Fall 2012

Algorithms for solving the reducts problem in rough sets

Shuang Wang
San Jose State University

SJSU

Algorithms for solving the reducts problem in rough sets

Shuang Wang
12/18/2012

ACKNOWLEDGEMENTS

I would like to express gratitude to Dr. Sami Khuri for his clear guidance and invaluable insight throughout the project without which this project could never have been completed.
I would also like to thank my committee member Dr. Robert Chun and Dr. George Toderici for providing me with their valuable feedback.

Finally would like to thank my husband, Bin Xiao, and my parents, Yunhai Wang and Lanying Sha, for helping me with good suggestions and supporting me in countless ways. Last but not the least, I would like to thank my friends, Rachel and Annie for their continuous encouragement.

Abstract

This work deals with finding minimal reducts of decision table based on the rough sets theory. Its goal is to develop algorithms capable of finding such reducts. Two algorithms are presented in this report: the first based on Boolean reasoning function, the second based on Genetic Algorithm. Test results on real data are given and conclusions are made.

Keywords: the rough sets theory, decision table, reducts, Boolean reasoning function, Genetic algorithm

Contents

I. INTRODUCTION 1
II. Basic Concepts of RS 1
2.1 Information system 1
2.2 Reducts 3
III. Decision Tables 7
3.1 Decision rules 7
3.2 Quality measures for rules [2] 8
IV. Decision Algorithms 9
4.1 Boolean reasoning function method 9
4.2 Genetic Algorithm 13
V. System Design 22
5.1 Top level 22
5.2 Boolean function-based reduct algorithm design 26
5.3 Genetic algorithm design 34
VI. Tests and Results 41
Test I: LED7 digit[16] 42
VII. Conclusion 47
7.1 Complexity 47
7.2 Data mining and machine learning 47
VIII. Future work 47
Reference 48
Appendix: Source Code (C++) 50

List of Tables

Table II-1 An example of Information System 2

Table II-2 Patients with symptoms of cold 3

Table II-3 No Cold-symptom IS 6

Table IV-1 Information system derived from Table II-3. 10

Table IV-2 The discernibility matrix for the information table provided in Table IV-1 10

Table IV-3 Decision system T 11

Table IV-4 discernibility matrix of original Table IV-3. 12

Table IV-5 Rules quality measures 13

Table IV-6 The distinction table of Table IV-3 16

Table IV-7 A population with 4 strings 20

Table VI-1 Summary of testing result 41

Table VI-2 result of LED7digit 42

Table VI-3 result of Zoo 43

Table VI-4 result of Iris 43

Table VI-5 Result of Flare 44

Table VI-6 Result of Breast 45

Table VI-7 Result of Banana 45

Table VI-8 result of Appendicitis 46

Table VI-9 result of Wine 46

List of Figures

Figure II-1 Approximation of X 5

Figure II-2 Approximation of subsets 5

Figure III-1 Decision table structure 7

Figure IV-1 Key processes in GA 13

Figure IV-2 Solution representation of a reduct 14

Figure IV-3 Building a distinction table from a decision table 15

Figure IV-4 Roulette wheel 20

Figure IV-5 Genetic algorithm frame 22

Figure V-1 System top level 23

Figure V-2 High level of Boolean reasoning algorithm 27

Figure V-3 Calculate the indiscernibility relation 28

Figure V-4 Calculate the decision-relative discernibility matrix 29

Figure V-5 Calculate the decision-relative discernibility function 30

Figure V-6 Convert an OR-AND expression to an AND-OR expression 32

Figure V-7 Verify the decision-relative discernibility function 33

Figure V-8 Classify by using decision-relative discernibility function 34

Figure V-9 High level of genetic-algorithm based reduct 35

Figure V-10 Decision table calculation 36

Figure V-11 Generate first generation of population 37

Figure V-12 Evaluate fitness value 38

Figure V-13 Selection and crossover 39

Figure V-14 Mutation 40

I. INTRODUCTION

Rough Set (RS) Theory was proposed by Pawlak, Zdzisław in 1981. RS analyzes relationships among Equivalence Class Partitions (ECP) existing in an Information System (IS). Two core concepts are created, discernibility relation and approximation, to constitute the basis of the theory. Through RS theory, significant attributes are extracted and an approximate equivalent IS with smaller scale can be built. RS algorithms are the processes to find concise equivalent IS to the original one. The processes are known as finding reducts. One of the RS applications in the artificial intelligence (AI) field of computer science is to find reductions of decision tables. Through an RS-based machine learning algorithm, reducts as patterns of decision tables can construct classifiers to categorize new objects. However, finding minimal reductions of a decision table is an NP-hard problem [13]. In this paper, we implement two minimal reduction algorithms: the Boolean reasoning function and genetic algorithm. The former is accurate and complete and works for small problem instances only, whereas the latter is suboptimal and can handle large problem instances. Different data sets are tested with these two implementations.

II. Basic Concepts of RS

2.1 Information system

Objects and their attributes are represented in a table, each row of which is an object (e.g., a person, an observed object, a test) and each column describes an attribute with some values based on a certain measurement. The table is called an information system. Formally, an information system is a pair I = (U, A), where U and A are both non-empty, finite sets. U is a set of objects, called a universe. A is a set of attributes. $\forall a \in A, a: U \rightarrow V_a$, where V_a is is the set of all possible values of a. [1]

Example 2.1.1

Let us analyze website usage with smart phones of people in different age groups. Then U can be defined as a set of different age groups: U = {≤18, 19-24, 25-34, 35-44, 45-54, 55-64, ≥65}, the set of attributes can be defined as different smartphone services A = {voice, messaging, browsing, gaming, multimedia, maps, social networking}. V_a is the daily average minute use of each service. The information system is represented by Table II-1.

Table II-1 An example of Information System

Ages	voice	messaging	browsing	gaming	multimedia	maps	social networking
≤18	10	10	20	10	10	5	15
19-24	45	15	30	35	30	35	30
25-34	30	6	45	30	20	20	45
35-44	45	5	40	10	20	20	20
45-54	30	7	35	5	50	10	15
55-64	80	2	40	0	10	5	5
≥65	30	1	20	0	10	0	2

We find that different age groups may have the same value on some attribute(s). Age groups 19-24 and 45-54 have the same daily average usage in voice. Age groups 25-34 and 35-44 have the same usage in maps. According to a set of attribute(s), the universe of objects can be partitioned into different blocks. Objects in the same block cannot be differentiated.

2.2 Reducts

An information system built on real world data, always contains redundant information, such as repeated rows, indiscernible rows, dependable attributes (these can be induced from other attributes). To infer a more concise equivalent information system is very useful in practice. The equivalent information system has fewer rows and columns and can be considered a pattern of the original one. In RS theory, we call this pattern a reduct. The process to obtain reducts is based on two concepts: 1) indiscernibility relation, and 2) set of approximation. Indiscernibility relation resolves the row redundancy, whereas approximation handles column redundancy.

Indiscernibility relation. Indiscernibility relation is an equivalence relation. The equivalence relation is defined by a set of attributes. Formally, let I = (U, A) be an information system. $\forall \boldsymbol{B}, \boldsymbol{B} \subseteq \mathbf{A}$, defines an associated binary relation $\mathbf{I}(\mathbf{B})$ on $\mathbf{U}$: $\boldsymbol{I}(\boldsymbol{B}) = \{(\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{U^2} | \forall \boldsymbol{a} \in \boldsymbol{B}, \boldsymbol{a}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{y})\}$, which is called B-indiscernible. If $(\mathbf{x1}, \mathbf{y2}) \in \mathbf{I}(\mathbf{B})$, then we say x1 and x2 are indiscernible with respect to B. The equivalence class of the B-indiscernible are denoted by B(x) or $[\boldsymbol{x}]_B$.

Table II-2 Patients with symptoms of cold

Patient	Runny / Stuffy nose	Body ache / headache	fever	fatigue	cold
P1	yes	no	yes	slight	yes
P2	yes	yes	no	slight	yes
P3	yes	yes	yes	heavy	yes
P4	no	yes	no	normal	no
P5	no	no	yes	slight	no
P6	no	no	no	slight	no
P7	yes	yes	no	slight	no
P8	no	yes	no	normal	no

Example 2.2.1

The definition of indiscernibility relation can be illustrated in Table II-2: A group of patients with cold symptoms. Let B1 = {Runny/Stuffy nose, Body ache/headache}, then IND(B1) = {{P1}, {P2, P3,P7}, {P4, P8}, {P5, P6}}; Let B2 = {fever, fatigue}, then IND(B2) = {{P1,P5}, {P2, P6, P7}, {P3}, {P4, P8}}.

Approximation. The set approximation is a core concept of RS Theory. It is based on the indiscernibility relation. However, it is an exploration of the relationship among different partitions included in a same set of objects. Different indiscernibility relations belonging to an information system may result in different partitions of a universe. We can "imprecisely" represent one partition with another partition. Why do we study this correlation? Let us look at Table II-2.

Patients can be divided into two groups by the attribute Cold, namely the group catching cold and the group not catching cold. In addition, patients can be divided into a certain number of groups according to a set of attributes listed in Table II-2, (e.g., Runny/Stuffy Nose, Body Ache / Headache). If the former and latter can obtain the same partition of the universe, the doctor may just use the former to diagnose the patients as having cold or not. It is a natural way for a doctor to diagnose patients.

Formally, let I = (U, A) be an information system. There are two subsets of A (i.e, $X \subseteq A$, $B \subseteq A$). We describe X with blocks of I(B) B – lower approximation and B – upper approximation are defined, respectively, as follows:

$$\underline{B}(X) = \bigcup_{x \in U} \{B(x) : B(x) \subseteq X\}$$

$$\overline{B}(X) = \bigcup_{x \in U} \{B(x) : B(x) \cap X \neq \emptyset\}$$

B – boundary region of X is defined as: $BN_B(X) = \overline{B}(X) - \underline{B}(X)$.

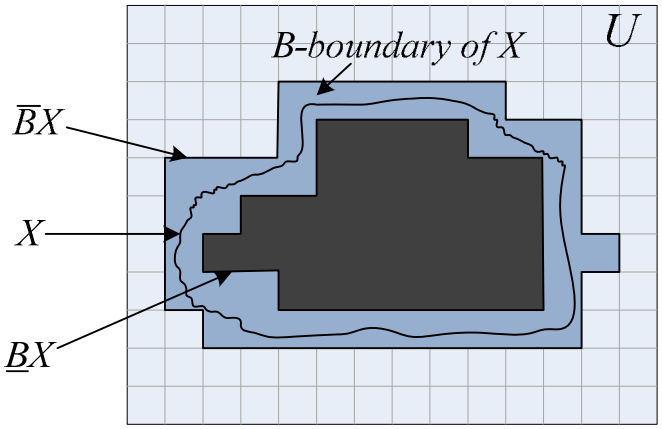


Figure II-1 Approximation of X

Example 2.2.2

Consider B1 and B2 as defined in Example 2.2.1. Represent B1- approximation and B2 – approximation of catching cold group (i.e., X1 = {P1, P2, P3}, non-catching cold group X2 = {P4, P5, P6, P7, P8} separately).

$\underline{B1}(X1) = \{P1\}, \overline{B1}(X1) = \{P1, P2, P3, P7\}$;

$\underline{B1}(X2) = \{P4, P5, P6, P8\}, \overline{B1}(X2) = \{P2, P3, P4, P5, P6, P7, P8\}$;

$\underline{B2}(X1) = \{P3\}, \overline{B2}(X1) = \{P1, P2, P3, P5, P6, P7\}$;

$\underline{B2}(X2) = \{P4, P8\}, \overline{B1}(X2) = \{P1, P2, P4, P5, P6, P7, P8\}$;

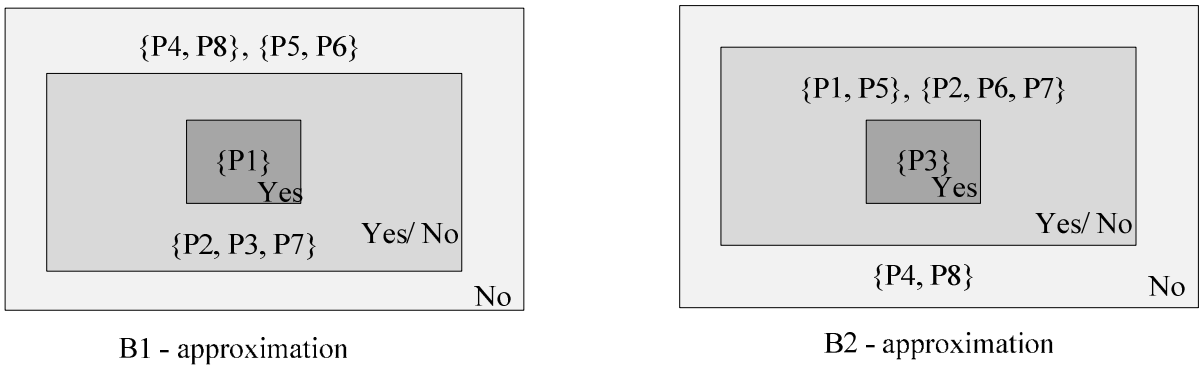


Figure II-2 Approximation of subsets

Reducts. Let I = (U, A) be an information system. A reduct is a minimal set of attributes B⊆A such that I(A)=I(B) [4]. Intuitively, a reduct of IS contains all of the concepts (categories) in the original one, but more concise with less number of rows and columns.

Example 2.2.3

Consider the following information system (removing cold attribute from Table II-2).

Table II-3 No Cold-symptom IS

Patient	Runny/Stuffy nose	Body ache / headache	fever	fatigue
P1	yes	no	yes	slight
P2	yes	yes	no	slight
P3	yes	yes	yes	heavy
P4	no	yes	no	normal
P5	no	no	yes	slight
P6	no	no	no	slight
P7	yes	yes	no	slight
P8	no	yes	no	normal

I = (U, {Runny/Stuffy Nose, Body Ache/ Headache, Fever, Fatigue}). We found that IND({Body Ache/Headache, Runny/Stuffy Nose, Fever}) partitioned patients the same way as {Runny/Stuffy Nose, Body Ache/ Headache, Fever, Fatigue}. The specific process regarding how to obtain the reducts of IS will be described later.

In the next section, we introduce a special important information system called decision tables.

III. Decision Tables

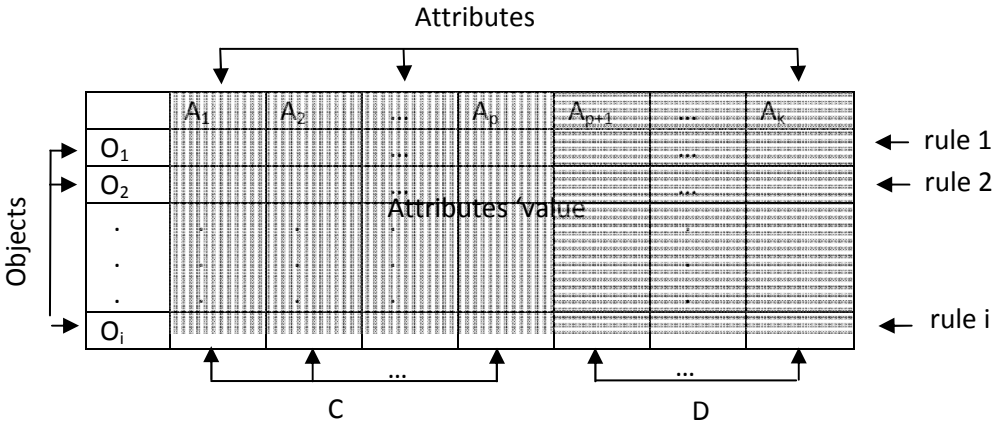


Figure III-1 Decision table structure

Just like Pawlak described in his book, a "decision table is a kind of prescription, which specifies what decisions (actions) should be undertaken when some conditions are satisfied" [1]. He also mentioned that a decision table is a useful tool for decision-making.

3.1 Decision rules

Let I = (U, A) be an information system. Let C, D $\subset$ A be called the condition and decision attributes separately. Every x $\in$ U determines a sequence $c_1(x), \ldots, c_n(x)$, $d_1(x), \ldots, d_2(x)$ called decision rules induced by x, where {c1, …, c2} = C and {d1, …, dm} = D. Therefore, a decision table can be denoted by T = (U, C, D), and C $\rightarrow_x$ D represents the decision rules induced by x [2].

Example 3.1.1

In Table II-2, condition attributes describing patients' symptoms are: Runny/Stuffy Nose, Body Ache / Headache, Fever, and Fatigue. The decision attribute is cold. Each row in Table II-2 determines a decision rule, for example row 3 determines the decision rule: "if patient has runny/ stuffy nose, body ache/ headache and fever, and feels heavily fatigued, he/she must catch a cold"

Does one rule uniquely define an object or how many objects share a rule? These questions can use the following parameters to measure.

3.2 Quality measures for rules [2]

Let T = (U, C, D) be a decision system. In the following definition, |X| denotes the cardinality of X.

Support of a rule $C \rightarrow_x D$ is denoted by $\mathrm{Supp}_x(C, D) = |C(x) \cap D(x)|$. This gives the number of objects that the same rule.

Strength of a rule $C \rightarrow_x D$ is denoted by $\sigma_x(C, D) = \frac{\mathrm{Supp}_x(C,D)}{|U|}$. This indicates how important the rule is according to its ratio in the decision table. Note that $0 < \sigma_x(C, D) < 1$.

The certainty factor of a rule $C \rightarrow_x D$ is denoted by $\mathrm{cer}_x(C, D) = \frac{|C(x) \cap D(x)|}{|C(x)|} = \frac{\mathrm{Supp}_x(C,D)}{|C(x)|}$. If $\mathrm{cer}_x(C, D) = 1$, then $C \rightarrow_x D$ will be called a certain decision rule; if $0 < \mathrm{cer}_x(C, D) < 1$ the decision rule will be referred to as the uncertain decision rule.

The coverage factor of rule $C \rightarrow_x D$, is defined as $\mathrm{cov}_x(C, D) = \frac{|C(x) \cap D(x)|}{|D(x)|} = \frac{\mathrm{Supp}_x(C,D)}{|C(x)|}$. This can be used as the reason for a decision [2].

Uncertain rules actually are counterpart elements that appear in the boundary region in Figure II-1.

Example 3.2.1

In Table II-2, if condition and decision attributes are the same as in Example 3.1.1, then IND(C) = {{P1}, {P2, P7}, {P3}, {P4, P8}, {P5}, {P6}}; IND(D) = {{P1, P2, P3}, {P4, P5, P6, P7, P8}}. Therefore, SuppP1(C, D) = SuppP3(C, D) = SuppP5(C, D) = SuppP6(C, D) = 1, SuppP2(C, D) = SuppP7(C, D) = SuppP4(C, D) = SuppP8(C, D) = 2. Accordingly, the strength of each rule σP1 = σP3 = σP5 = σP6 = $\frac{1}{8}$, σP2 = σP4 = σP7 = σP8 = $\frac{1}{4}$. cerP1(C, D) = cerP3(C,

D) = cerP5(C, D) = cerP6(C, D) = cerP4(C, D) = cerP8(C, D) = 1. cer_{P2}(C, D) = cer_{P7}(C, D) = $\frac{1}{2}$ (i.e., except for rule 2 and rule 7, all of the rules in Table II-2 are certain decision rules, according to the condition and decision attributes we defined in Example 3). Correspondingly, cov_{P1}(C, D) = cov_{P2}(C, D) = cov_{P3}(C, D) = $\frac{1}{3}$, cov_{P4}(C, D) = cov_{P8}(C, D) = $\frac{2}{5}$, cov_{P5}(C, D) = cov_{P6}(C, D) = cov_{P7}(C, D) = $\frac{1}{5}$.

In the next section, we discuss two findings of the minimum reducts of the decision table simply as decision algorithms.

IV. Decision Algorithms

4.1 Boolean reasoning function method

Essentially, decision rules are logic implications. Therefore, using the Boolean reasoning function to find the minimal reduct of a decision table is a reasonable process. We need the following concepts to construct a general method to compute reducts.

Definition 4.1.1 The discernibility matrix and discernibility function [3][14]

Let I = (U, A) be an information system with n objects. The discernibility matrix of I is a n × n symmetric matrix with elements $c_{ij} = \{a \in A \mid a(x_i) \neq a(x_j)\}$ for $i, j = 1, \dots, n$.

A discernibility function f_I for I is a propositional expression of m Boolean variables $a^*_1, \dots, a^*_m$(related to attributes $a_1, \dots, a_m$). $f_I(a^*_1, \dots, a^*_m) = \wedge \{\vee\ c^*_{ij} | 1 \leq j \leq i \leq n, c^*_{ij} \neq \emptyset\}$, where $c^*_{ij} = \{a^* | a \in c_{ij}\}$.

f_I actually records all discernible pairs from I, which represented in Boolean function called implicants The set of all minimal implicants called prime implicants of f_I determines the set of all reducts of I [15].

Example 4.1.1

We need to modify information system represented in Table II-3. Each row is one of equivalence classes based on all of the attributes (i.e., I = (U, A), where U = {[P1], [P2], [P3], [P4], [P5]}, A = {R, B, Fe, Fa}).

Table IV-1 Information system derived from Table II-3.

Eq. class	R/S nose(R)	B/H ache(B)	Fever(Fe)	Fatigue(Fa)
[P1]	yes	no	yes	s
[P2]	yes	yes	no	s
[P3]	yes	yes	yes	h
[P4]	no	yes	no	n
[P5]	no	no	yes	s
[P6]	no	no	no	s

Then we can obtain the discernibility matrix for the information table provided in Table IV-1 as shown in Table IV-2

Table IV-2 The discernibility matrix for the information table provided in Table IV-1

	[P1]	[P2]	[P3]	[P4]	[P5]	[P6]
[P1]	Ø					
[P2]	B, Fe	Ø				
[P3]	B, Fa	Fe, Fa	Ø			
[P4]	R, B, Fe, Fa	R, Fa	R, Fe, Fa	Ø		
[P5]	R	R, B, Fe	R, B, Fa	B, Fe, Fa	Ø	
[P6]	R, Fe	R, B	R, B, Fe, Fa	B, Fa	Fe	Ø

has the discernibility function below:

$f_I(R, B, Fe, Fa) \equiv (B \vee Fe) \wedge (B \vee Fa) \wedge (R \vee B \vee Fe \vee Fa) \wedge R \wedge (R \vee Fe)$

$\wedge (Fe \vee Fa) \wedge (R \vee Fa) \wedge (R \vee B \vee Fe) \wedge (R \vee B)$

$\wedge (R \vee Fe \vee Fa) \wedge (R \vee B \vee Fa) \wedge (R \vee B \vee Fe \vee Fa)$

$\wedge (B \vee Fe \vee Fa) \wedge (B \vee Fa)$

$\wedge Fe$

$\equiv (B \vee Fa) \wedge R \wedge Fe$

$\equiv (B \wedge R \wedge Fe) \vee (Fa \wedge R \wedge Fe)$

The primary implicants of $f_I(R, B, Fe, Fa)$ are $B \wedge R \wedge Fe$ or $Fa \wedge R \wedge Fe$. Therefore, there exist two reducts of I: {Body Ache/Headache, Runny/Stuffy Nose, Fever} and {Runny/Stuffy Nose, Fever, Fatigue}.

Another type of reduct is for decision systems. This extends the definition of a discernibility matrix and discernibility function from an information system.

Definition 4.1.2 The decision-relative discernibility matrix and decision-relative discernibility function.

Let T = (U, C, D) be a decision system. $M(I) = (c_{ij})$ is a dicernibilty matrix (Definition 4.1.1) of the corresponding information system I = (U, C ∪ D). $M^d(D) = (c^d{}_{ij})$ is decision-relative discernibility matrix, where

$$c^d{}_{ij} = \begin{cases} \emptyset, & \text{if } d(x_i) = d(x_j), \\ c_{ij} - \{d\}, & \text{otherwise} \end{cases}$$

The decision-relative discernibility function can be built from $M^d(D)$ in the same way as discernibility functions was built. (See Definition 4.1.1)

Table IV-3 Decision system T

Eq. class	R/S nose(R)	B/H ache(B)	Fever(Fe)	Fatigue(Fa)	Cold
[P1]	yes	no	yes	S	yes
[P2]	yes	yes	no	S	yes, no
[P3]	yes	yes	yes	H	yes
[P4]	no	yes	no	N	no
[P5]	no	no	yes	S	no
[P6]	no	no	no	S	no

Example 4.1.2 The construct decision-relative discernibility matrix of Table IV-3.

Table IV-4 discernibility matrix of original Table IV-3.

	[P1]	[P2]	[P3]	[P4]	[P5]	[P6]
[P1]	Ø					
[P2]	B, Fe	Ø				
[P3]	Ø	Fe, Fa	Ø			
[P4]	R, B, Fe, Fa	R, Fa	R, Fe, Fa	Ø		
[P5]	R	R, B, Fe	R, B, Fa	Ø	Ø	
[P6]	R, Fe	R, B	R, B, Fe, Fa	Ø	Ø	Ø

From the above decision-relative matrix, we can construct a decision-relative discernibility function of T by using a Boolean reasoning procedure similar to Example 4.1.1. Then we derive all primary applicants represented by $f^{d}M(D) = (R \wedge Fe) \vee (R \wedge B \wedge Fa)$ (*). $R \wedge Fe$ creates the following decision rules:

IF Running/Stuffy Nose = yes AND Fever = yes THEN Cold = yes;

IF Running/Stuffy Nose = yes AND Fever = no THEN Cold = yes OR Cold = no;

IF Running/Stuffy Nose = no AND Fever = no THEN Cold = no;

IF Running/Stuffy Nose = no AND Fever = yes THEN Cold = no;

Similarly, from the other implicant of (*): $R \wedge B \wedge Fa$ represents a reduct: {Running/Stuffy Nose, Body Ache/Headache, Fatigue} described in Table III-5, a series of rules can be made to decide the status of cold or not. In addition, we obtain the following decision rules and quality measures.

