note that such a system is not the so-called deductive database system. In this paper, we formalise these information tables or loosely these snap shots into MLS decision logic systems. Based on such logic systems, one can find all the formal inference rules of varying degree of robustness. This paper provides a model for a comprehensive and complete analysis of possible formal inference attacks. Further experimental works are needed to uncover practical effects or difficulties of such formal systems.

8. REFERENCES

Agrawal, R., Imielinski, T and Swami, A. (1993) "Mining Association Rules between Sets of Items in Large Databases." In Proceeding of ACM-SIGMOD international Conference on Management of Data, pp. 207-216, Washington, DC, June, 1993

Birkhoff, G.D. (1967) Lattice Theory, American Mathematical Society, Colloquium Publications, 1967

Denning, D. E. (1976) "A Lattice Model of Secure Information Flow", Communications of the ACM, Vol. 19, No. 5, May 1976, pp. 236 - 243.

Frost, R. (1986) Introduction to Knowledge Base Systems, Macmillan Publishing Company, New York, 1986.

Hinke, T. H. (1987) "Inference Aggregation Detection in Database Management Systems," Proceedings of 1988 IEEE Symposium on Security and Privacy, 1987.

Lin, T. Y. (1989) "Commutative Security Algebra and Aggregation", Research Direction in Database Security, II, Proceedings of the Second RADC Workshop on Database Security, December 22, 1989.

Lin, T. Y., Kerschberg, L. and Trueblood, R. (1990) "Security Algebra and Formal Models", *Database Security: Status and Prospects III*, IFIP-Transaction, edited by D. Spooner and C. E. Landwehr, North Holland, 1990, pp.75-96.

Lin, T. Y. (1990) "Probabilistic Measure on Aggregation," *Proceedings of the 6th Annual Computer Security Application Conference*, Tucson, Arizona, December 3-7, 1990, pp. 286-294.

Lin, T. Y. (1991) "Multilevel Database and Aggregated Security Algebra", *Database Security: Status and Prospects IV*, IFIP-Transaction, edited by S. Jajodia and C. E. Landwehr, North Holland, 1991, pp.325-348.

Lin, T. Y. (1993) "Inference Secure Multilevel Data Model", *Database Security: Status and Prospects VI*, IFIP-Transaction, edited by B. Thurasingham and C. E. Landwehr, North Holland, 1993, pp.317-332.

Lin, T. Y. (1996) "Rough Set Theory in Very Large Databases," *Symposium on Modelling, Analysis and Simulation*, CESA'96 IMACS Multi Conference (Computational Engineering in Systems Applications), Lille, France, July 9-12, 1996, Vol. 2 of 2, pp. 936-941.

Lin, T. Y. and Yao, Y. Y.(1996) "Mining Soft Rules Using Rough Sets and Neighbourhoods," *Symposium on Modelling, Analysis and Simulation*, CESA'96 IMACS Multiconference (Computational Engineering in Systems Applications), Lille, France, July 9-12, 1996, Vol. 2 of 2,pp.1095-1100.

Lin, T. Y. and Chen, R. (1996) "Supporting Rough Set Theory in Very Large Database Using ORACLE RDBMS," *Soft Computing in Intelligent Systems and Information processing Proceedings of 1996 Asian Fuzzy Systems Symposium,* Kenting, Taiwan, December 11-14, 1996, 332-337

Lin, T. Y. and Chen, R. (1997) "Finding Reducts in Very Large Databases," Proceedings of Joint Conference of Information Science, Research Triangle Park, North Carolina, March 1-5, 1997, pp. 350-352.

Lunt, T. F. (1989) "Aggregation and Inference: Facts and Fallacies," Proceedings of 1987 IEEE Symposium on Security and Privacy, 1989.

Maier, D. (1983) The Theory of Relational Databases, Computer Science Press, 1983.

Pawlak, Z. (1991) Lin, T. Y. and Chen, R. (1996) Rough sets. Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, 1991

9. BIOGRAPHY

Tsau Young (T. Y.) Lin received his Ph. D. from Yale University, and now is a Professor at the Department of Mathematics and Computer Science, San Jose State University, also a visiting scholar at BISC, University of California-Berkeley. He has authored or co-authored over hundred research articles. He is the founding president of international rough set society. He has served as the chairs, co-chairs, and members of program committees in many conferences, special sessions and workshops. He is a co-editor-in-chief, associate editor and member of editorial board of several international journals. His interests include approximation theory (in database retrievals and reasoning), data mining, data security, fuzzy sets, intelligent control, non-classical logic, Petri nets, and rough sets (alphabetical order).

Zuo Xiao-Ling is a Professor at the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He is the Editor-in-Cheif of the International Theoretical Computer Science Journal (Shanghai), Vice Chairman of Computer Science Education Research Association of China's Universities, Vice Chairman of China Discrete Mathematics Society, and the Chairman of the Committee of theory of Shanghai Computer Society. His interests include complexity theory, cryptography, discrete mathematics, fuzzy sets, genetic algorithm, neural networks, and rough sets (alphabetical order).