Table IV-5 Rules quality measures

Rule	Supp	Strength	Acc	Cov
IF Running/stuffy nose = yes AND fever = yes THEN cold = yes;	2	1/4	1	2/3
IF Running/stuffy nose = yes AND fever = no THEN cold = yes;	1	1/8	1/2	1/3
IF Running/stuffy nose = yes AND fever = no THEN cold = no;	1	1/8	1/2	1/5
IF Running/Stuffy Nose = no AND Fever = no THEN Cold = no;	3	3/8	1	3/5
IF Running/Stuffy Nose = no AND Fever = yes THEN Cold = no;	1	1/8	1	1/5

In the next part, we'll introduce another algorithm to finding the minimum reduct of decision table.

4.2 Genetic Algorithm

The genetic algorithm (GA), invented by Holland in 1975, was inspired by organism evolution. It is an adaptive heuristic search algorithm. It includes three basic processes shown in Figure IV-1, to mimic natural selection to obtain solutions to search problems.

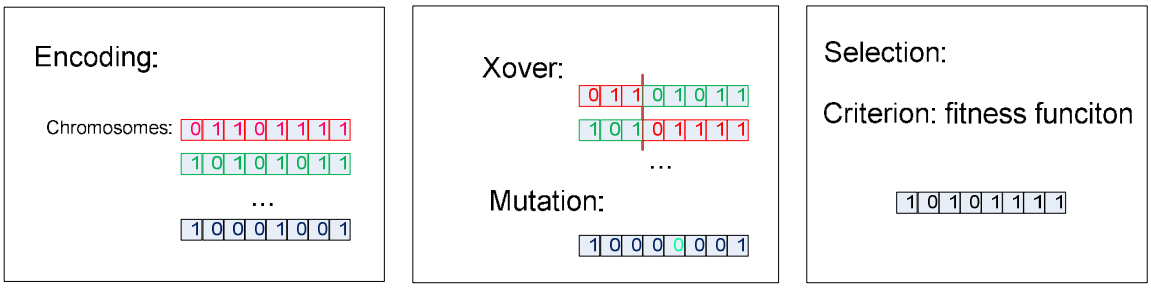


Figure IV-1 Key processes in GA

4.2.1 Encoding.

Solution Representation

In what follows we use a genetic algorithm to obtain a solution for the rough-set decision problem. First, we must find an encoding of potential solutions to the problem. This encoding must be meaningful to the problem and corresponding solution. Our problem is to find the minimum reduct of a decision table. What kind of strings can be represented as potential solutions? The natural way to describe the reduct is bitmap, which is shown in Figure IV-2. According to the reduct definition, the reduct representation is only a subset of attributes. Therefore, we create a binary string, the length of which is the same as the number of attributes in the decision table. Each bit of the string corresponds to one of the attribute of the decision table. If a certain bit is set to one, that means the corresponding attribute belongs to the reduct. It is possible that when the genetic algorithm ends, we may have multiple solutions because these strings have the same fitness score, which we will explain later.

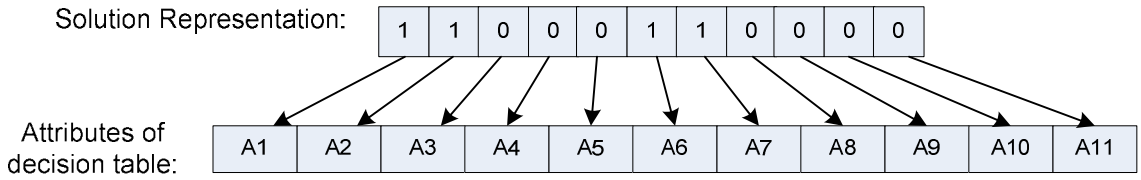


Figure IV-2 Solution representation of a reduct

The first population comes

Another question is how to get the first generation. Actually, it is problem-dependent. For the travelling salesman problem, for instance, we can create a first population from any permutation of city numbers as an individual, whereas, the current case is much more complicated. We first need to transform our decision table into a distinction table Figure IV-3

shows how to build up a distinction table. We randomly select a number of individuals from the distinction table. In addition, the distinction table is going to be involved in the fitness score computation for each individual. We represent the distinction table as a binary matrix called B [9]. Let b (i, (k,n)) be an element of B corresponding to the attribute I and pair (o_k, o_n):

For $i \in \{1, an\ N\}$:

$$b\big(i,(k,n)\big) = \begin{cases} 1, a_i(o_k) \neq a_i(o_n) \\ 0, a_i(o_k) = a_i(o_n) \end{cases}$$

$$b\big(N+1,(k,n)\big) = \begin{cases} 0, d_i(o_k) \neq d_i(o_n) \\ 1, d_i(o_k) = d_i(o_n) \end{cases} \quad (1)$$

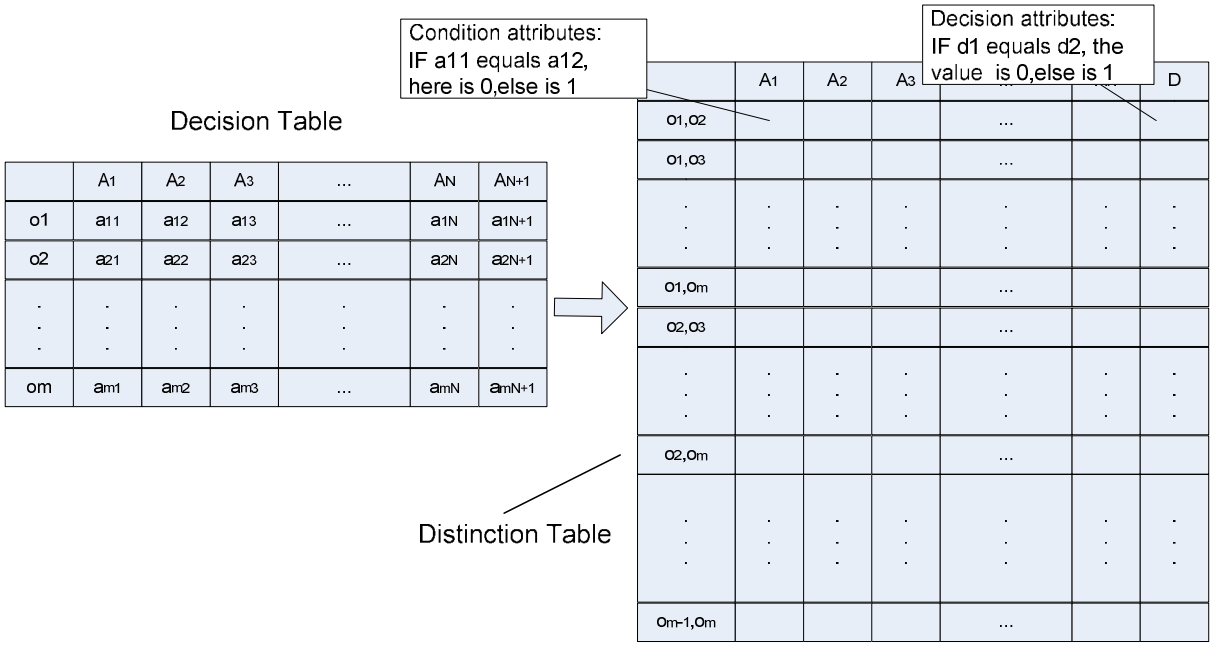


Figure IV-3 Building a distinction table from a decision table

Example 4.2.1 Build a distinction table from table IV-3.

Table IV-6 The distinction table of Table IV-3

Pairs	R/S nose(R)	B/H ache(B)	Fever(Fe)	Fatigue(Fa)	Cold
[P1],[P2]	0	1	1	0	1
[P1],[P3]	0	1	0	1	1
[P1],[P4]	1	1	1	1	0
[P1],[P5]	1	0	0	0	0
[P1],[P6]	1	0	1	0	0
[P2],[P3]	0	0	1	1	0
[P2],[P4]	1	0	0	1	0
[P2],[P5]	1	1	1	0	0
[P2],[P6]	1	1	0	0	0
[P3],[P4]	1	0	1	1	0
[P3],[P5]	1	1	0	1	0
[P3],[P6]	0	1	1	1	0
[P4],[P5]	0	1	1	1	1
[P4],[P6]	0	1	0	1	1
[P5],[P6]	0	0	1	1	1

The minimum reduct R is a subset of $\{A_1, A_2, \ldots A_N\}$, satisfying: $\forall(k, n)$,

$$\exists\, i \in R: b\big(i, (k, n)\big) = 1 \ \cup b\big(N + 1, (k, n)\big) = 1 \quad (2)$$

Later, we will see, from (1), it is not necessary to calculate all possible pairs to build a distinction table.

Once the distinction table is formed, we can select a certain number of strings (each row is a string) from it to build the first population of the GA.

4.2.2 Genetic operators.

Crossover

Crossover is actually a recombination of parents. It is the most important operation of the GA. New individuals generated by crossover are part of the next generation. Performing crossover is also problem-specific. Therefore, different types of crossover are crafted. For example, partially ordered xover (OX) is used in Travelling Salesman Problems (TSP), because there is a constraint that no city can be visited more than once. In our case, fortunately, there is no limitation at all. We just select two strings, and at some position, we recombine them, and two new offspring strings are born. Whether the offspring are part of the next generation (population) depends on their fitness score, which is explained in the selection section. However, one may ask for the significance of crossover performed on rows of the distinction table. For example, is it correct that the newly created strings have never appeared in the distinction table we just built? The right way to think about the meaning of a genetic operator is the objective for using a genetic algorithm. We want to find the minimum reduct of a decision table. After crossover, we create new strings with '1' and '0' scattered in different positions. Those '1's are what we care about because the corresponding attributes are included in the potential reduct for which we are searching. Whether it is one of the individual strings is not important any more. However, we need all of the individuals in the distinction table to measure whether the current new string is good enough or not. That is also the task of the fitness function.

Another important point to take into consideration is that Xover is carried out according to some percentage of the number of individuals in one population. We usually select 70% . Means for selecting the number of individuals performing crossover, are discussed later in this work.

Mutation

After understanding the meaning of the crossover operation, the mutation is much easier to understand. We implement the mutation by one bit flip. In other words, randomly select the position of the string and change it from 1 to 0 or vice versa. In addition, the mutation is performed according to some percentage, usually 5%, according to its biology meaning (mutations scarcely happen).

4.2.3 Selection.

Fitness function

It is very important to define a proper fitness function for the GA. The fitness function is used to assess the fitness of a string compared to the rest of the population. Each string (individual) gets its own fitness score computed from the fitness function. Individuals with the highest scores are selected or maintained after Xover and mutation. Finally, the GA stops after a certain number of generations, and the individual with the highest score in the last generation is the solution.

In our case [9], two parameters decide the fitness function: the number of "1"-s in the string r, named L_r and the number of covered rows of B, named C_r.

$$F(r) = \frac{N - L_r}{N} + \frac{C_r}{K} \quad (3)$$

where, $K = C_m^2 = \frac{m\,(m-1)}{2}$ (i.e. the number of rows in the distinction Table B). When $C_r = m$, a special bonus of 0.5 is added to the fitness score.

Example 4.2.2 Let's calculate the fitness value for the 10th row:

[P3],[P4]	1	0	1	1	0

in the Table IV-4 of Example 4.2.1. using (3), i.e., we compute the fitness value of string r: 10110. N = 5, L_r = 3; m = 6, K = $C_6^2 = 15$, C_r = 14. Bonus = $\left[\frac{14}{6}\right] \times 0.5 = 1.$

([a] means the integer part of a). Therefore, F(r) = $\frac{5-3}{3} + \frac{14}{15} + 1 = 2.60.$

As indicated in [9], calculation of C_r is the most time-consuming component when running the GA. Therefore, we use some techniques to make it faster. First, according to (2), $\forall k, n$, $b(i, (k, n)) = 1$ and $b(N + 1, (k, n)) = 0$ is what we really need to compute. That is what the rough-set based decision table is intended to do. Some conditional attributes must distinguish objects that belong to different decision classes. Second, we are following the suggestion found in [9] by which we save the bit string in the bitmap, and the bitwise operator such as AND or OR are conveniently available.

Selection

What kind of parent string is selected for crossover? Based on Darwin's evolutionary theory, the best ones should survive and can have offspring. There are too many methods used for the selection. Here we only introduce two general types of selection.

Roulette wheel selection

After we obtain the fitness score using Formula (3) for each string in the population, we can compute its relative fitness score [11]:

$$FN(x) = \frac{F(x)}{\sum F(x)} \quad (4)$$

We use the relative fitness score as a probability distribution and use this distribution to randomly select strings to perform crossover.

Example 4.2.3

Assume we have four strings then we can compute the fitness scores, according to Formula (3), the results are obtained in Table IV-7.

Table IV-7 A population with 4 strings

Strings	Fitness	Relative Fitness
1101110	31.5	.14
0110000	82.3	.38
1001100	60.1	.28
1100001	44.0	.20

We randomly select four strings from the string pool in Table IV-7, i.e., turning the roulette wheel 4 times,: 0110000, 1100001, 0110000, 1101110 are selected separately. The string 1001100 has not been selected. After this selection, we need to randomly pick pairs for crossover. Although the roulette wheel method has a natural selection character, there are two problems with it [12]. First: Degeneration sometimes happens. For example, it might happen that the best string is not selected. Second: Low efficiency is possible because strings with different scores distribute very unevenly. For instance, if the best-scoring string is larger than 90% of the entire roulette wheel then the other strings have no chance to be selected. The former will directly slower GA's convergence. We used the Elitism Selection method.

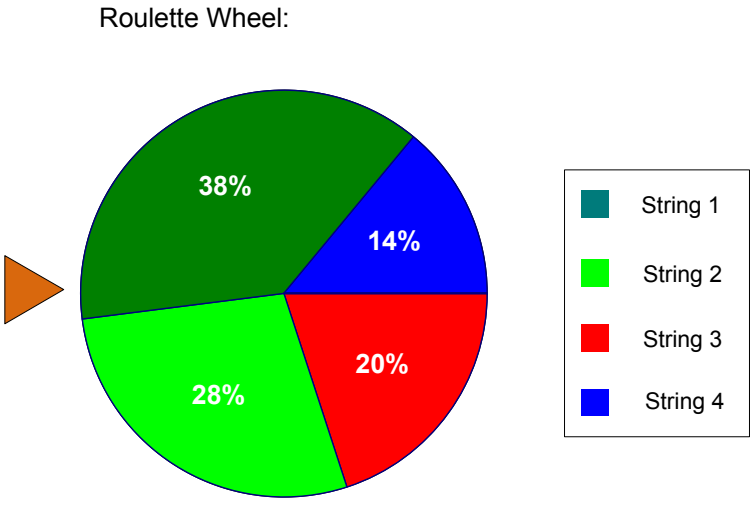


Figure IV-4 Roulette wheel

Elitism Selection

Just as the name implies, best strings are copied to the new generation. The rest of the population is processed in the usual way. Elitism can rapidly converge to a suboptimal solution because it avoids best string loss [12]. Our GA is implemented using the elitism method.

4.2.4 GA structure.

The GA can be depicted by the flowchart of Figure IV-5.There is always a hope that the new generation (population) is better than the previous one. However, it is not guaranteed that we can obtain the optimal solution when GA terminates. We know that we use GA to obtain a suboptimal solution to the NP-hard problem because the NP-hard problem cannot find a polynomial algorithm solution. Therefore, the termination condition is defined by the GA programmer. There are two kinds of termination. First, designate the number of the population, for instance, 2000. Once the GA program finishes generating the 2000th generation of the population, the best string is the solution. Second, if the best string is the same in five consecutive generations, the GA terminates. Both methods have advantages and disadvantages. It depends on the specific problem.

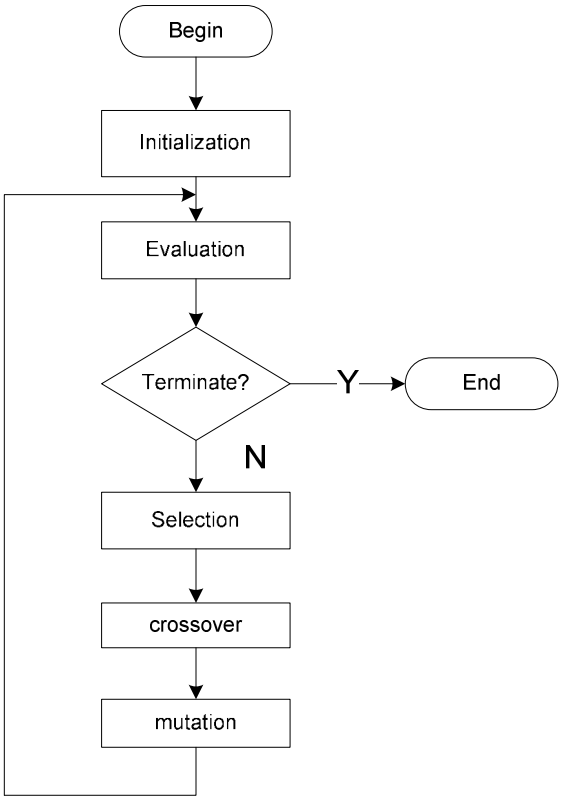


Figure IV-5 Genetic algorithm flowchart

In the next chapter, we'll design two above algorithm implementations in detail.

V. System Design

In this section, we will discuss the implementation of the algorithms described in Chapter 4. An algorithm based on Boolean reasoning will be presented first, followed by a genetic algorithm. To have a high performance system, we implement these two algorithms in C++.

5.1 Top level

The flow chart of the top level of our program is shown in Figure V-1. The program reads a file name from the command line and then reads training data and testing data from the designated file. The data will be converted to an internal data structure, which is the input of the Boolean reasoning algorithm and the genetic algorithm. The two algorithms can both use the decision-relative discernibility function (shortened as decision function in our program), and then

use that function and the training data to classify the testing data. By comparing the classifying results of these two algorithms, we can judge their accuracy. This will be discussed later.

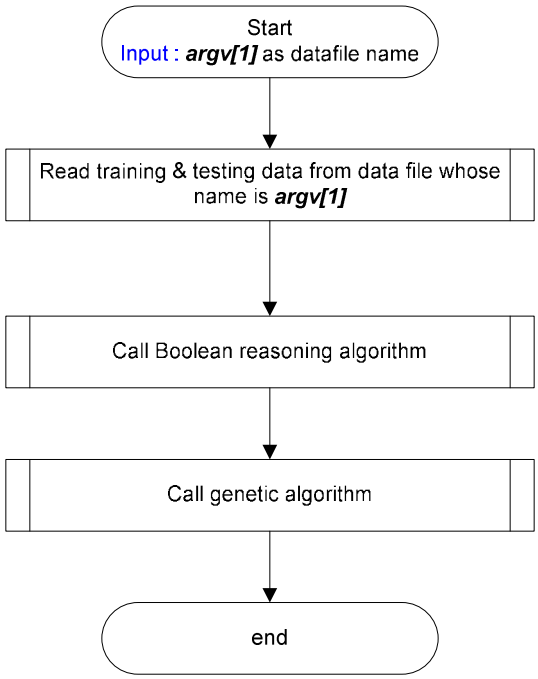


Figure V-1 System top level

Input file format. To support different structures of training and testing data flexibly and to provide some parameters to control the execution of the algorithm, a header is prefixed to the real data. The format of our data file can be defined with Backus Normal Form (BNF) as follows:

<data_file> ::= <header><WS><data-section>

<header> ::= <data-structure-definition><WS> <sub-attribute-definition><WS> <decision-attribute-definition><WS>

<max-decision-function-factors-definition><WS> < verification-flag>

< population-amount-definition>

<data-structure-definition> ::= “struct { “ <field-definition-list> “ } “

<field-definition-list> ::= <field-definition> | < field-definition-list> <field-definition>

<field-definition> ::= <range-field> | < non-range-field>

<range-field> ::= <range-type> <WS> < field-name > <WS> <lower-delta><WS><upper-delta> " ;"

<range-type>:= "int"|"short"|"float"|"double"

<field-name> ::= <identifier>

<lower-delta>::= <double>

<upper-delta>::=<double>

<non-range-field> ::= <non-range-type> <WS> <field-name>" ;"

<non-range-type>::="char"|"string"

<sub-attribute-definition> ::= "subAttributes { " <sub-attribute-list-list> " }"

<sub-attribute-list-list> ::= <sub-attribute-list> | <sub-attribute-list-list><WS><sub-attribute-list>

<sub-attribute-list>::="{ " <field-name-list> " } "

<field-name-list> ::= <field-name> | <field-name-list>" , " <field-name>

<decision-attribute-definition> ::= "DecisionAttr = "<field-name>

<max-decision-function-factors-definition> ::=" MaxDecisionFuncORItems = "<integer>

< verification-flag> ::= "DoVerification = " <flag-option>

<flag-option> ::="yes"|"no"

< population-amount-definition>::="PopuNum = "<integer>

<data-section> ::= <training-data> | <training-data><classify-flag> < testing-data>

<training-data> ::= < record-num-definition> <data-body>

<classify-flag>::="DoClassify = "<flag-option>

<testing-data> ::= < record-num-definition> < data-body >

<record-num-definition> ::= “recordNum = “ <integer>

<data-body > ::= <record > | < data-body><record>

<record> ::= <field-value> | <record> <WS> <field-value>

<field-value> ::= <integer> | <string> | <double> | <char> | <float>| <short>

<WS>::= <WSS>|<WS><WSS>

<WSS>::=’ ‘|’\t’|’\n’

The definitions of non-terminal integer, string, double, char, float, and short are the same as in the C/C++ language, so we do not provide them in detail here. To make it easier to understand, we give one example file format:

```
struct { string recName ;
      double RI -0.1 0.1 ;
      double Na -0.1 0.1 ;
      double Mg -0.1 0.1 ;
      double Al -0.1 0.1 ;
      double Si -0.1 0.1 ;
      double K -0.1 0.1 ;
      double Ca -0.1 0.1 ;
      double Ba -0.1 0.1 ;
      double Fe -0.1 0.1 ;
      int TypeGlass 0 0 ;
 }
subAttributes {
      { RI , Na , Mg , Al , Si , K , Ca , Ba , Fe } }

DecisionAttr = TypeGlass
```

```
MaxDecisionFuncORItems = 2000000
DoVerification = yes
PopuNum = 10

recordNum = 171
P001 1.51589 12.878 3.43036 1.40066 73.282 0.68931 8.04468 0 0.1224 1
P002 1.51764 12.9777 3.53812 1.21127 73.002 0.65205 8.52888 0 0 1
… … other 169 records of training data … …

DoClassify = yes
recordNum = 43

P01 1.52222 13.2105 3.7716 0.79076 71.9884 0.13041 10.2452 0 0 1
P02 1.51755 13.39 3.65935 1.1888 72.7892 0.57132 8.27064 0 0.0561 1
… … other 41 records of testing data … …
```

Each line of data, which is named as an object in RS Theory can be thought of as a record in a database. Therefore, in our program we use the term ‘record’ to avoid confusing it with the word ‘object’ as in the term “object-oriented programming,” which is more popular among modern programmers.

In the next part, we’ll show our Boolean reasoning function design in detail.

5.2 Boolean function-based reduct algorithm design

Boolean function-based reduct algorithm follows the procedure described in Section 4.1. First, we obtain an indiscernibility relation from the training data and then obtain a decision-relative discernibility matrix from the indiscernibility relation. Then we get a decision-relative discernibility function from the decision-relative discernibility matrix. Actually, the decision-

relative discernibility function we obtain directly from the decision-relative discernibility matrix is represented in OR-AND-logic expression. To classify as shown in Example 4.1.1 and 4.1.2, we need to convert the OR-AND-expression to AND-OR-expression by using the Boolean reasoning approach.

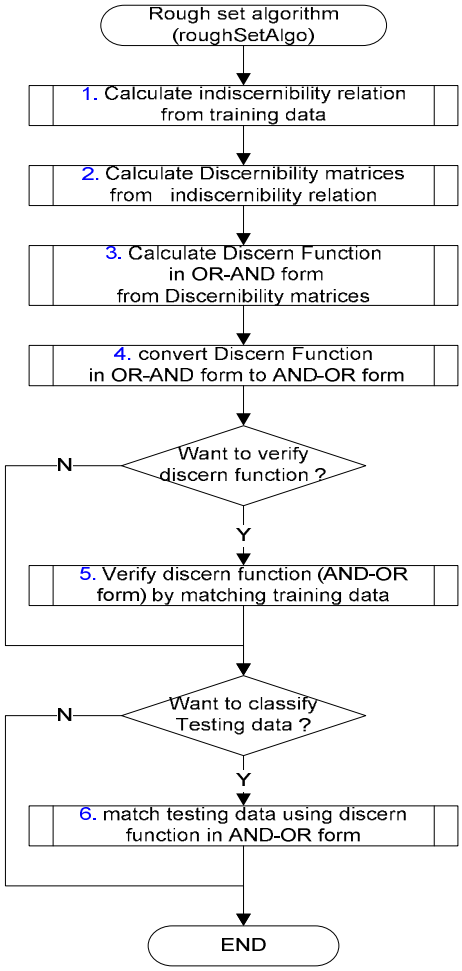


Figure V-2 High level of Boolean reasoning algorithm

After we obtain our AND-OR-expression, we can use it to classify the testing data if the user sets the "DoClassify" flag to yes in the input data file. By setting the "DoVerification" flag to yes, the user can also ask the program to verify the expression by matching training data to itself. The high-level flowchart of this algorithm is shown in Figure V-2. The detail of each step is shown from Figure V-3 to V-8.

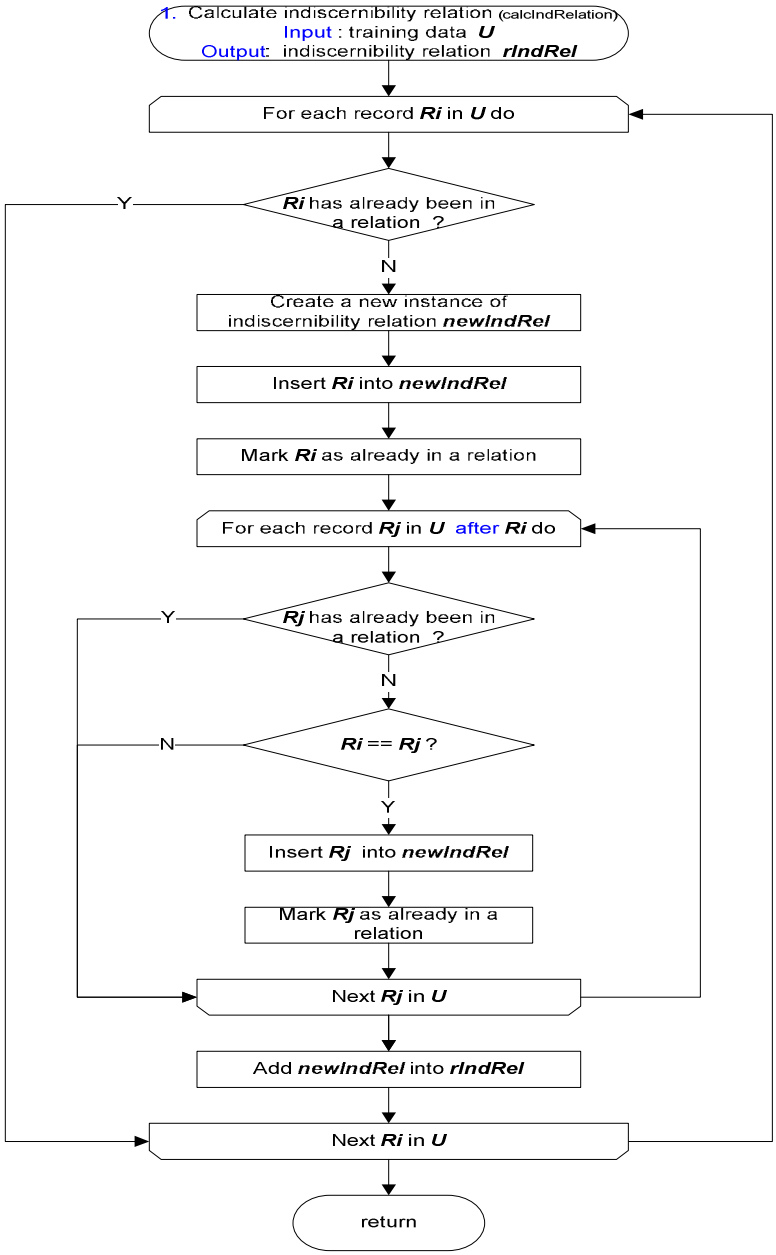


Figure V-3 Calculate the indiscernibility relation

Computation of indiscernibility relation. As shown in Figure V-3, to obtain the indiscernibility relations from the training data, we compare each record Ri to each other record Rj, where Ri and Rj in training data do not belong to any indiscernibility relation yet and j >i. If the two records are equal or compatible, we put them in a same indiscernibility relation set.

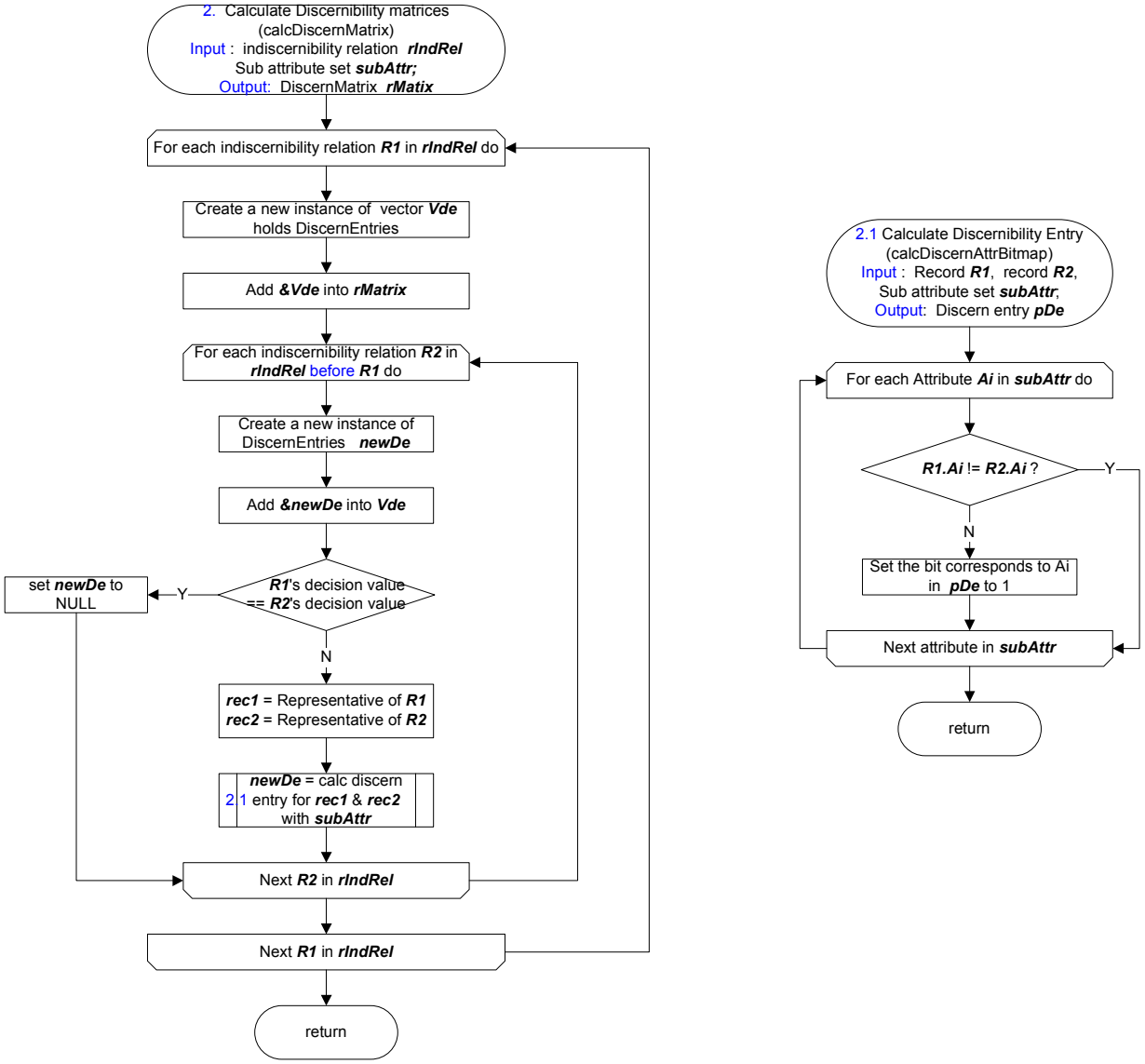
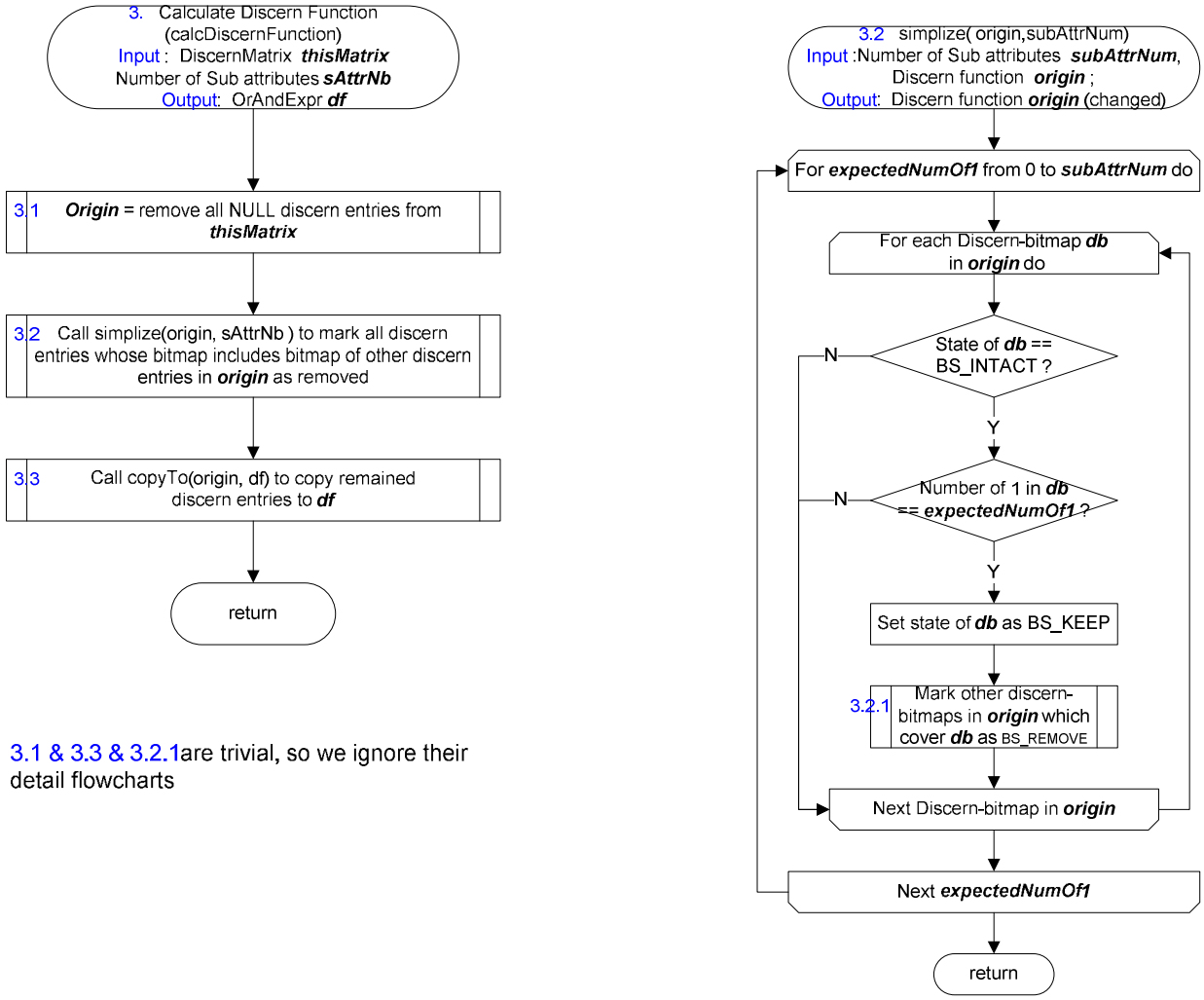


Figure V-4 Calculate the decision-relative discernibility matrix

Calculation of the decision-relative discernibility matrix. As shown in Figure V-4, the discernibility matrix is obtained by comparing each relation with each other. If the representative records of these two relations are not equal, we look for the attributes for which they differ. As with the genetic algorithm, the Boolean reasoning algorithm also uses a bitmap to encode the information. So element M[i,j] of the discernibility matrix M is actually a bitmap, the set bit of the attributes that differentiate between the two indiscernibility relations Ri and Rj.



3.1 & 3.3 & 3.2.1are trivial, so we ignore their detail flowcharts

Figure V-5 Calculate the decision-relative discernibility function

Calculation the decision-relative discernibility function. We can obtain the discernibility function by ANDing all non-empty bitmaps in the discernibility matrix. However, sometimes there are too many bitmaps in the matrix. Fortunately, a method has been proposed to simplify this function by removing the bitmap that includes other bitmaps. Figure V-5 shows this procedure.

Converting the OR-AND expression to an AND-OR expression. As mentioned earlier, the decision-relative discernibility function we obtain directly from the decision-relative discernibility matrix is a logic expression represented in OR-AND form. To classify this we need

an AND-OR-expression. Figure V-6 shows the procedure to convert the OR-AND expression to an AND-OR expression. We use a recursive function to perform this conversion. The main idea of the recursion can be explained by the following example.

Example 5.2.1 Suppose we have an OR-AND-expression (X_{11} OR X_{12}) AND (X_{21} OR X_{22} OR X_{23}) AND (X_{31} OR X_{32}) which can be divided into 3 OR expressions OEi (i=1,2,3) , where OE1 = X_{11} OR X_{12}, OE2= X_{21} OR X_{22} OR X_{23}, OE3 = X_{31} OR X_{32}.

When entered, each level l of recursion selects one factor f_l from OE_l in order X_{l1}, X_{l2},… . After level l selects out one f_l, the recursion will enter next level (level l+1) to select f_{l+1}. Since l = 4, in this example, means we have processed all OR expressions. At this point we obtain one AND expression which is f_1 AND f_2 AND f_3. After that we backtrack to level 3 to continue selecting the next factor from OE3, and then enter level 4 to construct a new AND expression. Since level 3 runs out of its factors, we backtrack to level 2 to select the next factor from OE2, and then recursively enter level 3 which selects a factor from the beginning, which is X_{31}. Repeat this procedure until level 1 runs out of its factors, and we obtain all AND expressions. For this example, when the first time l = 4, we can get AND expression X_{11} AND X_{21} AND X_{31}. The second time, when l= 4, we can get X_{11} AND X_{21} AND X_{32}. At this point level 3 runs out of its factors. After trackbacking to level2 and re-entering this level, it will select factor from beginning, so we get X_{11} AND X_{22} AND X_{31}. In this way, we can get all AND expressions converted from the original OR-AND expression.

Figure V-6 shows this procedure.

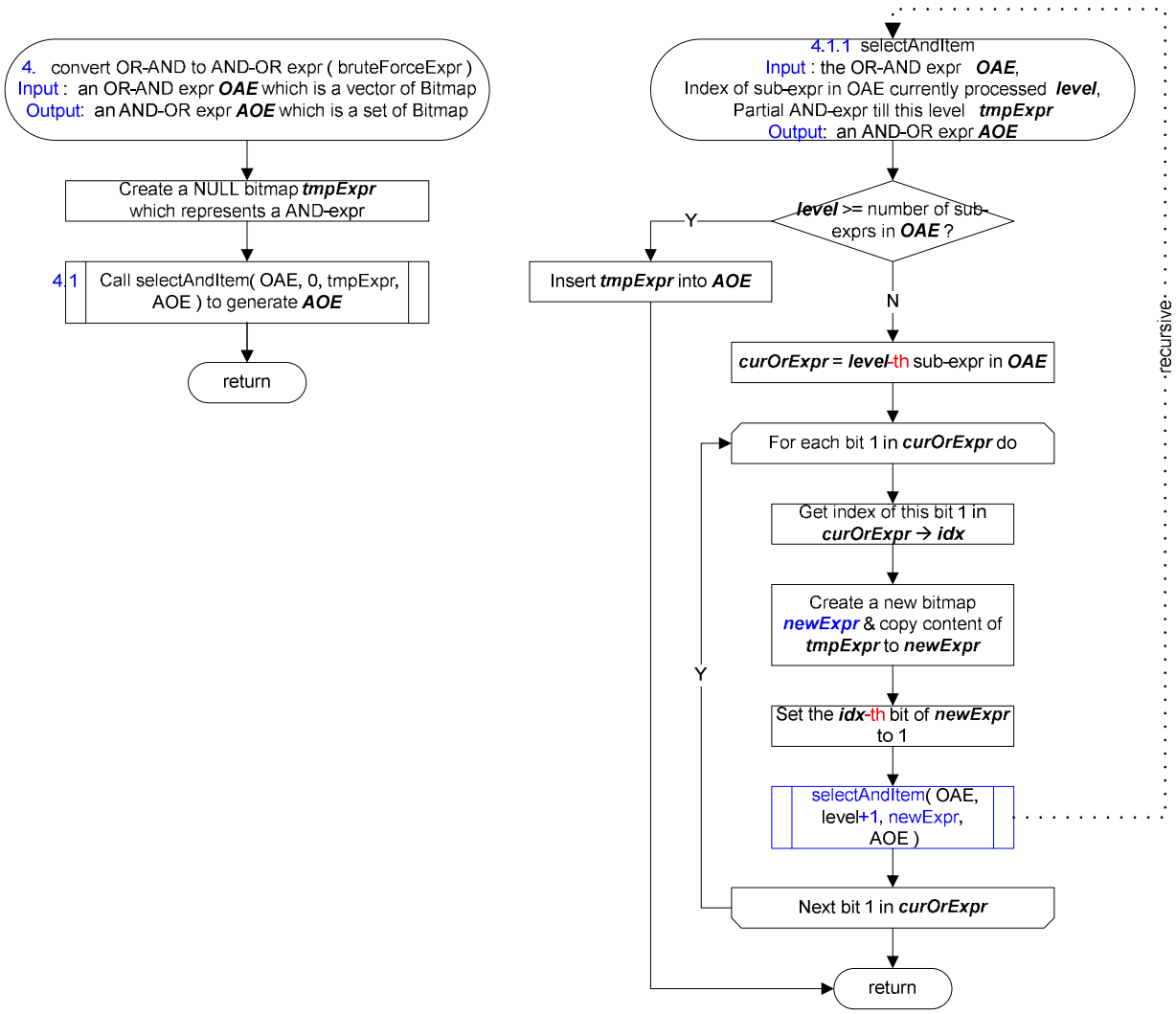


Figure V-6 Convert an OR-AND expression to an AND-OR expression

Verification and classification. Figure V-7 and V-8 show the procedure of Verification and classification respectively. Verification can be considered as classifying objects from training data, so we can discuss these two actions together. So far, we have a decision-relative discernibility function in AND-OR form, each factor of which is called a reduct in Example 4.1.1 and 4.1.2. A reduct is actually a subset of attributes ANDed together. Therefore, the procedure of classification is comparing each record Ri in the testing data to each record Rj in the training

data one by one. When we compare Ri and Rj, we only consider the attributes in the subset SA defined by a reduct. If Ri.Ai= Rj.Aj holds for all Ai belonging to SA, then we say we found a conditional match for Ri.

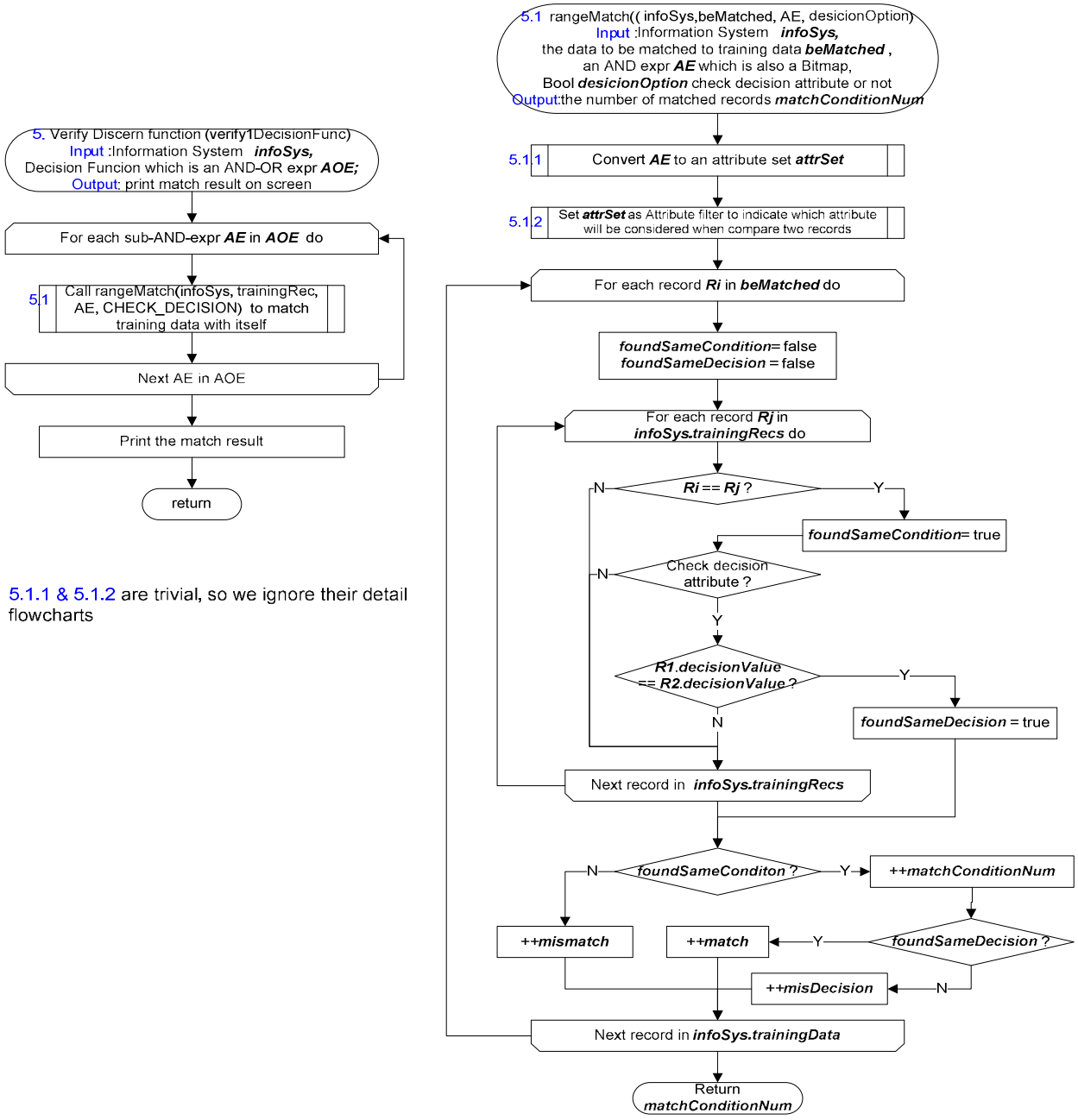


Figure V-7 Verify the decision-relative discernibility function

For each reduct, we use variable matchConditionNum to count the number of Ri that can find its

conditional match. The reduct with maxium matchConditionNum value is the final optimal reduct obtained by Boolean function reduct algorithm. Verification is similar to classification except for the record Ri is also from training data.

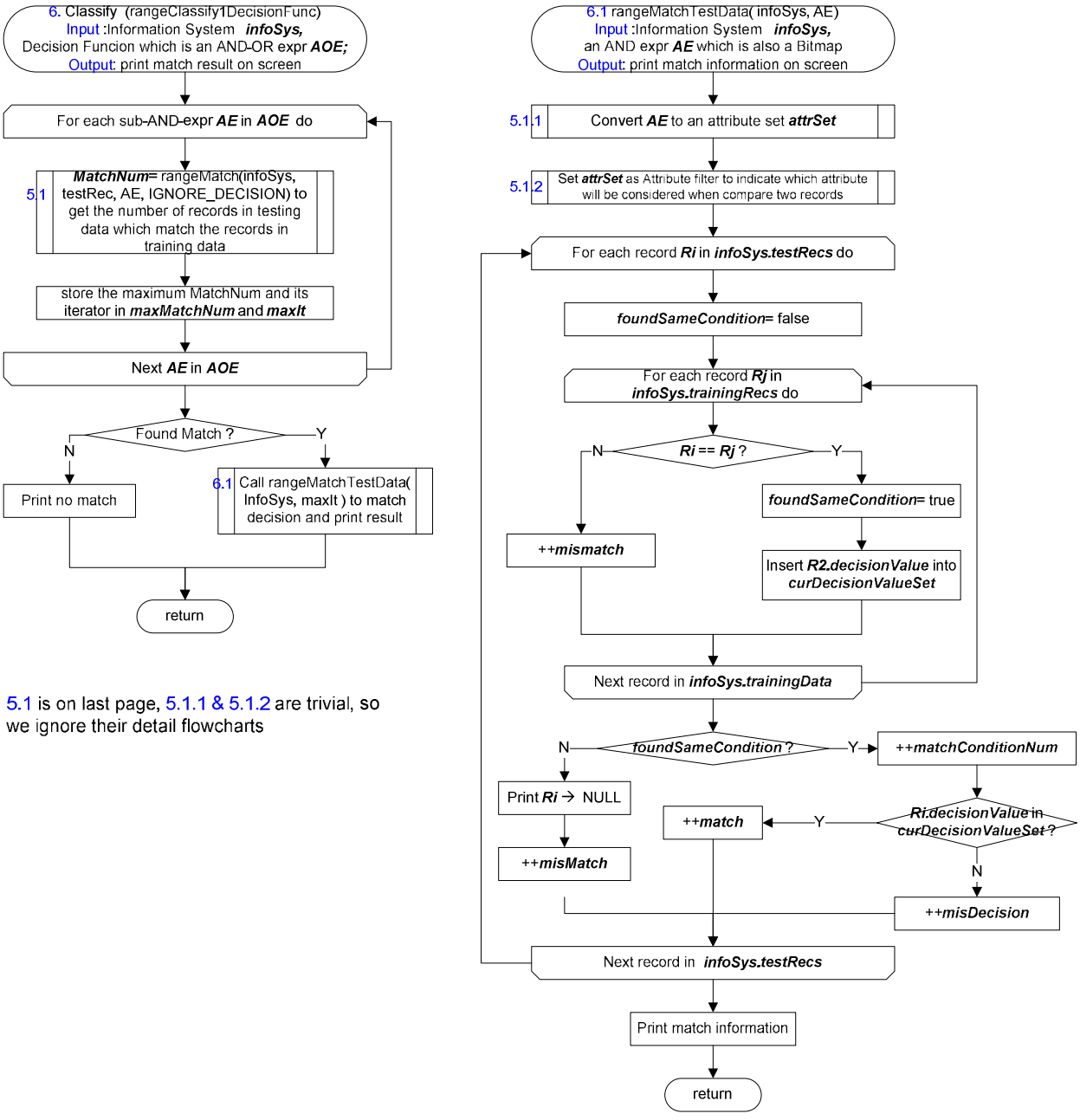


Figure V-8 Classify by using decision-relative discernibility function

In the next part, we'll show our genetic algorithm design in detail.

5.3 Genetic algorithm design

The implementation of Genetic algorithm reduct follows the procedure described in Section 4.2. The detail is shown in Figure V-9 to V-15.

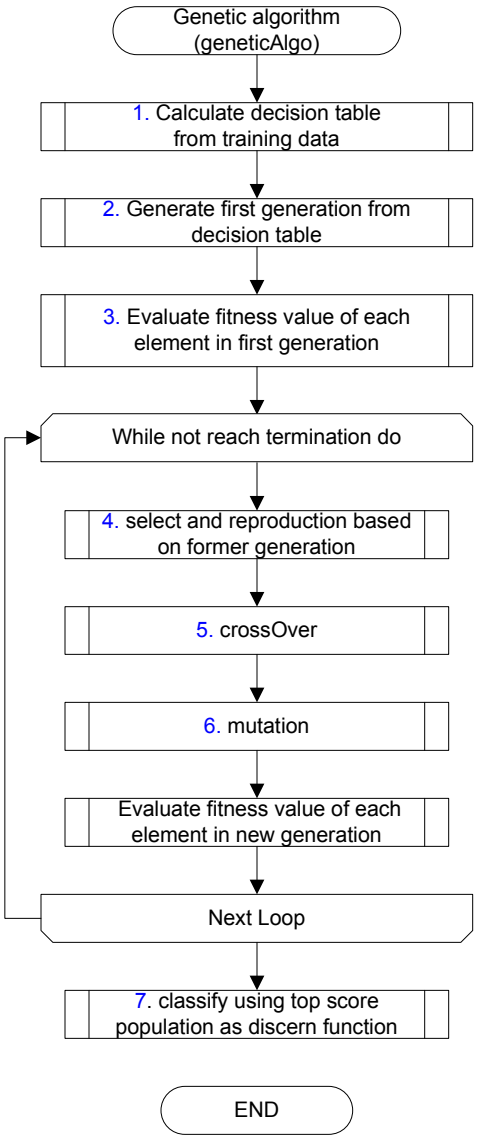


Figure V-9 High level of genetic-algorithm based reduct

Figure V-9, which is similar to the frame illustrated in Figure IV-5, gives the high level of the implementation of the genetic-algorithm reduct. In addition to the loop from evaluation to

mutation, we expand the initialization procedure in Figure IV-5 to calculating decision table and generating 1st generation. We also include classify function, which is same as in Boolean function-based reduct algorithm, so that we can compare the result of these two algorithms.

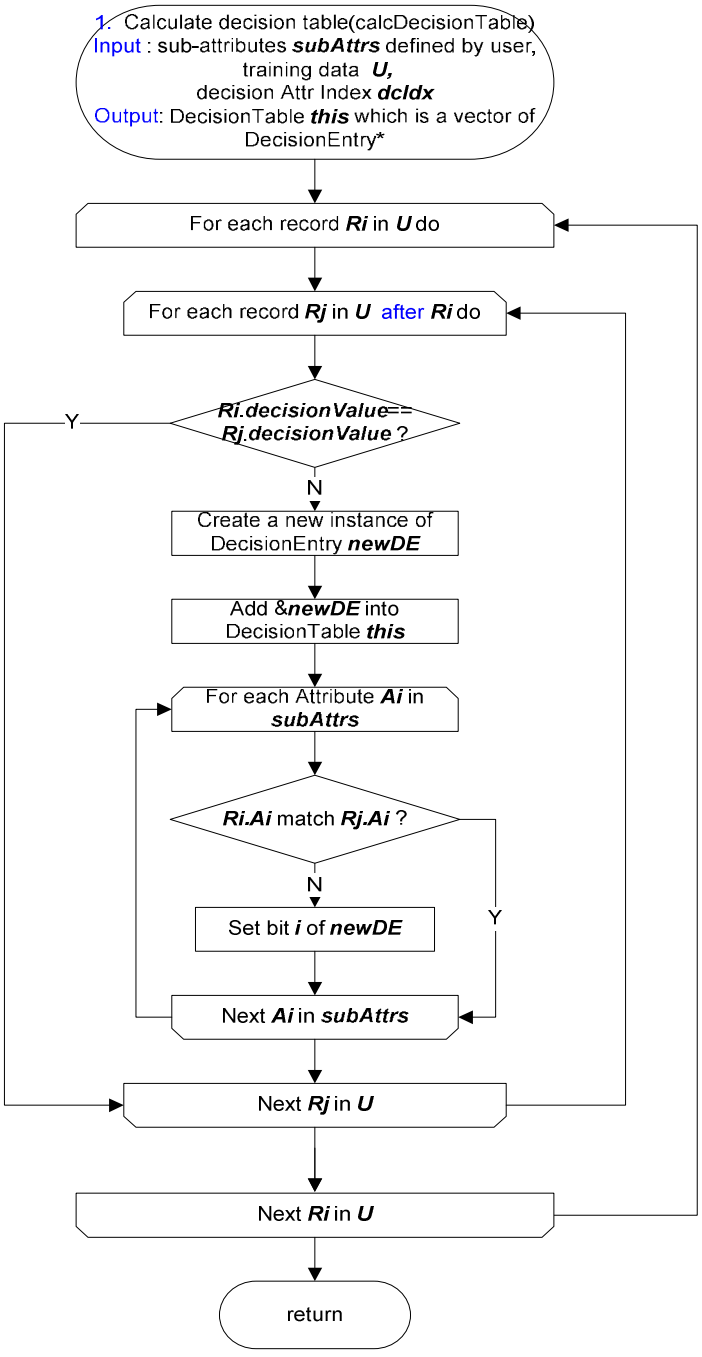


Figure V-10 Decision table calculation

Figure V-10 shows the procedure of calculating Decision table. We compare each record Ri to each other record Rj, where Ri and Rj in training data and j>i. If the decision values of the two records are different, we compare the values of their condition attributes further. If the values of certain condition attribute of these two records are different, we set the corresponding bit of a decision entry to 1. All the decision entries obtained from these n*(n-1)/2 comparisons make up the decision table.

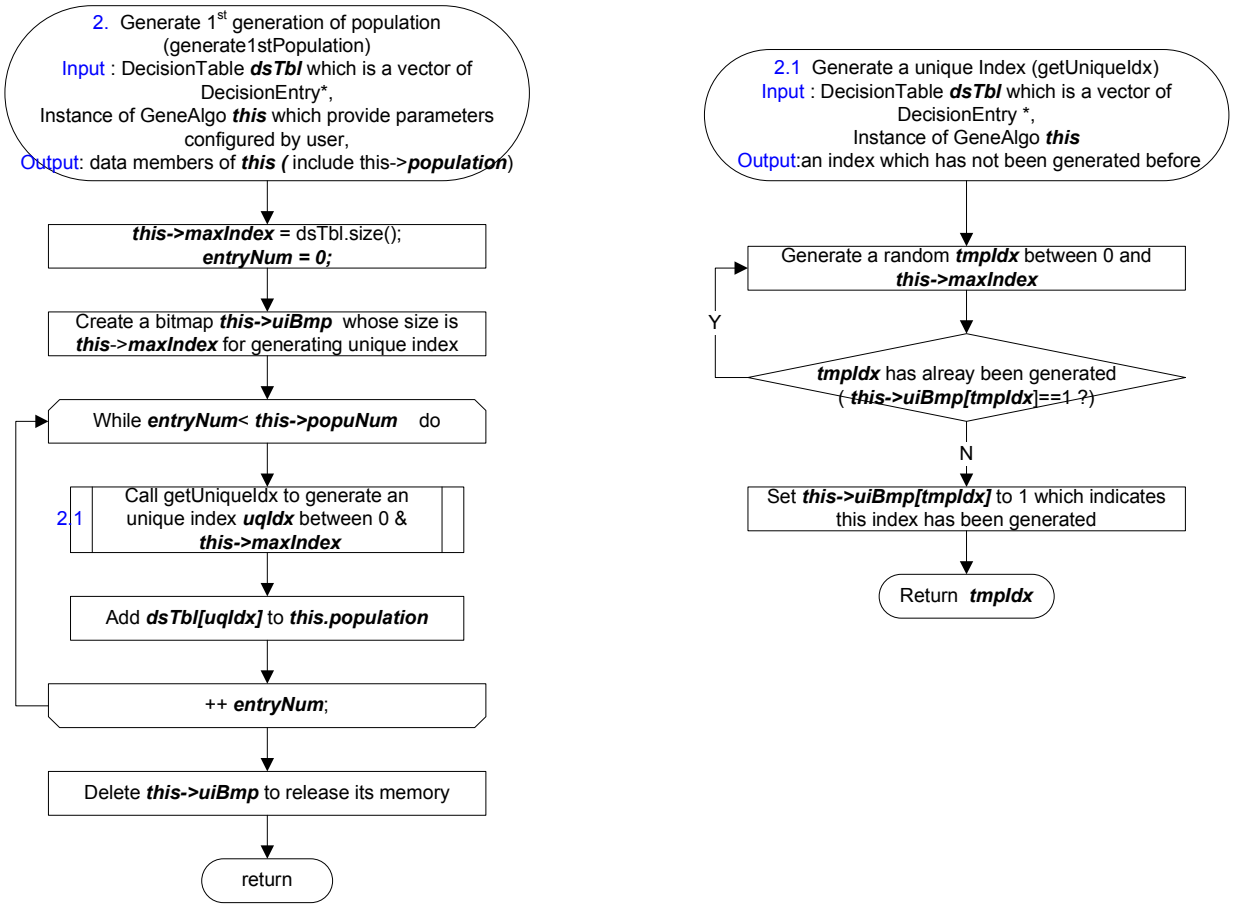


Figure V-11 Generate first generation of population

As shown in figure V-11, we generate the first generation of population by selecting certain number of unique entries from decision table randomly. The number of generations can be configured in the input data file by the parameter “PopuNum”.

Figure V-12 shows the procedure of calculating the fitness value for each individual in the current generation of population. It uses the formula of Fitness function provided in Section 4.2.3. As suggested by [9], we use bitmap to calculate C_r.

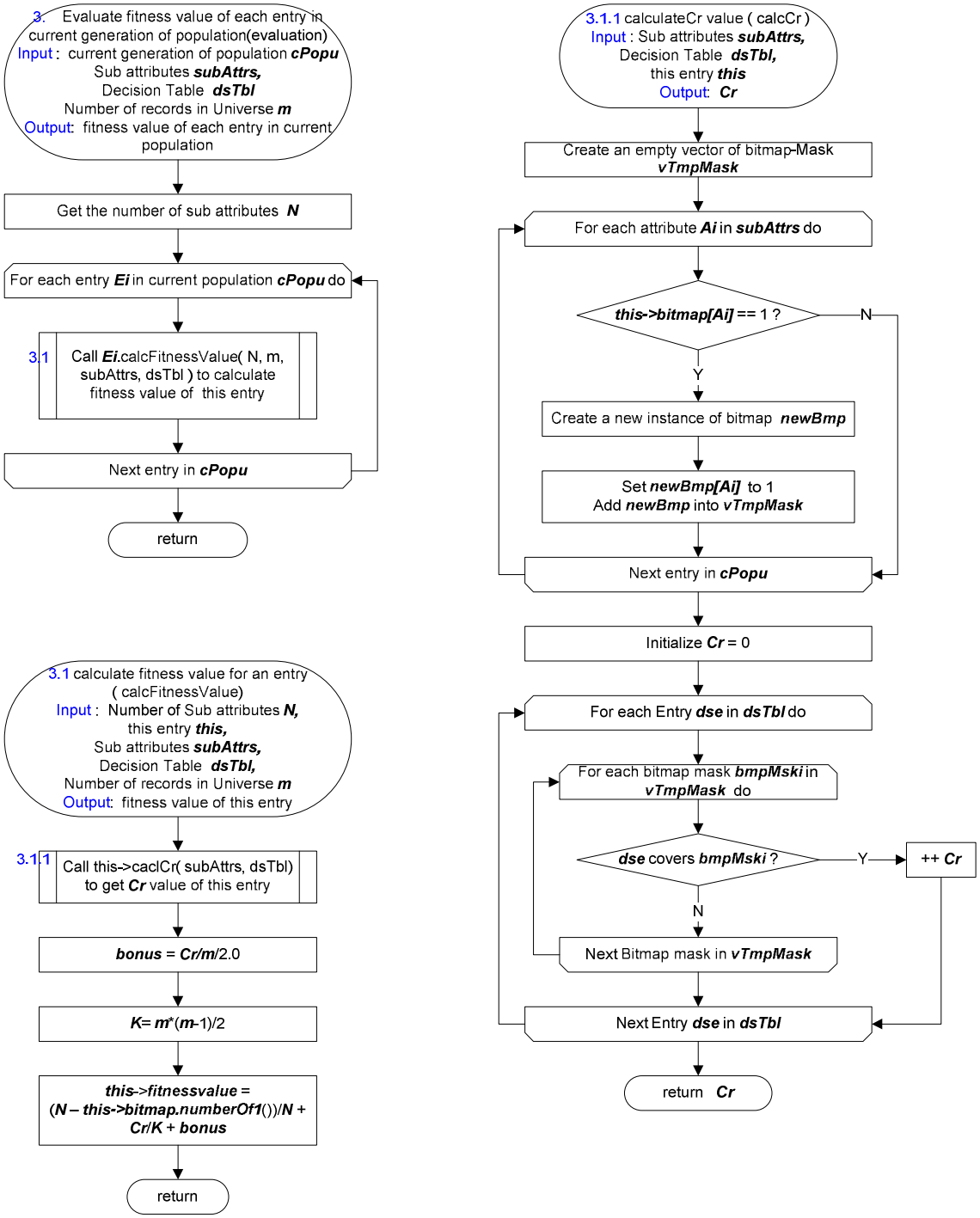


Figure V-12 Evaluate fitness value

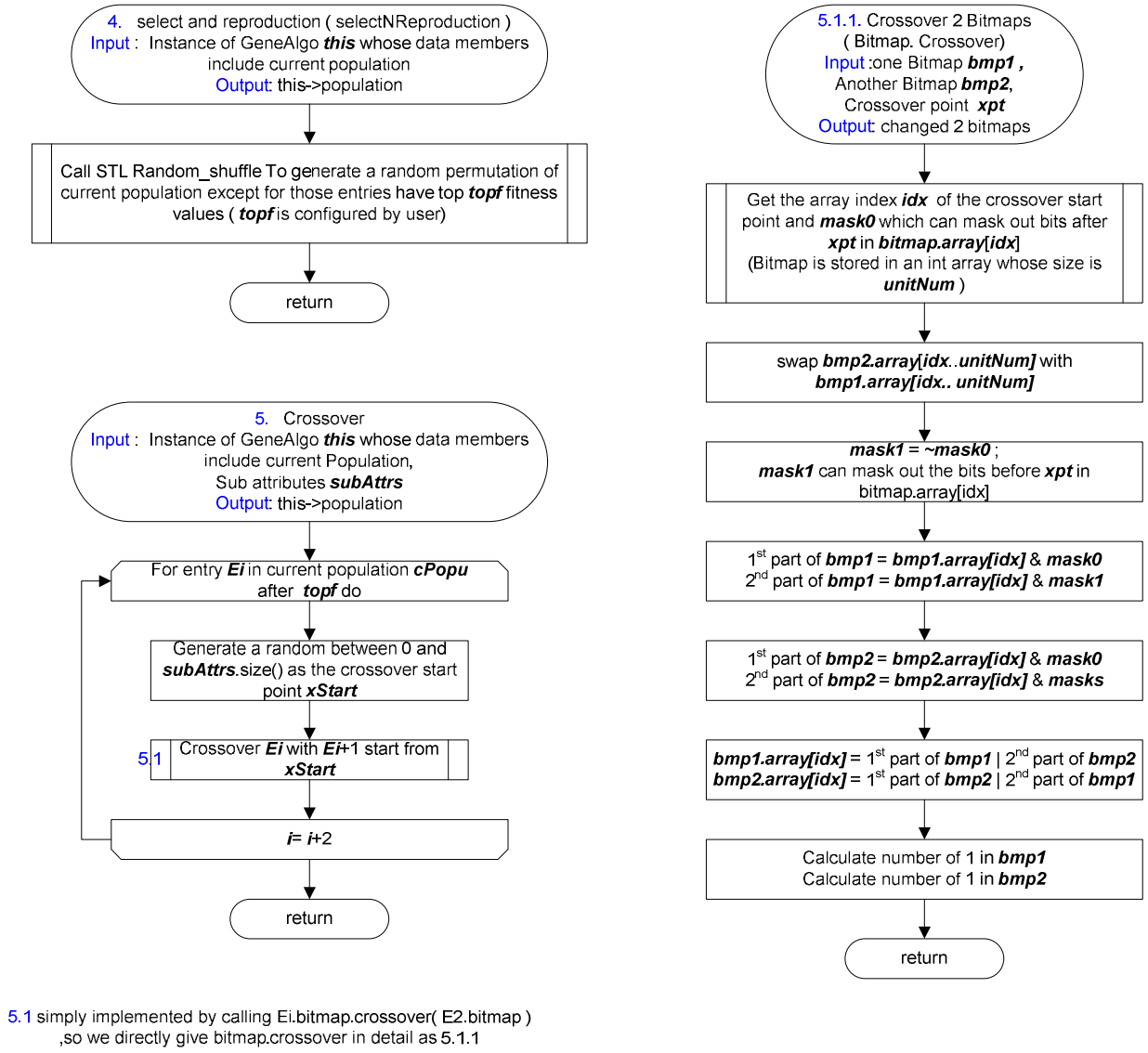


Figure V-13 Selection and crossover

Figure V-13 shows the implementation of selection and crossover. The selection procedure follows the idea of Elitism Selection described in Section 4.2.3. The individual with the top fitness score is copied into the new generation, whereas others are involved into crossover and mutation. As with in calculating fitness value, the crossover is also done by bitmap AND and OR operations, which speeds up the process.

Figure V-14 shows the procedure of mutation. The operation involves flipping one bit. However, we spend most of the time for this procedure on generating random numbers used to follow the

probability of mutation and to decide the individual and the position involved in the mutation randomly.

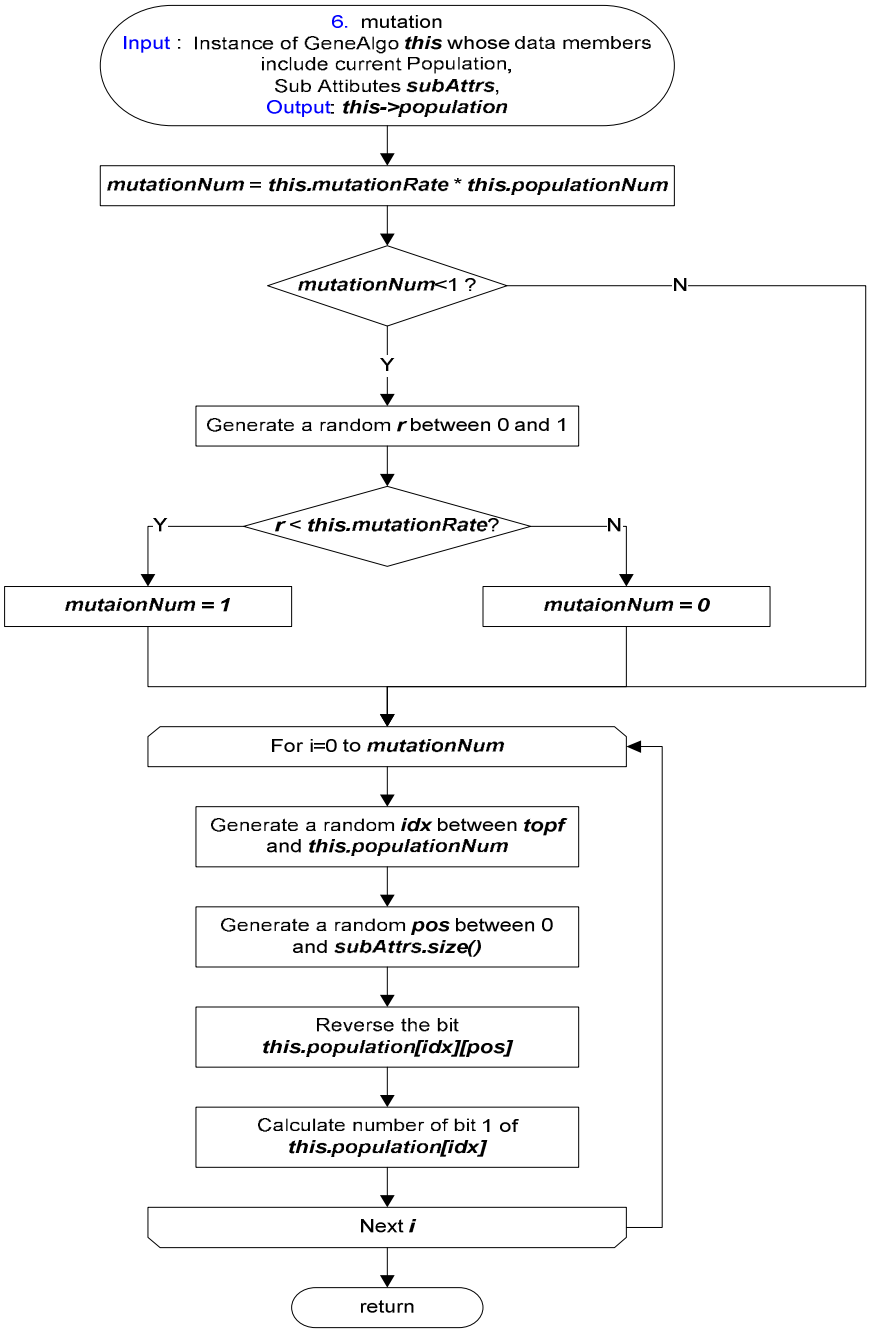


Figure V-14 Mutation

In the next chapter, we'll run the above two algorithm implementations with real world data.

VI. Tests and Results

We test our algorithm with data from the KEEL-dataset repository [16]. Eight data sets are selected from the standard classification data sets (i.e., led7digit, zoo, iris, flare, breast, banana, appendicitis, and wine). For estimating our system accuracy, five-fold cross validation is applied to the input data. The same data sets are tested with two different algorithms, the Boolean reasoning method and the genetic algorithm.

Table VI-1 Summary of testing result

					Rough Set Algorithm(Boolean Reasoning)						Genetic Algorithm					
Dataset Name	Attribute #	class #	Training Records #	Testing Record #	Condition Match #	Condition & Decision Match#	Non-Collision Match #	C Match Rate %	C&D Match Rate %	Non-Collistion Match Rate %	Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %
led7digit	7	10	2000	500	463	408	105	92.60	81.60	21.00	490	450	26	98.00	90.00	5.20
zoo	16	7	404	101	91	86.00	86.00	90.10	85.15	85.15	93	88	79	92.08	87.13	78.22
iris	4	3	600	150	97	94	91	64.67	62.67	60.67	134	130	107	89.33	86.67	71.33
flare	11	6	4264	1066	890	820	494	83.49	76.92	46.34	936	849	489	87.80	79.64	45.87
breast	9	2	1108	277	183	159	94	66.06	57.40	33.94	225	203	93	81.23	73.29	33.57
banana	2	2	21200	5300	3448	3136	2849	65.06	59.17	53.75	3448	3136	2849	65.06	59.17	53.75
appendicitis	7	2	424	106	31	30	29	29.25	28.30	27.36	61	58	42	57.55	54.72	39.62
Wine	13	3	712	178	102	96	94	57.30	53.93	52.81	93	90	85	52.25	50.56	47.75

Table VI-1 gives the results obtained by running our algorithm with 8 input data sets. The first five columns give information on the input datasets. The first column gives the name of the datset. Attribute number gives the number of attributes in each dataset. Class number is the number of decision values. Training Records number is the number of objects used to build the classifier model. Test Records number is the number of test objects used to build the classifier model. The middle six columns and the last six columns have identical column headers. The objective is to compare two algorithms with the same indicators. Condition Match number represents the number of objects in the test dataset whose condition attributes can be found in the training data. Condition & Decision Match number is the number of objects in the test dataset whose condition and decision attributes combination can be found in the related training dataset. Non-collision Match number is the number of objects in the test dataset classified by the rules, each of these objects has only one decision value according to one of the accurate rules. C Match

Rate = (Condition Match # / Testing Record #) * 100%, C&D Match Rate = (Condition & Decision Match # / Testing Record #) * 100%, Non-collision Match Rate = (Non-collision Match # / Testing Record #) * 100%.

We must pay more attention to C&D Match Rate. This value decides the accuracy of the decision system. Two algorithms have some differenct accuracy. Overall, the genetic algorithm is better than the Boolean reasoning function. Tables VI-2 to VI-9 gives the results obtained upon running the two algorithms with these 8 datasets. We use a five-fold cross validation test. The last line is the average test result based on the five-fold CV.

Test I: LED7 digit[16]

This test is to recognize what a 7-segment LED displays, i.e., to make decision in set{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. The condition attributes are 7-segments: LED1, LED2, … , LED7 with values in set {0, 1}. If there is no noise introduced, the problem may be much easier.

Table VI-2 result of **LED7digit**

					Rough Set Algorithm						Genetic Algorithm					
dataset Name	Attribute #	class #	Training Records #	Testing Record #	Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %	Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %
led2digit 1	7	10	400	100	93	83	24	93.00	83.00	24.00	99	92	3	99.00	92.00	3.00
led2digit 2			400	100	92	75	23	92.00	75.00	23.00	99	92	4	99.00	92.00	4.00
led2digit 3			400	100	93	85	16	93.00	85.00	16.00	98	91	0	98.00	91.00	0.00
led2digit 4			400	100	90	82	19	90.00	82.00	19.00	96	87	14	96.00	87.00	14.00
led2digit 5			400	100	95	83	23	95.00	83.00	23.00	98	88	5	98.00	88.00	5.00
led2digit (Ave)	7	10	2000	500	463	408	105	92.60	81.60	21.00	490	450	26	98.00	90.00	5.20

The results are shown in Table VI-2. As can be seen in the last row of the table, Boolean reasoning function method yields 81.60% accuracy in average and the GA is much better, reaching 90%.

Test II: Zoo [16]

This test aims to determine 7 predefined animals' classes such as mammals, bird and so on. The value of classes can be any number in {0, 1, 2, 3, 4, 5, 6, 7}. The 16 description (condition)

attributes are related to the distinct characters among species such as hair, feathers, eggs, fins, legs, backbones etc. with value in {0, 1}. 0 represents an animal that "doesn't have" the feature, 1 means it "has" the feature.

Table VI-3 result of **Zoo**

Data set Name	Attribute #	class #	Training Records #	Testing Record #	Rough Set Algorithm						Genetic Algorithm					
					Condition Match #	Condition & Decision Match#	No-Collision Match #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %	Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %
zoo1	16	7	80	21	19	18	18	90.48	85.71	85.71	20	19	15	95.24	90.48	71.43
zoo2			81	20	19	18	18	95.00	90.00	90.00	18	17	13	90.00	85.00	65.00
zoo3			81	20	18	18	18	90.00	90.00	90.00	18	18	17	90.00	90.00	85.00
zoo4			81	20	19	16	16	95.00	80.00	80.00	19	18	18	95.00	90.00	90.00
zoo5			81	20	16	16	16	80.00	80.00	80.00	18	16	16	90.00	80.00	80.00
zoo (Ave)	16	7	404	101	91	86	86	90.10	85.15	85.15	93	88	79	92.08	87.13	78.22

The results are tabulated in Table VI-3 in detail. The Boolean function reasoning method gives 85.15% average accuracy. The GA is a little bit better with 87.13% average.

Test III: Iris [16]

This test is from pattern recognition. The test is to differentiate 3 types of iris from {Iris-setosa, Iris-versicolor, Iris-virginica}. The description attributes are sepal length with a value in [4.3, 7,9]; sepal width with a value in[2.0, 4.4]; petal length with a value in [1.0, 6.9]; and petal width with a value in [0.1, 2.5].

Table VI-4 result of **Iris**

Dataset Name	Attribute #	class #	Training Records #	Testing Record #	Rough Set Algorithm						Genetic Algorithm					
					Condition Match #	Condition & Decision Match#	No-Collision Match #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %	Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %
iris1	4	3	120	30	15	15	15	50.00	50.00	50.00	27	27	20	90.00	90.00	66.67
iris2			120	30	26	25	22	86.67	83.33	73.33	22	21	20	73.33	70.00	66.67
iris3			120	30	16	16	16	53.33	53.33	53.33	28	26	21	93.33	86.67	70.00
iris4			120	30	24	22	22	80.00	73.33	73.33	29	29	21	96.67	96.67	70.00
iris5			120	30	16	16	16	53.33	53.33	53.33	28	27	25	93.33	90.00	83.33
iris (Ave)	4	3	600	150	97	94	91	64.67	62.67	60.67	134	130	107	89.33	86.67	71.33

The iris classification results are shown in Table VI-4. Boolean reasoning function method yields 62.67% average accuracy, which is worse than the GA's 86%. Later we'll give an explanation for this difference.

Test IV: Flare [16]

This test aims to decide some features about solar flare represented by letter set {H, D, C, B, E, F}. Each description attribute is one of captured features for one active region of sun such as largest spots size, spot distribution and etc, whose values come from letter sets or integer sets.

Table VI-5 Result of **Flare**

					Rough Set Algorithm						Genetic Algorithm					
Dataset Name	Attribute #	class #	Training Records #	Testing Record #	Condition Match #	Condition & Decision Match#	No-Collision Match #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %	Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %
flare1	11	6	852	214	177	164	99	82.71	76.64	46.26	185	171	95	86.45	79.91	44.39
flare2			853	213	181	168	101	84.98	78.87	47.42	187	170	100	87.79	79.81	46.95
flare3			853	213	174	161	93	81.69	75.59	43.66	187	168	92	87.79	78.87	43.19
flare4			853	213	185	169	99	86.85	79.34	46.48	191	175	101	89.67	82.16	47.42
flare5			853	213	173	158	102	81.22	74.18	47.89	186	165	101	87.32	77.46	47.42
flare (Ave)	11	6	4264	1066	890	820	494	83.49	76.92	46.34	936	849	489	87.80	79.64	45.87

The results are shown in Table VI-5. As can be seen in the table, the Boolean reasoning function method has 76.92% average accuracy and the GA gives 79.64% average in the average case.

Test V: Breast [16]

This test is used to determine whether the patient with breast cancer has recurrence or not after treatment. The decision value comes from the following set {no-recurrence-events, recurrence-events}. The condition attributes are age, tumor-size, etc..

Table VI-6 Result of **Breast**

dataset Name	Attribute #	class #	Training Records #	Testing Record #	Rough Set Algorithm						Genetic Algorithm					
					Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collision match Rate %	Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collision match Rate %
breast1	9	2	222	55	40	33	20	72.73	60.00	36.36	49	44	24	89.09	80.00	43.64
breast2			221	56	36	30	17	64.29	53.57	30.36	42	38	14	75.00	67.86	25.00
breast3			220	57	37	33	16	64.91	57.89	28.07	48	41	16	84.21	71.93	28.07
breast4			225	52	31	28	20	59.62	53.85	38.46	41	37	21	78.85	71.15	40.38
breast5			220	57	39	35	21	68.42	61.40	36.84	45	43	18	78.95	75.44	31.58
breast (Ave)	9	2	1108	277	183	159	94	66.06	57.40	33.94	225	203	93	81.23	73.29	33.57

The results are shown in Table VI-6. The GA is more accurate than the Boolean reasoning function method, i.e. GA yields 81.23% and Boolean reasoning function gives 57.40%.

Test VI: Banana [16]

This is an artificial test to determine the predefined shape of bananas. The decision value comes from the set {-0.1, 0.1}. There are two condition attributes with values..

Table VI-7 Result of Banana

dataset Name	Attribute #	class #	Training Records #	Testing Record #	Rough Set Algorithm						Genetic Algorithm					
					Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collision match Rate %	Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collision match Rate %
banana 1	2	2	4240	1060	691	624	579	65.19	58.87	54.62	691	624	579	65.19	58.87	54.62
banana 2			4240	1060	700	631	559	66.04	59.53	52.74	700	631	559	66.04	59.53	52.74
banana 3			4240	1060	657	600	541	61.98	56.60	51.04	657	600	541	61.98	56.60	51.04
banana 4			4240	1060	702	640	591	66.23	60.38	55.75	702	640	591	66.23	60.38	55.75
banana 5			4240	1060	698	641	579	65.85	60.47	54.62	698	641	579	65.85	60.47	54.62
banana (Ave)	2	2	21200	5300	3448	3136	2849	65.06	59.17	53.75	3448	3136	2849	65.06	59.17	53.75

The results are shown in Table VI-7. For this application, the two algorithms yield similar results on the average.

Test VII: Appedicitis [16]

No detail information could be found for this dataset.

Table VI-8 result of Appendicitis

Dataset Name	Attribute #	class #	Training Records #	Testing Record #	Rough Set Algorithm						Genetic Algorithm					
					Condition Match #	Condition & Decision Match #	No-Collision Match #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %	Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %
appendicitis1			84	22	5	5	5	22.73	22.73	22.73	15	15	12	68.18	68.18	54.55
appendicitis2			85	21	6	6	5	28.57	28.57	23.81	13	12	8	61.90	57.14	38.10
appendicitis3			85	21	11	10	10	52.38	47.62	47.62	10	9	9	47.62	42.86	42.86
appendicitis4			85	21	5	5	5	23.81	23.81	23.81	17	16	8	80.95	76.19	38.10
appendicitis5	7	2	85	21	4	4	4	19.05	19.05	19.05	6	6	5	28.57	28.57	23.81
appendicitis (Ave)	7	2	424	106	31	30	29	29.25	28.30	27.36	61	58	42	57.55	54.72	39.62

The appendicitis classifier has overall low accuracy. However, the GA has better result than the Boolean reasoning function. GA yields 54.72% average accuracy while the Boolean reasoning function only reaches 28.30%.

Test VIII. Wine [16]

This test aims to classify 3 cultivars of wine (value:0, 1, 2) comes from Italy based on the chemical constituent. All the constituent values are represented by float numbers.

Table VI-9 result of **Wine**

dataset Name	Attribute #	class #	Training Records #	Testing Record #	Rough Set Algorithm						Genetic Algorithm					
					Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	C&D Match Rate %	Non-Collistion match Rate %	Condition Match #	Condition & Decision Match#	No-Collision #	C Match Rate %	Match Rate %	Non-Collistion match Rate %
wine 1			142	36	21	21	21	58.33	58.33	58.33	17	16	15	47.22	44.44	41.67
wine 2			142	36	25	22	22	69.44	61.11	61.11	18	18	18	50.00	50.00	50.00
wine 3	13	3	142	36	13	13	13	36.11	36.11	36.11	14	14	14	38.89	38.89	38.89
wine 4			143	35	24	23	21	68.57	65.71	60.00	18	18	18	51.43	51.43	51.43
wine 5			143	35	19	17	17	54.29	48.57	48.57	26	24	20	74.29	68.57	57.14
Wine (Ave)	13	3	712	178	102	96	94	57.30	53.93	52.81	93	90	85	52.25	50.56	47.75

The wine's cultivars classification results are shown in Table VI-9. More interesting that the Boolean reasoning function is a little bit better then the GA, that is 53.93% and 50.56% average accuracy individually. This is the only example for which the Boolean function reasoning method gave a slightly better result than the GA.

In the next section, we'll give some concluding remarks and future direction.

VII. Conclusion

7.1 Complexity

Finding minimal reducts for the decision table is an NP-hard problem [13]. The Boolean reasoning function, an accurate method, can find the optimal solution. Accordingly, its complexity is non-polynomial. The genetic algorithm can be used as a heuristic for finding minimal reducts, the computation of the number of covered rows of the distinction table is time consuming.

7.2 Data mining and machine learning

Based on RS Theory, the pattern inside the dataset can be discovered, especially, when we only have a limited amount of data. Since there is nothing absolute in this world, finding an approximately equivalent information system to the original one is more realistic. However, comparing the two reduced algorithms, we found that the GA is "more efficient" than the Boolean reasoning function especially for a large number of attributes.

In this work, we found that the Boolean reasoning method is quite strict when compared to the GA. The genetic algorithm, through crossover and mutation, has the capability of exploring the search more freely than the Boolean reasoning methods. The GA, in many respects, is more flexible than the rigid Boolean reasoning method, thus producing better results on the average.

VIII. Future work

One possible extension of this work is to combine both algorithms, thus creating a hybrid algorithm[10]. The result is probably better than using the GA only. One could also use other adaptive search algorithms such as ant colony, simulated annealing.

Reference

[1] Pawlak, Z. (1991). *Rough sets—Theoretical aspects of reasoning about data.* Dordrecht, The Netherlands: Kluwer Academic.

[2] Pawlak, Z. (2004). Elementary rough set granules: Toward a rough set processor. In Pal, S. K. & Polkowski, L. (Eds.). *Rough-neural computing: Techniques for computing with words* (pp. 5-13). Heidelberg, Germany: Springer Verlag.

[3] Enroth, S. *Rough set—Introduction molecular bioinfomatic X3* [PDF slides]. Retrieved from http://www.anst.uu.se/stenr451/mb330/2009/lect12.pdf

[4] Stefanowski, J. (1998). On rough set based approaches to induction of decision rules. In Polkowski L. & Skowron A. (Eds.), *Rough set in data mining and knowledge discovery* (pp. 500-530). Heidelberg, Germany: Physica-Verlag.

[5] Cao, Y. F., Liu, S., Zhang, L. D., Qin, J., Wang, J. & Tang K. X. (2006). Prediction of protein structure class with rough sets. *BMC Bioinformatics, 7*(20). doi:10.1186/1471-2105-7-20

[6] Lazar, A. (2002). Heuristic knowledge discovery for archaeological data using genetic algorithm and rough sets. In Abbass, H. A., Newton, C. S., & Sarker, A. R., *Heuristics and optimization for knowledge discovery* (pp. 263-278). Hershey, PA: Idea Group.

[7] Strömbergsson, H., Prusis, P., Midelfart, H., Lapinsh, M., Wikberg, J. E. & Komorowski, J. (2006). Rough set-based proteochemometrics modeling of Gprotein-coupled receptor-ligand interactions. In Gacia-Moreno, B. (Ed.). *Proteins: Structure, function, and bioinformatics, 63*(1), (pp. 24-34). doi: 10.1002/prot.20777

[8] Ligand (biochemistry) (2009). In *Wikipedia, the free encyclopedia*. Retrieved May 10, 2012, from http://en.wikipedia.org/wiki/Ligand_(biochemistry).

[9] Wróblewski J. (1995). Finding minimal reducts using genetic algorithms. *Proceedings of the Second Annual Join Conference on Information Sciences*, pp. 186-189, September 28-October 1, Wrightsville Beach, NC. Also in: ICS Research report 16/95, Warsaw University of Technology.

[10] Wróblewski J. (1998). Genetic algorithms in decomposition and classification problem. In L. Polkowski, A. Skowron (Eds.). *Rough sets in knowledge discovery*

[11] Khuri S. (2010). Design and analysis of algorithms. *Computer Science 255. Introduction to GA.*

[12] http://www.obitko.com/tutorials/genetic-algorithms/selection.php, last retrieved on Sept. 20th. 2012.

[13] Skowron, A. & Rauszer, C. (1992). *The discernibility matrices and functions in information systems*. In R. Slowinski (Ed.). pp. 331-362.

[14] Pawlak, Z. & Skowron, A. (2007). Rough sets and Boolean reasoning. *Information Science, 177*(1) 41-73, Electronic Edition pubzone.org

[15] Skowron, A. & Rauszer, C. (1992). The discernibility matrices and functions in information systems. In R. Słowiński (Ed.). *Intelligent decision support—Handbook of applications and advances of the rough sets theory, system theory, knowledge engineering and problem solving, 11*. Dordrecht, The Netherlands: Kluwer Academic, pp. 331-362.

[16] http://sci2s.ugr.es/keel/category.php?cat=clas, last retrieved on Oct. 25th 2012.

Appendix: Source Code (C++)

```
/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : AndOrExpr.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/23 15:47:02
  Last Modified :
  Description   : implementation of class AndOrExpr etc...
  Function List :
              operator<<
              VecDecisionFunc.~VecDecisionFunc
  History       :
  1.Date        : 2012/9/23 15:47:02
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#include    "AndOrExpr.h"
#include    <iomanip>

using namespace std;

namespace rough_set {
/*****************************************************************************
*                                                                            *
*                   implementation of AndExpr                              *
*                                                                            *
*****************************************************************************/
/*
AndExpr::AndExpr( const AndExpr & expr2){
    compiler will synthesize one by calling Bitmap( Bitmap& );
}
*/

bool    AndExpr::operator < ( const AndExpr & rhs ) const {
    if ( ( this->bitNum_  != rhs.bitNum_  ) ||
         ( this->unitNum_ != rhs.unitNum_ ) ) {
        throw runtime_error("\'<\' Do not support different length Bitmap comparison");
    }

    for ( BitIdx_t i=0; i<rhs.unitNum_ ; i++ ) {
        if ( this->pBits_[i]< rhs.pBits_[i]) {
            return true;
        } else if ( this->pBits_[i] == rhs.pBits_[i] ) {
            continue;
        } else {
            return false;
        }
    }
    return false;
}

bool    AndExpr::operator == ( const AndExpr & rhs ) const {
    if ( ( this->bitNum_  != rhs.bitNum_  ) ||
         ( this->unitNum_ != rhs.unitNum_ ) ) {
        throw runtime_error("\'==\' Do not support different length Bitmap comparison");
    }

    for ( BitIdx_t i=0; i<rhs.unitNum_ ; i++ ) {
        if ( this->pBits_[i]!= rhs.pBits_[i]) {
            return false;
        }
    }
    return true;
}
```

```
ostream& operator <<( ostream &os , const AndExpr & rhs){
    BitIdx_t numOf1Scanned = 0 , numOf1= rhs.getNumOf1();
    BitIdx_t tmpnum, numOfBits = tmpnum=rhs.getBitNum();
    int width = 0;
    do { ++width; } while ( tmpnum /=10 );
    if ( numOf1 ) {
        os<<" ( ";
        for(BitIdx_t i=0; i<numOfBits; i++ ) {
            if ( rhs.getBit( i )) {
                numOf1Scanned++;
                os<<setw(width)<<i;
                if ( numOf1Scanned < numOf1 ){
                   os <<'^';
                } else {
                    if ( numOf1Scanned == numOf1){
                        os<<" ";
                    } else {                // too many 1s
                        throw runtime_error("Data inconsistant");
                    }
                }
            }
        }
    } else {
        os<<"null";
    }
    os<<" ) ";
    return os;
}


/*****************************************************************************
*                                                                            *
*                   implementation of AndOrExpr                             *
*                                                                            *
*****************************************************************************/
AndOrExpr::~AndOrExpr(){
/* after changed to set , compiler will call the destructor for each item in the set
    for ( AndOrExpr::iterator eprIt = begin(); eprIt != end(); eprIt++ ) {
        delete (*eprIt);
    }
    */
}

const size_t MAX_OUT_EXPR_NUM = 2000100;
ostream& operator <<( ostream &os , const AndOrExpr & rhs){
    AndOrExpr::const_iterator theEnd;
    size_t  orExprNum = rhs.size();
    size_t  lastNum;
    if ( orExprNum ==0 ) {
        os<<"No items"<<endl;
        return os;
    } else {
        if ( orExprNum > MAX_OUT_EXPR_NUM ) {
            os<<orExprNum<<" exprs , only print first "<<MAX_OUT_EXPR_NUM<< " items"<<endl;
            lastNum = MAX_OUT_EXPR_NUM;
        } else {
            lastNum = orExprNum;
        }
    }
    size_t  outNum = 0;
    for( AndOrExpr::const_iterator it = rhs.begin();it!= rhs.end(); it++) {
        os<<(*it);
        if( ++outNum < lastNum ) {
            os<<" V"<<endl;
        } else {
            os<<endl;
            break;
        }
    }
    os<< "Total: "<<lastNum<<" AND-exprs"<<endl<<endl;
```

```
    return os;
}


/*************************************************************************
*                                                                        *
*                  implementation of VecAndOrExpr                         *
*                                                                        *
*************************************************************************/
ostream& operator <<( ostream &os , const VecAndOrExpr & rhs){
    size_t matrixNum = rhs.size();
    os<<endl;
    if ( matrixNum > 1 ) {
        for ( VecAndOrExpr::const_iterator vecIt = rhs.begin();
              vecIt != rhs.end(); vecIt++ ) {
            os<<"Decision Function in And-Or-Expr for scenario "<< 1+vecIt- rhs.begin()<<" =
"<<endl<<**vecIt;
        }
    } else {
        os<<"Decision Function in And-Or-Exp = "<<endl<<**rhs.begin();
    }
    return os;
}

VecAndOrExpr::~VecAndOrExpr(){
    for ( VecAndOrExpr::iterator vecIt = begin(); vecIt != end(); vecIt++ ) {
        delete (*vecIt);
    }
}

}

/*************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *************************************************************************
  File Name     : AndOrExpr.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/23 15:10:39
  Last Modified :
  Description   : class definiition of And-Or-Logic_Express and vector of
                    And-Or-Expr
  Function List :
  History       :
  1.Date        : 2012/9/23 15:10:39
    Author      : Shuang Wang
    Modification: Created file

*************************************************************************/
#ifndef AND_OR_EXPR_H_
#define AND_OR_EXPR_H_

#include    "Bitmap.h"
#include    <set>
#include       <vector>
#include    <iostream>

namespace rough_set {

class   AndExpr: public Bitmap {               // And of attributes, eg. x1^x2^x3....
friend  std::ostream & operator<< ( std::ostream & os, const AndExpr& rhs);

public:
    AndExpr( BitIdx_t bitNum ): Bitmap( bitNum) {}
    //AndExpr( AndExpr & expr2);   //call Bitmap(& Bitmap)
    AndExpr( const Bitmap & bmp):Bitmap( bmp) {}
    ~AndExpr() {}
    bool    operator < ( const AndExpr & rhs ) const ;
    bool    operator == ( const AndExpr & rhs ) const ;
```

```
};

class   AndOrExpr:public std::set< AndExpr> {
friend  std::ostream & operator<< ( std::ostream & os, const AndOrExpr& rhs);

public:
    ~AndOrExpr();
};

class   VecAndOrExpr: public std::vector< AndOrExpr*> {
friend  std::ostream & operator<< ( std::ostream & os, const VecAndOrExpr& rhs);
public:
    ~VecAndOrExpr();
};

}
#endif /* AND_OR_EXPR_H_ */
/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : Attribute.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/10/17 23:53:10
  Last Modified :
  Description   : implement functions of class Attribute which need include
------------------DataTypeFactoryBase to avoid an include loop
  Function List :
  History       :
  1.Date        : 2012/10/17 23:53:10
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/

#include    "Attribute.h"
#include    "DataTypeFactories.h"   //this can be included here but not in "Attribute.h"
using namespace std;
namespace rough_set {           // stuff used in rough sets peojects

DataTypeBase * Attribute::createValueObj(  ) {
    return dataFactory_->create();
}

bool    Attribute::supportRange() {
    return     dataFactory_->supportRange();
}

ostream & operator << (ostream & os , Attributes& rhs) {
    for ( Attributes::iterator  attrIt = rhs.begin(); attrIt!= rhs.end(); attrIt++) {
        int width = attrIt->getFieldLen()+PADDING;
//        std::cout<<" "<<width<<" ";
        os<<std::setw(width)<<attrIt->getName();
        attrIt->setFieldLen(width);
    }
       os<<std::endl;
       return os;
}

}

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : Attribute.h
  Version       : Initial Draft
  Author        : Shuang Wang
```

```
  Created       : 2012/6/30 21:24:19
  Last Modified :
  Description   : definition of class Atrribute
  Function List :
  History       :
  1.Date        : 2012/6/30 21:24:19
    Author      : Shuang Wang
    Modification: Created file

*********************************************************************************/
#ifndef  ATTRIBUTE_H_
#define  ATTRIBUTE_H_

//#include    "DataTypeFactories.h"         // to avoid include loop
#include       <iostream>
#include       <iomanip>
#include       <vector>
namespace rough_set {           // stuff used in rough sets peojects

const int PADDING =3;
class   DataTypeFactoryBase   ;
class   DataTypeBase;
class Attribute{
    const   std::string name_;
    int     fieldLen;
    DataTypeFactoryBase* dataFactory_;
    double  lower_;
    double  upper_;
public:
    Attribute(  const std::string & iName, DataTypeFactoryBase* iDataCreator,
                double  lower, double upper ):
                    name_(iName), fieldLen(iName.size()),
                    dataFactory_( iDataCreator ) ,
                    lower_(lower), upper_(upper) {
    }

    DataTypeBase * createValueObj(  );

    const std::string& getName() const {
        return name_;
    }

    int getFieldLen() const {
        return fieldLen;
    }

    void    setFieldLen( int newLen ) {
        fieldLen = newLen;
    }

    bool    supportRange();

    double  getUpper() const { return upper_; }
    double  getLower() const { return lower_; }

};

//typedef std::vector<Attribute>      Attributes;
class Attributes:public std::vector<Attribute> {
    friend std::ostream & operator << (std::ostream & os , Attributes& rhs);
};


}
#endif


/*********************************************************************************
```

```
 ***************************************************************************
  File Name     : AttributeSubSet.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/5 23:06:14
  Last Modified :
  Description   : implementation of class AttributeSubSet & VecAttributeSub-
                  Sets
  Function List :
  History       :
  1.Date        : 2012/7/5 23:06:14
    Author      : Shuang Wang
    Modification: Created file

***************************************************************************/
#include "AttributeSubSet.h"

using namespace std;
namespace rough_set {

AttrIdx_t AttributeSubSet::totAttrNum_;
ostream & operator << ( ostream &os, const AttributeSubSet& rhs) {
    os<<"\t{";                        // start a new subset
    int  end = * rhs.rbegin();
    for ( AttributeSubSet::const_iterator colIt = rhs.begin();
                colIt != rhs.end(); colIt++ ){
        os<< *colIt;
        if ( *colIt != end )
            os<<", ";
    }
    os<<" }"<<endl;
    return os;
}

ostream & operator << ( ostream &os, const VecAttrSubSets& rhs) {
    os<<"{ ";
    for ( VecAttrSubSets::const_iterator attrSubsetIt=rhs.begin();
            attrSubsetIt!= rhs.end(); attrSubsetIt++ ){
        AttributeSubSet & curSet= ** attrSubsetIt;
        os<< curSet;
    }
    os<<"}"<<endl<<endl;
       return os;
}

VecAttrSubSets::~VecAttrSubSets(){
    for ( VecAttrSubSets::iterator attrSubsetIt=begin();
            attrSubsetIt!= end(); attrSubsetIt++ ){
#ifdef XDEBUG
        cout<<"will delete Attribute Subset "<< **attrSubsetIt <<endl;
#endif
        delete (* attrSubsetIt);
    }
}


}



/***************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ***************************************************************************
  File Name     : AttributeSubSet.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/5 22:34:39
  Last Modified :
  Description   : definition of class AttributeSubSet to define which column
```

```
                  will be considered when compare two record.
                  and definition of class VecAttrSubSets
  Function List :
  History       :
  1.Date        : 2012/7/5 22:34:39
    Author      : Shuang Wang
    Modification: Created file

*******************************************************************************/
#ifndef ATTR_SUB_SET_H_
#define ATTR_SUB_SET_H_
#include    <set>
#include    <vector>
#include    <iostream>
namespace rough_set {           // stuff used in rough sets peojects

//typedef     std::set<int>   AttributeSubSet;
typedef     size_t      AttrIdx_t;
class   AttributeSubSet:public std::set<AttrIdx_t>  {
//public:
    static  AttrIdx_t totAttrNum_;
    friend std::ostream & operator << ( std::ostream &os, const AttributeSubSet& rhs) ;
public:
    static void setTotAttrNum( int n) {
        totAttrNum_ = n;
    }

    static int  getTotAttrNum( ) {
        return totAttrNum_ ;
    }

};

class   VecAttrSubSets: public std::vector<AttributeSubSet*> {
    friend std::ostream & operator << ( std::ostream &os, const VecAttrSubSets& rhs) ;
public:
    ~VecAttrSubSets();
} ;

}

#endif /* ATTR_SUB_SET_H_ */


/*******************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : Bitmap.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/8 14:38:55
  Last Modified :
  Description   : implementation of class Bitmap
  Function List :
  History       :
  1.Date        : 2012/7/8 14:38:55
    Author      : Shuang Wang
    Modification: Created file

*******************************************************************************/

#include    "Bitmap.h"
#include    <exception>
#include    <cstring>               // for memset
#include    <iomanip>               // for setw
using namespace std;

namespace rough_set {
```

```
Bitmap::Bitmap(BitIdx_t n):bitNum_(n) ,
        unitNum_(( n+ BIT_PER_UNIT - 1)/BIT_PER_UNIT ) ,numOf1_(0){
    pBits_ = new BitmapUnit[ unitNum_ ];
    //memset(pBits_,0, sizeof( pBits_));  // a bug
    memset(pBits_,0, unitNum_* sizeof(BitmapUnit));
}

Bitmap::Bitmap( const Bitmap & one): bitNum_( one.bitNum_),
    unitNum_( one.unitNum_), numOf1_( one.numOf1_ ) {
    pBits_ = new BitmapUnit[ unitNum_];
    memcpy( pBits_, one.pBits_, unitNum_ );
}

Bitmap::~Bitmap() {
    delete []pBits_;
}

void    Bitmap::getBitPosition( BitIdx_t bitIndex,
                BitIdx_t & arrayIndex, BitmapUnit& mask) const {
    if ( bitIndex >= bitNum_ ) {
        throw runtime_error(string("Bit Index out of range"));
    }
    arrayIndex = bitIndex/BIT_PER_UNIT ;
    if ( arrayIndex >= unitNum_ ) {
        throw runtime_error(string("Array Index out of range"));
    }
    BitIdx_t bitpos = BIT_PER_UNIT -1 - (bitIndex &( BIT_PER_UNIT -1 )) ;           //= %
    mask = 1<< bitpos;
}

/* get the value (0 or 1 ) of a certain bit */
bool    Bitmap::getBit ( BitIdx_t bitIndex ) const {
    BitIdx_t arrayIndex;
    BitmapUnit mask;
    getBitPosition( bitIndex, arrayIndex, mask);
    return ( 0!= (pBits_[arrayIndex]&mask) );
}

void    Bitmap::setBit ( BitIdx_t bitIndex) {
    BitIdx_t arrayIndex;
    BitmapUnit mask;
    getBitPosition( bitIndex, arrayIndex, mask);
    bool oldBit = getBit( bitIndex);
    pBits_[arrayIndex] |= mask;         // set the bit to 1;
    if ( !oldBit)    ++numOf1_;
}

void    Bitmap::resetBit ( BitIdx_t bitIndex ) {
    BitIdx_t arrayIndex;
    BitmapUnit mask;
    bool oldBit = getBit( bitIndex);
    getBitPosition( bitIndex, arrayIndex, mask);
    pBits_[arrayIndex] &= ( ~mask );    // set the bit to 0;
    if ( oldBit ) --numOf1_;
}

void    Bitmap::reverseBit( BitIdx_t bitIndex ){
    BitIdx_t arrayIndex;
    BitmapUnit mask;

    getBitPosition( bitIndex, arrayIndex, mask);
    bool oldBit = getBit( bitIndex);
    pBits_[arrayIndex] ^= mask ;        // xor 1 , reverse it.
    if( oldBit) --numOf1_;
        else ++numOf1_;
}

void    Bitmap::countNumOf1() {
    BitIdx_t count = 0;
    for ( BitIdx_t i = 0; i< unitNum_ ; i++ ) {
        BitmapUnit tmp = pBits_[i];
```

```
        while( tmp ) {
            count+= tmp &1;
            tmp>>=1;
        }
    }
    numOf1_ = count;
}

void    Bitmap::getCrossOverPosNMask( BitIdx_t bitIndex,
                BitIdx_t & arrayIndex, BitmapUnit& mask) const {
    if ( bitIndex >= bitNum_ ) {
        throw runtime_error(string("Bit Index out of range"));
    }
    arrayIndex = bitIndex/BIT_PER_UNIT ;
    if ( arrayIndex >= unitNum_ ) {
        throw runtime_error(string("Array Index out of range"));
    }
    BitIdx_t bitpos = BIT_PER_UNIT -1 - bitIndex &( BIT_PER_UNIT -1 ) ;           //= %
    mask = static_cast<BitmapUnit>(-1)<< bitpos;
}

void    Bitmap::crossOver( Bitmap& op2 , BitIdx_t crossPoint){
    BitIdx_t arrayIndex;
    BitmapUnit mask0;
    getCrossOverPosNMask( crossPoint, arrayIndex, mask0);
    /* 1. exchange bytes after the byte this bit exists in */
    for ( BitIdx_t i= arrayIndex+1; i< unitNum_; i++ ){
        int tmp = pBits_[i];
        pBits_[i] = op2.pBits_[i];
        op2.pBits_[i] = tmp;
    }
    /* 2. exchange those bits after this bit in same byte */
    BitmapUnit mask1 = ~mask0;
    BitmapUnit op1_1stPart= pBits_[arrayIndex]&mask0;
    BitmapUnit op1_2ndPart= pBits_[arrayIndex]&mask1;
    BitmapUnit op2_1stPart= op2.pBits_[arrayIndex]&mask0;
    BitmapUnit op2_2ndPart= op2.pBits_[arrayIndex]&mask1;
    pBits_[arrayIndex] = op1_1stPart | op2_2ndPart;
    op2.pBits_[arrayIndex] = op2_1stPart | op1_2ndPart;
    /* 3. count # of 1 ( numOf1_ ) */
    this->countNumOf1();        /* after crossover, # of 1 was changed, so count it again */
    op2.countNumOf1();
}

bool    Bitmap::operator < ( const Bitmap & longerBmp) const {
    for (BitIdx_t i=0;i<unitNum_;i++) {
        if( (pBits_[i] & longerBmp.pBits_[i]) != pBits_[i] ) {
            return false;
        }
    }
    return true;
}

ostream& operator <<( ostream &os , const Bitmap & rhs){
    /*for(int i=0; i<rhs.bitNum_; i++ ) {
        os<<rhs.getBit( i );
    }*/
    bool isNull = true;
    for(BitIdx_t i=0; i<rhs.bitNum_; i++ ) {
        if ( rhs.getBit( i )) {
            os<<setw(3)<<i;
            isNull = false;
        }
    }
    if ( isNull )
        os<<"NULL";
    else
        os<<" ";
    return os;
}
```

```
}


/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : Bitmap.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/8 14:37:21
  Last Modified :
  Description   : definition of class Bitmap
  Function List :
  History       :
  1.Date        : 2012/7/8 14:37:21
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/

#ifndef BITMAP_H_
#define BITMAP_H_

#include <iostream>

namespace rough_set {

const int BIT_PER_BYTE = 8;
typedef   unsigned char BitmapUnit;
const int BIT_PER_UNIT = BIT_PER_BYTE*sizeof(BitmapUnit);
typedef     size_t      BitIdx_t;
class Bitmap {
friend std::ostream& operator <<( std::ostream &os , const Bitmap & rhs);
protected:
    BitIdx_t    bitNum_;
    BitIdx_t    unitNum_;
    BitIdx_t    numOf1_;            // number of 1 in this bitmap
    BitmapUnit  *pBits_;

private:
    void    getBitPosition( BitIdx_t bitIndex, BitIdx_t & arrayIndex , BitmapUnit& mask) const;
    void    getCrossOverPosNMask( BitIdx_t bitIndex,
                BitIdx_t & arrayIndex, BitmapUnit& mask) const ;
    void    countNumOf1();
public:
    Bitmap(BitIdx_t n);
    Bitmap( const Bitmap & one);
    virtual ~Bitmap();

    bool    getBit ( BitIdx_t bitIndex ) const ;
    void    setBit ( BitIdx_t bitIndex) ;
    void    resetBit ( BitIdx_t bitIndex );
    void    reverseBit( BitIdx_t bitIndex );
    void    crossOver( Bitmap& op2 , BitIdx_t crossPoint);
    void    resetAll ( ) {
        memset(pBits_,0, unitNum_* sizeof(BitmapUnit));
    }
    BitIdx_t    getBitNum() const {
        return bitNum_;
    }
    BitIdx_t     getNumOf1()  const {
        return numOf1_;
    }

    // bmp1 < bmp2 means bmp1 included in bmp2
    bool    operator < ( const Bitmap & longerBmp) const ;

};
```

```
}
#endif /* BITMAP_H_ */

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : DataSource.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/6/30 23:41:33
  Last Modified :
  Description   : implementation of class DataSource
  Function List :
  History       :
  1.Date        : 2012/6/30 23:41:33
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/

#include    "DataSource.h"
#include    "Record.h"
#include       "DataTypeFactories.h"         // for DataTypeFactories
#include    <string>
#include    <exception>
#include       <sstream>
using namespace std;

namespace rough_set{

bool    DataSource::hasDecisionAttr_ = false;
DataTypeFactories   DataSource::dataTypeUsable_;
int     DataSource::decisionAttrIdx_ = DataSource::initDataTypeUsable();
//template<typename T> T RangeDataType<T>::lower_ = static_cast<T>(0);
//template<typename T> T RangeDataType<T>::upper_ = static_cast<T>(0);

int    DataSource::initDataTypeUsable(void ) {
    dataTypeUsable_.insert(make_pair(string("char"), new DataTypeFactory<char> ()));
    dataTypeUsable_.insert(make_pair(string("string"), new DataTypeFactory<string>()));
    dataTypeUsable_.insert(make_pair(string("int"), new RangeDataTypeFactory<int>()));
    dataTypeUsable_.insert(make_pair(string("short"), new RangeDataTypeFactory<short>()));
    dataTypeUsable_.insert(make_pair(string("float"), new RangeDataTypeFactory<float>()));
    dataTypeUsable_.insert(make_pair(string("double"), new RangeDataTypeFactory<double>()));
    return 0;
}

void    DataSource::openDataFile( const   char * fileName){
    inDtFile_.open( fileName);
    if ( inDtFile_.fail()) {
        throw runtime_error(string("File open failed ")+string( fileName));
    } else {
        fileName_ = fileName;
    }
    //getDataStructDef();   //called by getTrainingData
}

inline void DataSource::expect( string token, string expectStr ) {
    if ( token != expectStr ){
               //ostringstream line;
               //line<<__LINE__;
        throw runtime_error("expect token \'"+expectStr+"\' at "+token
            +"\n Code: "+__FILE__+" "+__FUNCTION__/*+" "+line.str()*/ );
    }
}

void    DataSource::getDataStructDef( void ){
    string token;
    inDtFile_>>token;
    expect(token,"struct");
```

```
    inDtFile_>>token;
    expect(token,"{");
    string type, name;
    while( !inDtFile_.eof()){
        inDtFile_>>type;
        if (type =="}")
            break;
        inDtFile_>>name;
        DataTypeFactories::iterator dtIt = dataTypeUsable_.find( type);
        string strLower;
        string strUpper;
        double  lower = 0, upper = 0;
        if ( dtIt == dataTypeUsable_.end() ) {
            throw runtime_error( string("Attribute type:\'") + type +"\' is not supported");
        } else {
            if ( dtIt->second->supportRange() ) {
                inDtFile_>> strLower;
                            inDtFile_>> strUpper;
                istringstream   tmp(strLower);
                tmp>> lower;
                istringstream   tmp2(strUpper);
                tmp2>> upper;
                cout<<"type "<<type<<" supports range: v"<<lower<<" to v+"<<upper<<endl;
            }
        }
        vDataStructDef_.push_back( TblFields(type,name, lower, upper) );
        inDtFile_>>token;
        expect(token,";");
    }
}

void    DataSource::getAttributes( Attributes & vAttr ){
    DataTypeFactories::iterator dtIt;
    typedef vector<Attribute> VecAttributes;
    VecAttributes  vecDbAttributes;
    for ( VecFields::iterator  dbStructIt = vDataStructDef_.begin();
                            dbStructIt!= vDataStructDef_.end();
                            dbStructIt++ ) {
        if( (dtIt= dataTypeUsable_.find( dbStructIt->fdType ))!= dataTypeUsable_.end()){
             vAttr.push_back(Attribute( dbStructIt->fdName, dtIt->second,
                                        dbStructIt->lower_, dbStructIt->upper_ ) );
             int colIndex = dbStructIt- vDataStructDef_.begin();
             name2Col_.insert( make_pair( dbStructIt->fdName, colIndex));
        } else {
            cout<<"Data type : "<< dbStructIt->fdType<<" is not usable"<<endl;
        }
    }
    if ( vAttr.size()<1) {
        throw runtime_error("Wrong data structure definition");
    }
    AttributeSubSet::setTotAttrNum(vAttr.size());
}

int     DataSource::getRecordNum(){
    string token;
    inDtFile_>>token;
    expect( token, "recordNum");
    inDtFile_>>token;
    expect( token, "=");
    int recNum;
    inDtFile_>>recNum;
    cout<<"Will read "<<recNum<<" rows from file "<<fileName_<<endl;
       return recNum;
}

void   DataSource::getValues(Attributes & vAttr, Universe& value ){
    int recNum = getRecordNum();
    for ( int i=0; i< recNum ; i++) {
        Record * newRow = new Record();         // create a vector hold one row of records
        value.push_back(newRow);
        DataTypeBase* pValue;
```

```
        for( Attributes::iterator   vDbAttrIt = vAttr.begin();
                                    vDbAttrIt!= vAttr.end();
                                    vDbAttrIt++ ) {
            newRow->push_back(pValue=vDbAttrIt->createValueObj());
            inDtFile_>> *pValue  ;
#ifdef XDEBUG
            cout<<"Input value of field "<< vDbAttrIt->getName()<<"   ";
            cout<<*pValue<<endl;
#endif
            // get the maxium width of this column
            ostringstream valueStr;             // each loop should clear it;
            valueStr<<*pValue;                  // or else , it will be appened
            int len = valueStr.str().size();
            if ( len> vDbAttrIt->getFieldLen() ) {
                vDbAttrIt->setFieldLen(len);
            }
            if ( inDtFile_.fail()) {
                cout<<"Reading rec#: "<< i<<" attribute #:"<< vDbAttrIt- vAttr.begin()<<endl;
                throw runtime_error(string("Error in read file : ")+fileName_);
            }
        }
    }
}


void    DataSource::read1Set( string token, VecAttrSubSets& vAttrSubsets){
    MapName2Col::iterator name2ColIt;
    if( (name2ColIt=name2Col_.find( token) )== name2Col_.end()) {
        throw runtime_error(string("Expect a column name near "+ token ));
    } else {
        AttributeSubSet *pNewSet= new AttributeSubSet();    // create a new attribute subset
        pNewSet->insert( name2ColIt->second );      // add this column # into the set
        vAttrSubsets.push_back(pNewSet);            // add this subset into the vector of subsets
        inDtFile_>>token;
        while( token !="}") {
            expect( token, ",");
            inDtFile_>>token;                       // read the name of the column
            if( (name2ColIt=name2Col_.find( token) )== name2Col_.end()) {
                throw runtime_error(string("Expect a column name near "+ token ));
            } else {
                pNewSet->insert( name2ColIt->second );      // add this column # into the set
            }
            inDtFile_>>token;                       // read } or ,
        }
    }
}

void   DataSource::getSubAttributes( VecAttrSubSets& vAttrSubsets){
    string token;
    inDtFile_>>token;
    expect( token, "subAttributes");
    inDtFile_>>token;
    expect( token, "{");
    inDtFile_>>token;
    if ( token == "{") {
        do {
            inDtFile_>>token;
            read1Set( token,vAttrSubsets ); // processed }
            inDtFile_>>token;
        } while ( token =="{");     // next subset
        expect(token,"}");
    } else {
        read1Set( token,vAttrSubsets );
    }
}

void   DataSource::getDecisionAttr( ){
    string token;
    inDtFile_>>token;
    expect( token, "DecisionAttr");
    inDtFile_>>token;
```

```
    expect( token, "=");
    inDtFile_>>token;
    MapName2Col::iterator name2ColIt;
    hasDecisionAttr_ = false;
    if ( token !="NULL") {
        if( (name2ColIt=name2Col_.find( token) )== name2Col_.end()) {
            cout<<"Warning, Decision Attribute:\'"<<token
                <<"\' is not in Data Structure definition"<<endl;
            cout<<"Assume Decision Attribute is NULL"<<endl;
        } else {
            hasDecisionAttr_ = true;
            decisionAttrIdx_ = name2ColIt->second;
            cout<<"Info: Decision Attribute :\'"<<token
                <<"\',Index= "<<decisionAttrIdx_<<endl;
        }
    } else {
        cout<<"Info: No decision Attribute"<<endl;
    }
}

int     DataSource::getPopuNum() {
    string token;
    inDtFile_>>token;
    expect( token, "PopuNum");
    inDtFile_>>token;
    expect( token, "=");
    int popuNum;
    inDtFile_>>popuNum;
    return popuNum;
}

bool    DataSource::getVerificationFlag() {
    string token;
    inDtFile_>>token;
    expect( token, "DoVerification");
    inDtFile_>>token;
    expect( token, "=");
    string yesOrNo;
    inDtFile_>>yesOrNo;
    return ( yesOrNo == "yes");
}

size_t  DataSource::getMaxORItemsSupport() {
    string token;
    inDtFile_>>token;
    expect( token, "MaxDecisionFuncORItems");
    inDtFile_>>token;
    expect( token, "=");
    size_t  data;
    inDtFile_>>data;
    return data;
}

/*
int  DataSource::getRangeValue() {
    string token;
    inDtFile_>>token;
    expect( token, "MatchRange");
    inDtFile_>>token;
    expect( token, "=");
    int   data;
    inDtFile_>>data;
    return data;
}
*/

bool    DataSource::getClassifyFlag() {
    string token;
    inDtFile_>>token;
    expect( token, "DoClassify");
    inDtFile_>>token;
```

```
    expect( token, "=");
    string yesOrNo;
    inDtFile_>>yesOrNo;
    return ( yesOrNo == "yes");
}

}
/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : DataSource.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/6/30 23:02:18
  Last Modified :
  Description   : definition of class DataSource where IS get data from
  Function List :
  History       :
  1.Date        : 2012/6/30 23:02:18
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#ifndef DATA_SOURCE_H_
#define DATA_SOURCE_H_

#include    "Attribute.h"
#include       "Record.h"
#include       "Universe.h"
#include       "AttributeSubSet.h"    // for VecAttrSubSets&
#include       "DataTypeFactories.h"  // for DataTypeFactories   dataTypeUsable_;
#include    <vector>            // for data structure def
#include    <string>            // for mame
#include    <fstream>           // for input file stream
namespace rough_set {           // stuff used in rough sets peojects

class DataSource {
public:
    ~DataSource() {
        inDtFile_.close();
    }
    void    openDataFile( const   char * fileName);
    void    getDataStructDef(void);
    void    getAttributes( Attributes & vAttr );
    void    getValues(Attributes & vAttr, Universe& value );
    void    getSubAttributes( VecAttrSubSets& vAttrSubsets);
    void    getDecisionAttr();
    int     getPopuNum();
    bool    getVerificationFlag();
    bool    getClassifyFlag();
    size_t  getMaxORItemsSupport();
    //int     getRangeValue();
    static  bool hasDecisionAttr(){
        return  hasDecisionAttr_;
    }
    static  int  getDecisionAttrIdx(){
        return  decisionAttrIdx_;
    }
    static  int    initDataTypeUsable(void);

private:
    std::string     fileName_;
    std::ifstream   inDtFile_;       // input data file
    struct TblFields {
        const std::string  fdType;
        const std::string  fdName;
        double  lower_;             // to support range
        double  upper_;
        TblFields(      const std::string cstrType,
```

```
                        const std::string cstrName,
                        double lower, double upper ) :
                            fdType( cstrType), fdName(cstrName),
                            lower_(lower), upper_(upper)  {
        }
    };
    typedef std::vector<TblFields> VecFields;
    VecFields vDataStructDef_;

    typedef std::map<std::string, int>  MapName2Col;
    MapName2Col name2Col_;

    static DataTypeFactories   dataTypeUsable_;
    static  bool    hasDecisionAttr_;
    static  int     decisionAttrIdx_;
    int     getRecordNum();
    void    read1Set( std::string token, VecAttrSubSets& vAttrSubsets);
    inline void    expect( std::string keyWords, std::string expectStr );

};

}

#endif /* DATA_SOURCE_H_ */


/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : DataType.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/6/30 21:27:46
  Last Modified :
  Description   : definition of class template DataType for value type in
                  Rough Set Information System
  Function List :
  History       :
  1.Date        : 2012/6/30 21:27:46
    Author      : Shuang Wang
    Modification: Created file

******************************************************************************/
#ifndef DATA_TYPE_H_
#define DATA_TYPE_H_
#include    "Attribute.h"       // if Attribute.h include DataType.h will get a include loop
#include    <iostream>
#include    <vector>
#include       <set>                          // for subset of attributes
namespace rough_set {           // stuff used in rough sets peojects

class DataTypeBase {                // base for all data types existed in Data Source
public:
    friend std::ostream& operator<< ( std::ostream& os, const DataTypeBase& id );
    friend std::istream& operator>> ( std::istream& is, DataTypeBase& id );

       DataTypeBase( ) {
    }
    virtual ~DataTypeBase() {};
    virtual size_t valueSize()  = 0; // for test
    virtual bool    operator!= ( const DataTypeBase& rhs) const =0;
    virtual bool    operator== ( const DataTypeBase& rhs) const =0;
    virtual  bool match( const DataTypeBase& rhs, const Attribute& attr) = 0;   //support range
private:
    virtual std::istream& input( std::istream& is )=0;
    virtual std::ostream& output( std::ostream& os )const =0;
};

inline std::ostream & operator<< ( std::ostream & os, const DataTypeBase& id ){
```

```
    id.output( os );
    return os;
}

inline std::istream & operator>> ( std::istream & is, DataTypeBase& id ){
    id.input( is );
    return is;
}

template < typename T>          // class for data types existed in Data Source
class DataType: public DataTypeBase{

public  :
    DataType(): value_() {     //inline constructor
    }
    virtual ~DataType() {
    }

    virtual size_t valueSize() {
        return sizeof(value_);
    }

    virtual bool    operator!= ( const DataTypeBase& rhs)  const{
               const DataType<T>& drhs = dynamic_cast<const DataType<T>& >( rhs );
        //return ( value_ != drhs.value_);
        return !( *this == drhs );
    }

    virtual bool    operator== ( const DataTypeBase& rhs)  const{
               const DataType<T>& drhs = dynamic_cast<const DataType<T>& >( rhs );
        return ( value_ == drhs.value_);
    }

    virtual  bool match( const DataTypeBase& rhs, const Attribute& attr) {
               const DataType<T>& drhs = dynamic_cast<const DataType<T>& >( rhs );
        return ( value_ == drhs.value_);
    }
    virtual std::istream& input( std::istream& is ) {
       is>>value_;
       return is;
    }

    virtual std::ostream& output( std::ostream& os ) const {
        os<<value_;
        return os;
    }

protected :
    T  value_;                // hold the data of this type for a record in data source
};


template < typename T>
class RangeDataType: public DataType<T> {   //class for range match

public:
    RangeDataType() {
        }
    virtual ~RangeDataType() {
    }

    #if 0
    virtual void setRange( double lower, double upper) {
        lower_ = static_cast<T>( lower );
        upper_ = static_cast<T>( upper );
    }

    virtual bool    operator== ( const DataTypeBase& rhs)  const{
               const RangeDataType<T>& drhs = dynamic_cast<const RangeDataType<T>& >( rhs );
        const T delta = upper_ - lower_;
        return ( (( value_ <= drhs.value_) && (drhs.value_ -value_<= delta )) ||
```

```
            (( value_ >= drhs.value_) && (value_ - drhs.value_ <= delta )) );
    }
    #endif
    virtual  bool match( const DataTypeBase& rhs, const Attribute& attr) {
               const RangeDataType<T>& drhs = dynamic_cast<const RangeDataType<T>& >( rhs );
        double  upper = attr.getUpper();
        double  lower = attr.getLower();
        const T delta = static_cast<T>(upper - lower);
        return ( (( value_ <= drhs.value_) && (drhs.value_ - value_<= delta )) ||
            (( value_ >= drhs.value_) && (value_ - drhs.value_ <= delta )) );
    }

private:
    #if 0
    static T   lower_;             // real range = [value_+lower_ , value_+upper_]
    static T   upper_;
    #endif
};

//#define XDEBUG
}
#endif /* DATA_TYPE_H_ */

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : DataTypeFactories.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/6/25 23:52:44
  Last Modified :
  Description   : definition of data type factories and its derived class
------------------template for each data type
  Function List :
  History       :
  1.Date        : 2012/6/25 23:52:44
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#ifndef     DATA_TYPE_FACTORY_H_
#define     DATA_TYPE_FACTORY_H_

#include    "DataType.h"
#include    <map>
#include    <string>

namespace rough_set {           // stuff used in rough sets peojects

//************************************************************
//
//     definition of DataTypeFactoryBase & DataTypeFactory
//
//************************************************************
class DataTypeFactoryBase{
    public :
        virtual DataTypeBase * create() = 0;
        virtual bool    supportRange() = 0;
};

template< typename T >
class DataTypeFactory : public DataTypeFactoryBase {
public :
    virtual DataTypeBase * create(  ) {
        return (new DataType<T>(  ));     // this is amazing
    }
    virtual bool    supportRange() { return false ; }
};
```

```
template< typename T >
class RangeDataTypeFactory : public DataTypeFactoryBase {
public :
    virtual DataTypeBase * create(  ) {
        return (new RangeDataType<T>(  ));     // this is amazing 2.
    }
    virtual bool    supportRange() { return    true ; }
};

typedef std::map<std::string, DataTypeFactoryBase*> DataTypeFactories;

}

#endif

/*************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *************************************************************************
  File Name     : DecisionValueSet.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/29 13:37:52
  Last Modified :
  Description   : implementation of class DecisionValueSet
  Function List :
  History       :
  1.Date        : 2012/7/29 13:37:52
    Author      : Shuang Wang
    Modification: Created file

*************************************************************************/
#include    "DecisionValueSet.h"
#include    "DataType.h"            // for DataTypeBase

using namespace std;

namespace rough_set {
bool DecisionValueSet::operator == ( const DecisionValueSet& rhs)  const {
    if ( empty() || rhs.empty()) {
        throw runtime_error("Decision Value Set should not be empty");
    }
    if (size()!= rhs.size())
        return false;
    for( DecisionValueSet::iterator it = begin(); it!=end();it++) {
        bool    curIn = false;
        const   DataTypeBase *  pCurDaValue = *it;
        for( DecisionValueSet::iterator it2 = rhs.begin(); it2!=rhs.end();it2++) {
            const DataTypeBase *  pCurDaValue2 = *it2;
            if ( *pCurDaValue == *pCurDaValue2 ) {
                curIn = true;
                break;
            }
        }
        if ( ! curIn  )                 // each item in set1 should be in set2
            return false;
    }
    return true;
}

bool DecisionValueSet::has(const DataTypeBase * pCurDaValue) {
    if ( empty() )
        return false;
    bool found = false;
    for ( DecisionValueSet::iterator it = begin();it != end(); it++ ) {
        const DataTypeBase * pExistedDaValue = *it;
        if ( * pExistedDaValue == * pCurDaValue ) {
            found = true;
            break;
        }
```

```
    }
    return found;
}

void DecisionValueSet::insertUnique(const DataTypeBase * pCurDaValue) {
    if ( ! has(pCurDaValue)) {
        insert( pCurDaValue);
    }
}

ostream & operator<< ( ostream & os , const DecisionValueSet & dvs){
    os<<"\t[ ";
    if (!dvs.empty()){
        for ( DecisionValueSet::const_iterator it = dvs.begin();
            it != dvs.end(); it++ ) {
            os<<*(*it)<<" ";
        }
    }
    os<<"]";
       return os;
}


}

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : DecisionValueSet.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/29 13:35:19
  Last Modified :
  Description   : definition of class DecisionValueSet
  Function List :
  History       :
  1.Date        : 2012/7/29 13:35:19
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#ifndef DECISION_VALUE_H_
#define DECISION_VALUE_H_

#include    <set>
#include    <exception>
#include    <iostream>

namespace rough_set {
class DataTypeBase;
class DecisionValueSet: public std::set<const DataTypeBase*> {
friend std::ostream & operator<< ( std::ostream & os , const DecisionValueSet & dvs);
public:
    bool    operator == ( const DecisionValueSet& rhs)  const ;
    inline  bool has(const DataTypeBase * pCurDaValue) ;
    void    insertUnique(const DataTypeBase * pCurDaValue) ;
};

}

#endif /* DECISION_VALUE_H_ */

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : DiscernBitmap.cpp
  Version       : Initial Draft
```

```
  Author        : Shuang Wang
  Created       : 2012/7/13 22:45:58
  Last Modified :
  Description   : implementation of class DiscernBitmap
  Function List :
  History       :
  1.Date        : 2012/7/13 22:45:58
    Author      : Shuang Wang
    Modification: Created file

*******************************************************************************/
#include    "DiscernBitmap.h"
#include    <sstream>               // for ostringstream
#include    <iomanip>               // for setw
#include    <exception>
using namespace std;

namespace rough_set {

BitIdx_t    DiscernBitmap::outputWidth_ = 0;

DiscernBitmap::DiscernBitmap(BitIdx_t n):Bitmap(n),state_(BS_INTACT){
}

ostream& operator <<( ostream &os , const DiscernBitmap & rhs){
    bool isNull = true;
    ostringstream tmpStr;
    tmpStr<<"<";
    tmpStr<<static_cast<const Bitmap&>(rhs);       // force it to call operator<< of base class
Bitmap
    tmpStr<<">";
    os<<setw(rhs.outputWidth_)<<tmpStr.str();
    return os;
}

}

/*******************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *******************************************************************************
  File Name     : DiscernBitmap.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/13 22:37:12
  Last Modified :
  Description   : definition of class DiscernBitmap,
  Function List :
  History       :
  1.Date        : 2012/7/13 22:37:12
    Author      : Shuang Wang
    Modification: Created file

*******************************************************************************/
#ifndef     DISCERN_BITMAP_H_
#define     DISCERN_BITMAP_H_

#include    "Bitmap.h"
namespace   rough_set {
enum BitmapState {
    BS_INTACT = 0 ,
    BS_REMOVE  ,
    BS_KEEP
} ;

class   DiscernBitmap:public Bitmap {
friend std::ostream& operator <<( std::ostream &os , const DiscernBitmap & rhs);
    static  BitIdx_t     outputWidth_;
    BitmapState     state_;
public:
```

```
        DiscernBitmap(BitIdx_t n);           // used by DiscernEntry ( typedef)
    static void setOutPutWidth( BitIdx_t width) {
        outputWidth_ = width ;
    }
    void    setState(BitmapState iState) {
        state_ = iState;
    }
    BitmapState getState() {
        return state_;
    }

    //std::string  bmp2LogicExpress();
};

}
#endif /* DISCERN_BITMAP_H_ */

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : DiscernFunc.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/15 18:46:14
  Last Modified :
  Description   : implementation of DiscernFunc
  Function List :
  History       :
  1.Date        : 2012/7/15 18:46:14
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#include    "DiscernFunc.h"
#include       "DiscernBitmap.h"
#include       <iostream>
#include       <string>

using namespace std;

namespace rough_set {
/*****************************************************************************
*                                                                            *
*                   implementation of DiscernFunc                            *
*                                                                            *
*****************************************************************************/
#if 0
//the  DicsernBitmap * of DecisionFunc is a reference to the pointer in
// DiscernMatrix , so it does not need destructor
DecisionFunc::~DecisionFunc(){
    for ( DecisionFunc::iterator dfIt = begin(); dfIt != end(); dfIt++ ) {
        delete (*dfIt);
    }
}
#endif

void    DiscernFunc::removeInclude(const DiscernBitmap & shorterBmp ){
    for(  DiscernFunc::iterator it = begin(); it!= end(); it++) {
        DiscernBitmap & curBmp = **it;
        if ( curBmp.getState()== BS_INTACT ) {
            if ( shorterBmp < curBmp ) {   // x1<x2 , means x1 is included in x2
                curBmp.setState(BS_REMOVE);
            }
        }
    }
}

void    DiscernFunc::simplize(unsigned subAttrNum){
    bool    finished =false;
```

```
    BitIdx_t     expectedNumOf1 = 0;
    while(++expectedNumOf1<=subAttrNum) {
        for(  DiscernFunc::iterator it1 = begin(); it1!= end(); it1++) {
            DiscernBitmap & curBmp = **it1;
            if ((curBmp.getState() == BS_INTACT ) &&    // was n't peocessed before
                ( curBmp.getNumOf1()== expectedNumOf1)) {
                curBmp.setState(BS_KEEP);
                removeInclude( curBmp);
            }
        }
    }
}

void    DiscernFunc::copyTo(OrAndExpr & tgt){
    for(  DiscernFunc::iterator it1 = begin(); it1!= end(); it1++) {
        DiscernBitmap & curBmp = **it1;
        if (curBmp.getState() == BS_KEEP ) {
            tgt.push_back( new OrExpr(*it1) );           //put the pointer to DiscernBitmap
        }
    }
}

}

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : DiscernFunc.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/15 18:38:48
  Last Modified :
  Description   : definition of class DiscernFunc (Discern function )
  Function List :
  History       :
  1.Date        : 2012/7/15 18:38:48
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/

#ifndef DECISION_FUNC_H_
#define DECISION_FUNC_H_
#include       "OrAndExpr.h"         // for trans2AndOrExpr( AndOrExpr &);
#include    <vector>

namespace rough_set {

class   DiscernBitmap;          //not include "DiscernBitmap"
class DiscernFunc :public std::vector<DiscernBitmap*> {

//friend std::ostream& operator <<( std::ostream &os , const DiscernFunc & rhs);

    void    removeInclude(const DiscernBitmap & shorterBmp );

public:
    //~DiscernFunc();  //the  DicsernBitmap * of DiscernFunc is a reference to the pointer in
                       // DiscernMatrix , so it does not need destructor
    void    simplize(unsigned subAttrNum);
    void    copyTo(OrAndExpr & tgt);
};

}
#endif /* DECISION_FUNC_H_ */
/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
```

```
  File Name     : DiscernMap.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/16 23:55:50
  Last Modified :
  Description   : implementation of class DiscernMap
  Function List :
  History       :
  1.Date        : 2012/9/16 23:55:50
    Author      : Shuang Wang
    Modification: Created file

******************************************************************************/

#include    "DiscernMap.h"
#include    <string>
using namespace std;
namespace rough_set{

ostream & operator<< ( ostream & os , const DiscernMap & dm){
    DiscernMap::const_iterator it =dm.begin();
    for ( ;it!= dm.end(); it++) {
        os<<it->first<<"\t-->"<<it->second<<endl;
    }
    os<<"map size="<< dm.size()<<endl;
       return os;
}

}

/******************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : DiscernMap.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/16 23:51:20
  Last Modified :
  Description   : definition of DiscernMap
  Function List :
  History       :
  1.Date        : 2012/9/16 23:51:20
    Author      : Shuang Wang
    Modification: Created file

******************************************************************************/
#ifndef DISCERN_MAP_H_
#define DISCERN_MAP_H_

#include    "DecisionValueSet.h"
#include    <map>
#include    <iostream>

namespace rough_set {
class DiscernMap : public std::map<std::string, DecisionValueSet> {
    friend std::ostream & operator<< ( std::ostream & os , const DiscernMap & dm);
};

}

#endif

/******************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : DiscernMatrix.cpp
  Version       : Initial Draft
```

```
  Author        : Shuang Wang
  Created       : 2012/7/8 14:40:46
  Last Modified :
  Description   : implementation of class DiscernMatrix
  Function List :
  History       :
  1.Date        : 2012/7/8 14:40:46
    Author      : Shuang Wang
    Modification: Created file

******************************************************************************/
#include    "DiscernMatrix.h"
#include       "Record.h"                          // for intermidiate variable
#include    "IndRelations.h"         // for output [x1]...
#include       "DiscernFunc.h"
#include    <sstream>               // for ostringstream
#include       <iomanip>
#include    <exception>

using namespace std;
namespace rough_set {

DiscernEntries::~DiscernEntries() {
    for( DiscernEntries::iterator it = begin(); it!= end(); it++) {
        delete (*it);
    }
}

ostream& operator <<( ostream &os , const DiscernEntries & rhs){
    for( DiscernEntries::const_iterator it = rhs.begin();it!= rhs.end(); it++) {
        os<<*(*it)<<" ";
    }
    os<< endl;
    return os;
}

/*****************************************************************************
*                                                                            *
*                   implementation of DiscernMatrix                          *
*                                                                            *
*****************************************************************************/
DiscernMatrix::~DiscernMatrix() {
    for ( DiscernMatrix::iterator it = begin(); it!= end(); it++) {
        delete(*it);
    }
}

void DiscernMatrix::getAllDiscernBmp( DiscernFunc & discernBmp ) {
    for( DiscernMatrix::iterator it = begin(); it!=end(); it++) {
        DiscernEntries & row = *(*it);
        for( DiscernEntries::iterator it2 = row.begin();it2!= row.end(); it2++) {
            if ( ( *it2)->getNumOf1()!=0 ){     //it is not NULL
                discernBmp.push_back( *it2 );
            }
        }
    }
}

void DiscernMatrix::calcDiscernFunction(   unsigned subAttrNum, OrAndExpr & df ){
    DiscernFunc origin;
    getAllDiscernBmp( origin );
    origin.simplize(subAttrNum);
    origin.copyTo(df);
}

ostream& operator <<( ostream &os , const DiscernMatrix & rhs){
    if ( rhs.size()> (1<<20) )  {
        os<<"Too many elements in Matrix "<<rhs.size()<<" skip it\'s output"<<endl;
        return os;
    }
    DiscernEntry::setOutPutWidth( rhs.fieldWidth_);
```

```
    // output the title row [X1]   [X2]....
    IndRelation & curIndRelation = *( rhs.pMyIndRelation_);
       os<<setw(rhs.fieldWidth_)<<'*'<<' ';            // do not merge the space to '*'
    for ( IndRelation::const_iterator it =curIndRelation.begin();
            it!= curIndRelation.end(); it++ ) {
        ostringstream osStr;
        osStr<<'['<<*(*(*((*it)->begin()))->begin())<<']';   // this first record is
representative member
                                                // the first attribute of this member is X1
        os<<setw(rhs.fieldWidth_)<<osStr.str()<<" ";
    }
       os<<endl;
    for ( DiscernMatrix::const_iterator it = rhs.begin();it!= rhs.end(); it++) {
        int curEcIndex = it - rhs.begin();
        // output the title of this row  [Xi]
        ostringstream osStr;
        osStr<<'['<<*(*(*(curIndRelation[curEcIndex])->begin())->begin())<<']';   // this first
record is representative member
        os<<setw(rhs.fieldWidth_)<<osStr.str()<<' ';
        os<<*(*it);
    }
    os<<endl;
    return os;
}

/*****************************************************************************
*                                                                            *
*                  implementation of VecDiscernMatrix                          *
*                                                                            *
*****************************************************************************/
VecDiscernMatrix::~VecDiscernMatrix(){
    for ( VecDiscernMatrix::iterator vecIt = begin(); vecIt != end(); vecIt++ ) {
        delete (*vecIt);
    }
}

void VecDiscernMatrix::calcDiscernFunctions( const VecAttrSubSets& rvAttrSubSet,VecDecisionFunc &
vdf ){
    for ( VecDiscernMatrix::iterator vecIt =begin(); vecIt != end(); vecIt++ ) {
        int index = vecIt - begin();
        OrAndExpr * pDiscernFunc = new OrAndExpr();
        (*vecIt)->calcDiscernFunction( (rvAttrSubSet.at(index))->size(),*pDiscernFunc);
        vdf.push_back(pDiscernFunc);
    }
}

ostream& operator <<( ostream &os , const VecDiscernMatrix & rhs){
    size_t matrixNum = rhs.size();
    os<<endl;
    if ( matrixNum > 1 ) {
        for ( VecDiscernMatrix::const_iterator vecIt = rhs.begin();
              vecIt != rhs.end(); vecIt++ ) {
            os<<"Discern Matrix for scenario "<< 1+vecIt- rhs.begin()<<" = "<<endl<<**vecIt;
        }
    } else {
        os<<"Discern Matrix = "<<endl<<**rhs.begin();
    }
    return os;
}

}

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : DiscernMatrix.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/8 14:42:22
```

```
  Last Modified :
  Description   : definition of class DiscernMatrix
  Function List :
  History       :
  1.Date        : 2012/7/8 14:42:22
    Author      : Shuang Wang
    Modification: Created file

**********************************************************************/

#ifndef DISCERN_MATRIX_H_
#define DISCERN_MATRIX_H_

#include    "DiscernBitmap.h"
#include    <vector>

namespace rough_set {

typedef DiscernBitmap  DiscernEntry;
class DiscernEntries:public std::vector<DiscernEntry*> {
friend std::ostream& operator <<( std::ostream &os , const DiscernEntries & rhs);

public:
    ~DiscernEntries();

};

class   IndRelation;
class   DiscernFunc;
class   OrAndExpr;
class DiscernMatrix : public std::vector<DiscernEntries*>{
    int fieldWidth_;
    IndRelation * pMyIndRelation_;
friend std::ostream& operator <<( std::ostream &os , const DiscernMatrix & rhs);

public:
    ~DiscernMatrix();
    void associateIndRelation( IndRelation * itsIndRelation ) {
        pMyIndRelation_ = itsIndRelation;
    }
    void setFieldWidth( int width) {
        fieldWidth_ = width;
    }
    void getAllDiscernBmp( DiscernFunc & df );
    void calcDiscernFunction( unsigned , OrAndExpr & df );
};

class   VecAttrSubSets;
class   VecDecisionFunc;
class VecDiscernMatrix: public std::vector< DiscernMatrix*> {
friend std::ostream& operator <<( std::ostream &os , const VecDiscernMatrix & rhs);
public:
    ~VecDiscernMatrix();
    void calcDiscernFunctions( const VecAttrSubSets& rvAttrSubSet, VecDecisionFunc & vdf );
};

}
#endif /* DISCERN_MATRIX_H_ */

/**********************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 **********************************************************************
  File Name     : IndRelations.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/6 00:01:16
  Last Modified :
  Description   : implementation of classes related to IndRelation
  Function List :
```

```
  History       :
  1.Date        : 2012/7/6 00:01:16
    Author      : Shuang Wang
    Modification: Created file

*******************************************************************************/
#include "Record.h"
#include "IndRelations.h"
#include "DataSource.h"
#include <assert.h>
using namespace std;

namespace rough_set {
/*******************************************************************************
*                                                                              *
*                   implementation of IndRecs                        *
*                                                                              *
*******************************************************************************/
ostream & operator << ( ostream &os, const IndRecs& rhs) {
    os<<"\t{ ";
    Record* pEndRec = * rhs.rbegin();
    for(  IndRecs::const_iterator indRecIt = rhs.begin();
                indRecIt!=rhs.end(); indRecIt++ ) {
        os<< *((**indRecIt)[0]);
        if ( *indRecIt != pEndRec )
            os<<" , ";
    }
    os<<" } ";
    if ( DataSource::hasDecisionAttr()) {
        // output the Disicion value set of this EC
        os<< rhs.decisionValues_;
    }
    os<<endl;
    return os;
}


void IndRecs::insert( Record * pRecord ){
    set<Record*>::insert( pRecord);         // call base insert
    if ( DataSource::hasDecisionAttr()) {
        // add decision value into a set if it is not in this set.
        int daIndex = DataSource::getDecisionAttrIdx();
        const DataTypeBase * pCurDaValue = pRecord->at(daIndex);
        if (decisionValues_.empty()){
            decisionValues_.insert( pCurDaValue );
        } else {
            decisionValues_.insertUnique(pCurDaValue);
        }
    }
}

/*******************************************************************************
*                                                                              *
*                   implementation of IndRelation                          *
*                                                                              *
*******************************************************************************/
IndRelation::~IndRelation(){
    for ( IndRelation::iterator indRelIt = begin();
          indRelIt != end(); indRelIt++ ) {
#ifdef  XDEBUG
        cout<<"will delete pointer to IndRelation"<<**indRelIt;
#endif
        delete (*indRelIt);

    }
}

void IndRelation::calcDiscernMatrix( const AttributeSubSet& rAttrSubSet, DiscernMatrix & rMatix){
    int attrNum = rAttrSubSet.getTotAttrNum();
    rMatix.associateIndRelation(this);
    rMatix.setFieldWidth( (rAttrSubSet.size()+2)<<1);      // 2 inlcudes "<" and one space
```

```
    //int attrNum = rAttrSubSet.size();
    for ( IndRelation::iterator it1 = begin(); it1!=end(); it1++) {
        DiscernEntries * pNewEntryVec  = new DiscernEntries();
        rMatix.push_back(pNewEntryVec);
               IndRelation::iterator it2 = begin();
        for ( ; it2!=it1; it2++) {
            DiscernEntry * pNewDiscernEntry = new DiscernEntry(attrNum);
            pNewEntryVec->push_back( pNewDiscernEntry );
            // 1. if has decision attribute and
            //    the value of decision attribute for these two records are same
            //    then set the bitmap as NULL
            if ( DataSource::hasDecisionAttr() &&
                ((*it1)->decisionValues_== (*it2)->decisionValues_ ) ){
                pNewDiscernEntry->resetAll();
                continue;
            }
            // 2. normally , set the bit corresponding to the attribute
            //    that can distinguish these two EC
            const Record & rRecord1 = **((*it1)->begin());    //get the representative member of
this EC;
            const Record & rRecord2 = **((*it2)->begin());
            rRecord1.calcDiscernAttrBitmap( rRecord2, rAttrSubSet, pNewDiscernEntry);
        }
        assert(it2 == it1);
        // create a NULL element
        DiscernEntry * pNewDiscernEntry = new DiscernEntry(attrNum);
        pNewEntryVec->push_back( pNewDiscernEntry );
    }
}

ostream & operator << ( ostream &os, const IndRelation& rhs) {
    os<<"{ "<<endl;
    for ( IndRelation::const_iterator indRelIt = rhs.begin();
          indRelIt != rhs.end(); indRelIt++ ) {
        IndRecs& curSet= **indRelIt;
        os<< curSet;
    }
       os<<"}"<<endl<<endl;
    return os;
}
/*****************************************************************************
*                                                                            *
*                   implementation of VecIndRelation                          *
*                                                                            *
*****************************************************************************/
VecIndRelation:: ~VecIndRelation(){
    for ( VecIndRelation::iterator vecIndIt = begin();
          vecIndIt != end(); vecIndIt++ ) {
#ifdef  XDEBUG
        cout<<"will delete pointer to VecIndRelation"<<**vecIndIt;
#endif
        delete (*vecIndIt);
    }
}

void VecIndRelation::calcDiscernMatrixs(
                const VecAttrSubSets& rvAttrSubSet,
                VecDiscernMatrix & rvMatrix)    {
    for ( VecIndRelation::iterator vecIndIt = begin();
          vecIndIt != end(); vecIndIt++ ) {
        size_t curIndex = vecIndIt - begin();
        DiscernMatrix * pNewMatrix = new  DiscernMatrix();
        (*vecIndIt)->calcDiscernMatrix( *(rvAttrSubSet.at(curIndex)),*pNewMatrix);
        rvMatrix.push_back( pNewMatrix);
    }
}

ostream & operator << ( ostream &os, const VecIndRelation& rhs){
    size_t filterNum = rhs.size();
    os<<endl;
    if ( filterNum > 1 ) {
```

```
        for ( VecIndRelation::const_iterator vecIndIt = rhs.begin();
              vecIndIt != rhs.end(); vecIndIt++ ) {
            os<<"IND"<< 1+vecIndIt- rhs.begin()<<" = "<<**vecIndIt;
        }
    } else {
        os<<"IND = "<<**rhs.begin();
    }
    return os;
}

}
/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : IndRelations.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/5 23:50:28
  Last Modified :
  Description   : definition of classes related to IndRelation (Indiscern
------------------relation)
  Function List :
  History       :
  1.Date        : 2012/7/5 23:50:28
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#ifndef IND_RELATION_H_
#define IND_RELATION_H_

#include       "DiscernMatrix.h"
#include    "AttributeSubSet.h"
#include    "DecisionValueSet.h"
#include    <set>
#include    <vector>
#include    <iostream>

namespace rough_set {           // stuff used in rough sets peojects
class Record;                   // for Record * , do not need to include Record.h
class DataTypeBase;             // for decisionValue_
//    typedef     std::set<Record*> IndRecs;       //Ind stands for indiscernibility
//    typedef     std::vector<IndRecs*> IndRelation;
//    typedef     std::vector<IndRelation*> VecIndRelation;
class IndRecs: public std::set<Record*> {
    friend  class IndRelation;
    friend std::ostream & operator << ( std::ostream &os, const IndRecs& rhs) ;
private:
    DecisionValueSet  decisionValues_;
public:
    void insert( Record * pRecord );
};

class IndRelation: public std::vector<IndRecs*> {
    friend std::ostream & operator << ( std::ostream &os, const IndRelation& rhs) ;
public:
    ~IndRelation();
    void calcDiscernMatrix( const AttributeSubSet& rAttrSubSet, DiscernMatrix & rMatix);
};

class VecIndRelation:public std::vector<IndRelation*>{
    friend std::ostream & operator << ( std::ostream &os, const VecIndRelation& rhs) ;
public:
    ~VecIndRelation();
    void calcDiscernMatrixs( const VecAttrSubSets& rvAttrSubSet, VecDiscernMatrix & rMatix);
};

}
```

```
#endif /* IND_RELATION_H_ */
/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : InformationSystem.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/1 09:51:54
  Last Modified :
  Description   : implementation of  class InformationSystem
  Function List :
  History       :
  1.Date        : 2012/7/1 09:51:54

    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#include    "InformationSystem.h"
//#include    "Record.h"          // alredy included in InformationSystem.h
#include    <vector>
#include    <iomanip>                // for setw
#include    <set>               // for recodr member of each equivalent set
#include       <sstream>                    // for ostreamstring
#include       <cassert>                    // for assert
#include    <ctime>             // for timeAndDate( ostream & os )
using namespace std;

namespace rough_set {           // stuff used in rough sets peojects

void    timeAndDate( ostream & os ) {
    const int TIME_LEN = 20;
       char tmpbuf[TIME_LEN];

    /* Set time zone from TZ environment variable. If TZ is not set,
     * the operating system is queried to obtain the default value
     * for the variable.
     */
    _tzset();

       /* Display operating system-style date and time. */
       _strdate_s( tmpbuf,TIME_LEN );
    os<<"date:"<<tmpbuf;

    _strtime_s( tmpbuf,TIME_LEN );
    os<<", time:"<<tmpbuf<<endl;
}

InformationSystem::~InformationSystem(){
    //releaseUniverseMem();
}

// read data structure definition and data/values from file
void InformationSystem::getTrainingData( const char * fileName ) {
    trainingDS_.openDataFile( fileName);
    trainingDS_.getDataStructDef();             // test file does not have this one
    trainingDS_.getAttributes( attributes_ );
    Record::associateAttribute( &attributes_ ); // for range match record need attribute
information
    trainingDS_.getSubAttributes(vAttrSubsets_);
    trainingDS_.getDecisionAttr();
    //matchRange_ = trainingDS_.getRangeValue();
    maxDecisionAndItems_ = trainingDS_.getMaxORItemsSupport();
    doVerification_ = trainingDS_.getVerificationFlag();
    //PopuNum should be put after Verification flag
    geneAlgo_.setPopuNum(trainingDS_.getPopuNum());
    trainingDS_.getValues(attributes_,trainingRecs_);
    doClassify_ = trainingDS_.getClassifyFlag();
    if ( doClassify_) {
```

```
        trainingDS_.getValues(attributes_,testRecs_);
    }
}

#if 0
void    InformationSystem::getTestData(const char * fileName) {
//  testDS_.openDataFile( fileName);
//   testDS_.getAttributes( attributes_ );
//   testDS_.getSubAttributes(vAttrSubsets_);
//   testDS_.getDecisionAttr();

    // !!!! put test data also in training file
    trainingDS_.getValues(attributes_,testRecs_);
}
#endif

void InformationSystem::dispTrainingData( ostream & os ) {
    os<<endl<<"Training Data:"<<endl;
    os<<attributes_;
    os<<trainingRecs_;
    os<<"Subset of Condition Attributes:"<<endl;
    os<<vAttrSubsets_;
}

void    InformationSystem::dispTestData( std::ostream & os ){
    os<<endl<<"Testing Data:"<<endl;
    os<<testRecs_;
}

void    InformationSystem::bmp2AttrSet( const Bitmap & rAttrBmp,
                                AttributeSubSet & rAttrSet ) {
    for ( BitIdx_t i=0; i<rAttrBmp.getBitNum(); i++) {
        if ( rAttrBmp.getBit(i)) {
            rAttrSet.insert(i);
        }
    }
}
void    InformationSystem::training( ostream & os ,const AndExpr& rAttrBmp, bool printRules ) {
    AttributeSubSet discernAttr;
    bmp2AttrSet( rAttrBmp, discernAttr);
    for( Universe::const_iterator it = trainingRecs_.begin();
                    it!= trainingRecs_.end(); it++) {
        ostringstream key ;
        // construct the keyword of the map
        for ( AttributeSubSet::const_iterator attrIt = discernAttr.begin();
                    attrIt!= discernAttr.end();  attrIt++) {
            key<<*((*it)->at(*attrIt));
        }

        // associate the decision result with the keyword
        assert( trainingDS_.hasDecisionAttr());
        const AttrIdx_t decisionAttrIdx = trainingDS_.getDecisionAttrIdx();
               DiscernMap::iterator dit = discernMap_.find( key.str() );
        if ( dit == discernMap_.end()){
            DecisionValueSet  newDecisionValueSet;
            newDecisionValueSet.insert( (*it)->at(decisionAttrIdx));
                        discernMap_.insert(make_pair(key.str(), newDecisionValueSet));
        } else {
            DecisionValueSet  & theDecisionValueSet = dit->second;
            theDecisionValueSet.insertUnique((*it)->at(decisionAttrIdx));
        }
    }
    if ( printRules)    {
        os<<discernMap_;
    }
}

void    InformationSystem::selfMatch( const AndExpr& rAttrBmp ){
    AttributeSubSet discernAttr;
    bmp2AttrSet( rAttrBmp, discernAttr);
    for( Universe::const_iterator it = trainingRecs_.begin();
```

```
                    it!= trainingRecs_.end(); it++) {
        ostringstream key ;
        // construct the keyword of the map
        for ( AttributeSubSet::const_iterator attrIt = discernAttr.begin();
                    attrIt!= discernAttr.end();  attrIt++) {
            key<<*((*it)->at(*attrIt));
        }

        // associate the decision result with the keyword
        assert( trainingDS_.hasDecisionAttr());
        const AttrIdx_t decisionAttrIdx = trainingDS_.getDecisionAttrIdx();
               DiscernMap::iterator dit = discernMap_.find( key.str() );
        if ( dit == discernMap_.end()){
            ++misMatch_;
        } else {
            DecisionValueSet  & theDecisionValueSet = dit->second;
            if ( theDecisionValueSet.has((*it)->at(decisionAttrIdx)))
                ++matchNum_;
            else
                ++misDecision_;
        }
    }
}

void    InformationSystem::verify1DecisionFunc(   ostream & os, const AndOrExpr & df ) {
    for ( AndOrExpr::const_iterator it= df.begin();  it!=df.end(); it++ ) {
#ifdef  EXACT_MATCH
        training( os, *it );
        selfMatch( *it);
        // reset map
        discernMap_.clear();
#else
        rangeMatch( trainingRecs_, *it, CHECK_DECISION );
#endif
    }
    os  <<"Verification result............"<<endl
        <<"# of ORed AND Expresion "<<df.size()
        <<", matched conditions:"<<matchCondition_
        <<", matched desicions: "<<matchNum_
        <<", mismatched records: "<<misMatch_
        <<", wrong decision #:"<<misDecision_
        <<endl;
}

size_t    InformationSystem::matchTestData( ostream & os , const AndExpr& rAttrBmp ,bool print )
{
    size_t matchNum = 0, mismatch = 0, matchDecision=0;
    AttributeSubSet discernAttr;
    bmp2AttrSet( rAttrBmp, discernAttr);
    for( Universe::const_iterator it = testRecs_.begin();
                    it!= testRecs_.end(); it++) {
        ostringstream key ;
        // construct the keyword of the map
        for ( AttributeSubSet::const_iterator attrIt = discernAttr.begin();
                    attrIt!= discernAttr.end();  attrIt++) {
            key<<*((*it)->at(*attrIt));
        }

        // find the decision result in the map
               DiscernMap::iterator dit = discernMap_.find( key.str() );
        if ( dit == discernMap_.end()){
            if ( print) {
                os<<**it<<" ---> NULL"<<endl;
            }
            ++mismatch;
        } else {
            if ( print) {
                os<<**it<<" ---> "<< dit->second<<endl;
            }
            ++matchNum;
            if ( trainingDS_.hasDecisionAttr()) {
```

```
                const AttrIdx_t decisionAttrIdx = trainingDS_.getDecisionAttrIdx();
                DecisionValueSet  & theDecisionValueSet = dit->second;
                if ( theDecisionValueSet.has((*it)->at(decisionAttrIdx))) {
                    ++matchDecision;
                }
            }
        }
    }

    if ( print) {
        os  <<matchNum<<" match records, "
            <<matchDecision<< " match decisions, decision match rate="
            <<static_cast<double>(matchDecision)*100.0/testRecs_.size()
            <<"%, "<<mismatch<<" mismatch records"<<endl;
    }
    return matchNum;
}


void    InformationSystem::classify4EachDecisionFunc(   ostream & os, const AndOrExpr & df ) {
    size_t  maxMatchNum = 0 ,maxMatchIdx;
    size_t  minRuleSize = size_t(-1), minRuleSizeIdx,minRuleSizeMatchNum=0;
    size_t  maxRuleSize = 0, maxRuleSizeIdx,maxRuleSizeMatchNum=0;
    size_t  curMatchNum;
    size_t  curIdx= 0;
    AndOrExpr::const_iterator maxIt = df.end();
    for ( AndOrExpr::const_iterator it= df.begin();  it!=df.end(); it++ , curIdx++) {
        training( os, *it );
        size_t ruleSize = discernMap_.size();
        curMatchNum = matchTestData( os,*it);
        if( ruleSize> maxRuleSize){
            maxRuleSize = ruleSize;
            maxRuleSizeIdx = curIdx;
            maxRuleSizeMatchNum = curMatchNum;
        }

        if ( minRuleSize > ruleSize ) {
            minRuleSize = ruleSize;
            minRuleSizeIdx = curIdx;
            minRuleSizeMatchNum = curMatchNum;
        }
        //cout<<"Loop "<< curIdx <<" minRulesize= "<<minRuleSize<< " rulSize= "<<ruleSize<<endl;
        if ( curMatchNum > maxMatchNum ) {
            maxMatchNum = curMatchNum;
            maxMatchIdx = curIdx;
            maxIt = it;
        }
        // reset map
        discernMap_.clear();
    }
    os<<"--------------- Classify result -----------------"<<endl;
    if ( maxIt != df.end() ) {
        training( os, *maxIt, true);
        curMatchNum = matchTestData( os, *maxIt, true);
        assert( curMatchNum == maxMatchNum );
        os<<"Match "<< curMatchNum <<" records in test data , AND expr idx = "<<
maxMatchIdx<<endl;
        os<<"Decision Attr is: "<< *maxIt <<endl;
        os<<"RuleSize = "<<discernMap_.size()<<endl;
    } else {
        os<<"No rules match any records in test data"<<endl;
    }
    os  <<"MinRuleSize= "<< minRuleSize <<", minRuleSizeIdx= "
        << minRuleSizeIdx<<", matched "<< minRuleSizeMatchNum<<" records in test data"<<endl;
    os  <<"MaxRuleSize= "<< maxRuleSize <<", maxRuleSizeIdx= "
        << maxRuleSizeIdx<<", matched "<< maxRuleSizeMatchNum<<" records in test data"<<endl;
    discernMap_.clear();        // prepare for gene algo to use discernMap_
}

void    InformationSystem::classify( ostream & os, const VecAndOrExpr & df  ){
    os<<"Started  classification @ ";
```

```
    timeAndDate(os);
    for( VecAndOrExpr::const_iterator it = df.begin();
                it != df.end();   it++ ) {
        const AndOrExpr & rDecisionFunc = **it;
        os<<"verification for scenario "<<it - df.begin()<<endl;
#ifdef  EXACT_MATCH
        classify4EachDecisionFunc( os, rDecisionFunc );
#else
        rangeClassify1DecisionFunc(os, rDecisionFunc );
#endif
    }
    os<<"Finished classification @ ";
    timeAndDate(os);
    os<<endl;
}

void    InformationSystem::calcIndRelation(){
    for( VecAttrSubSets::iterator vAttrSubSetIt = vAttrSubsets_.begin();
            vAttrSubSetIt !=vAttrSubsets_.end(); vAttrSubSetIt++ ) {
        Record::setColFilter( *vAttrSubSetIt  );
        trainingRecs_.calcIndRelation( indRelations_  );
    }
}

void    InformationSystem::calcDiscernMatrix() {
    indRelations_.calcDiscernMatrixs(vAttrSubsets_,vDiscernMatrix_);
}

void    InformationSystem::calcDiscernFunc(){
    vDiscernMatrix_.calcDiscernFunctions(vAttrSubsets_, vDiscernFunc_ );
}

void    InformationSystem::verifyDecisionFuncs(ostream & os) {
    os<<"Started  verification @ ";
    timeAndDate(os);
    for( VecAndOrExpr::const_iterator it = vAndOrExprs_.begin();
                it != vAndOrExprs_.end();   it++ ) {
        const AndOrExpr & rDecisionFunc = **it;
        os<<"verification for scenario "<<it - vAndOrExprs_.begin()<<endl;
        verify1DecisionFunc( os, rDecisionFunc );
    }
    os<<"Finished verification @ ";
    timeAndDate(os);
    os<<endl;
}

void    InformationSystem::roughSetAlgo( ostream & os ){
    dispTrainingData( os );
    calcIndRelation();
    os<< indRelations_;
    calcDiscernMatrix();
    os<< vDiscernMatrix_;
    calcDiscernFunc();
    os<< vDiscernFunc_<<endl;
    vDiscernFunc_.trans2AndOrExprs( os,vAndOrExprs_,maxDecisionAndItems_);
    os<< vAndOrExprs_;
    if ( doVerification_ )    verifyDecisionFuncs(os);
    if ( doClassify_ ) classify(os,vAndOrExprs_);
}

void    InformationSystem::geneticAlgo( ostream & os ){
    if (trainingDS_.hasDecisionAttr()) {  // genetic algo needs decision Attribute
        os<<"Genetic Algorithm Section..."<<endl;
        geneAlgo_.copySubAttr(vAttrSubsets_);
        geneAlgo_.calcDecisionTables( trainingRecs_ ,attributes_,
trainingDS_.getDecisionAttrIdx());
        geneAlgo_.outDecisionTables(os);
        geneAlgo_.start(os, attributes_ ,geneResult_ );
        if ( doClassify_ ) classify(os,geneResult_);
    }
}
```

```
void    InformationSystem::doCalculate( ostream & os ){
    roughSetAlgo( os);
    geneticAlgo(os);
}

}
/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : InformationSystem.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/6/30 22:10:56
  Last Modified :
  Description   : Definition of class InformationSystem
  Function List :
  History       :
  1.Date        : 2012/6/30 22:10:56
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/

#ifndef INFOR_SYSTEM_H_
#define INFOR_SYSTEM_H_

#include    "Attribute.h"
#include    "DataSource.h"
#include    "AttributeSubSet.h"     // for VecAttrSubSets
#include    "IndRelations.h"        // for VecIndRelation
#include    "DiscernMatrix.h"       // for VecDiscernMatrix
#include    "DiscernFunc.h"        // for VecDecisionFunc
#include    "AndOrExpr.h"
#include    "GeneAlgo.h"            //
#include       "DiscernMap.h"
#include    <iostream>
#include    <vector>
#include    <set>

namespace rough_set {               // stuff used in rough sets peojects

enum  DecisionOption {
    CHECK_DECISION,
    IGNORE_DECISION
};

class Record;                       // for indiscernibility relation

class InformationSystem {

public:
       InformationSystem():misMatch_(0), matchNum_(0),
        matchCondition_(0),misDecision_(0){}
       ~InformationSystem();
    void    getTrainingData(const char * filename);
    // void    getTestData(const char * filename = NULL);  merged to getTrainingData()
    void    doCalculate( std::ostream & os );
private:
    Attributes  attributes_;
    Universe    trainingRecs_;      // training records
    DataSource  trainingDS_;        // training data source
    Universe    testRecs_;      // training records
    DataSource  testDS_;        // training data source
    VecAttrSubSets      vAttrSubsets_;
    VecIndRelation      indRelations_;
    VecDiscernMatrix    vDiscernMatrix_;
    VecDecisionFunc     vDiscernFunc_;
    VecAndOrExpr        vAndOrExprs_;
```

```
    GeneAlgo            geneAlgo_;
    VecAndOrExpr        geneResult_;
    DiscernMap  discernMap_;
    size_t      misMatch_;
    size_t      matchNum_;
    size_t      matchCondition_;
    size_t      misDecision_;
    size_t      maxDecisionAndItems_;
    int         matchRange_;
    bool        doVerification_;
    bool        doClassify_;
//////////////////////////////////////////////////////
//            private member functions
//////////////////////////////////////////////////////
    void    roughSetAlgo( std::ostream & os );
    void    geneticAlgo( std::ostream & os );
    void    calcIndRelation();
    void    calcDiscernMatrix();
    void    calcDiscernFunc();
    //void    dispData( std::ostream & os );        //changed to dispTrainingData
    void    verifyDecisionFuncs(  std::ostream & os);
    void    verify1DecisionFunc(  std::ostream & os, const AndOrExpr &);
    void    dispTrainingData( std::ostream & os );
    void    dispTestData( std::ostream & os );
    void    bmp2AttrSet( const Bitmap & rAttrBmp, AttributeSubSet & rAttrSet );
    void    training( std::ostream & os ,const AndExpr&, bool print=false);
       void    selfMatch( const AndExpr& rAttrBmp );
    size_t  rangeMatch(const Universe& u,const AndExpr& rAttrBmp , DecisionOption dp );
    void    classify( std::ostream & os ,const VecAndOrExpr & df);
    void    classify4EachDecisionFunc( std::ostream & os,  const AndOrExpr & df  );
    void    rangeClassify1DecisionFunc(std::ostream & os,  const AndOrExpr & df);
    size_t  matchTestData( std::ostream & os , const AndExpr& rAttrBmp,bool print =false);
    size_t  rangeMatchTestData( std::ostream & os , const AndExpr& rAttrBmp );
};

}

#endif /* INFOR_SYSTEM_H_ */


/*****************************************************************************

                Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : OrAndExpr.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/23 22:02:40
  Last Modified :
  Description   : implementation of class OrAndExpr etc..
  Function List :
  History       :
  1.Date        : 2012/9/23 22:02:40
    Author      : Shuang Wang
    Modification: Created file

******************************************************************************/
#include    "OrAndExpr.h"
#include    "DiscernBitmap.h"           // for prDscnBmp_
#include       <cassert>                                    // for assert
using namespace std;

namespace rough_set {

extern void    timeAndDate( ostream & os );

BitIdx_t    OrExpr::getBitNum() const {
    return prDscnBmp_->getBitNum();
}
```

```
BitIdx_t     OrExpr::getNumOf1()  const {
    return prDscnBmp_->getNumOf1();
}

bool    OrExpr::runOutOfItems() const {
    return( curSelectedItemNum_ >= prDscnBmp_->getNumOf1());
}

BitIdx_t    OrExpr::getNextItem() {
    for (   BitIdx_t bitPos = curBitIdx_ ;
            bitPos<prDscnBmp_->getBitNum(); ++bitPos) {
        if ( prDscnBmp_->getBit( bitPos) ){
            curBitIdx_ = bitPos+1;  // go to next bit
            ++curSelectedItemNum_;
            return bitPos;
        }
    }
    throw runtime_error("Can not find a Item");
}

ostream& operator <<( ostream &os , const OrExpr & rhs){
    bool isNull = true;
    BitIdx_t numOf1Scanned=0 ,numOf1 = rhs.prDscnBmp_->getNumOf1() ;
    os<<"( ";
    for(BitIdx_t i=0; i<rhs.prDscnBmp_->getBitNum(); i++ ) {
        if ( rhs.prDscnBmp_->getBit( i )) {
            isNull = false;
            numOf1Scanned++;
            os<<i;
            if ( numOf1Scanned < numOf1){
                os<<" V ";
            } else {
                if ( numOf1Scanned == numOf1){
                    os<<" ";
                } else {                // too many 1s
                    throw runtime_error("Data inconsistant");
                }
            }
        }
    }
    if ( isNull ) {
        os<<" NULL ";
    }
    os<<")";
    return os;
}

/*****************************************************************************
*                                                                            *
*                   implementation of OrAndExpr                             *
*                                                                            *
*****************************************************************************/
bool    OrAndExpr::isOrItemsTooLarge( size_t max ){
/*    double  combination =1.0;
    for( OrAndExpr::const_iterator it = begin();it!= end(); it++) {
        combination *= (*it)->getNumOf1();
    }
    return (combination>max );
    */
    return( size() > 1000 );
}

void    OrAndExpr::randSelectAndOrExpr( AndOrExpr & rAndOrExpr ) {
    cout<<"Under Construction !"<<endl;
}

bool    OrAndExpr::selectAndItem(
                size_t level,           // the index of OR-sub expression from which we will
select an AND-sub expr
                const AndExpr& tmpExpr,        // partial result passed to next level, a copy
constructor generate a new obj each call
```

```
                AndOrExpr & result     //
        ){
    bool ret =false;
    if ( level>= size()){       //passed the last OR-AND sub-expr
        if ( allExprWDup_.size()< maxNum_) {
            allExprWDup_.push_back( tmpExpr);
        } else {
            return false ;      // cut the recursive
        }
        if ( result.find(tmpExpr) != result.end()) {
            ++dupilcation_;     // find a duplication
        }
        if ( result.size()< maxNum_ ) {
            result.insert( tmpExpr);
            return true;
        } else {
            return false;       // cut the recursive
        }
    } else {
        OrExpr  * pCurOrExpr = this->at(level);
        while(! pCurOrExpr->runOutOfItems() ) { // already select all OR items in this sub-expr
            BitIdx_t    nextIdx = pCurOrExpr->getNextItem();
            AndExpr     newExpr( tmpExpr);      // should be here,each time use a clean copy
            newExpr.setBit( nextIdx);
            if ( !(ret = selectAndItem(level+1, newExpr, result ) ))
                break;
        }
        pCurOrExpr->resetSelectVars();
        return ret;
    }
}

void    OrAndExpr::bruteForceExpr( AndOrExpr & rAndOrExpr ,size_t  max ) {
    assert(this->size()>0);
    BitIdx_t newBmpWith = this->at(0)->getBitNum();
    AndExpr     tmpExpr( newBmpWith );
    maxNum_ = max;
    selectAndItem( 0, tmpExpr, rAndOrExpr );
}

void    OrAndExpr::trans2AndOrExpr( ostream &os,AndOrExpr & rAndOrExpr , size_t max){
    if ( isOrItemsTooLarge( max )) {
        randSelectAndOrExpr(rAndOrExpr);
    } else {
        os<<"Started  recursive @ ";
        timeAndDate(os);
        bruteForceExpr( rAndOrExpr , max );
        os<<"Finished recursive @ ";
        timeAndDate(os);
    }
    bool  allIn = true;
    // verify all expressions be recorded in set
    for ( vector< AndExpr>::iterator it = allExprWDup_.begin();
            it!= allExprWDup_.end(); it++ ){
        if ( rAndOrExpr.find( *it) == rAndOrExpr.end()){
            os<<*it<<" does not exist in result set !!!"<<endl;
            allIn = false;
        }
    }
    if ( allIn) {
        os<<"All expressions in Vector are found in set successfully !"<<endl;
    }
    os<<"Finished transformation:"<<endl
        <<"Totally generate "<< allExprWDup_.size()<<" AND-exprs"<<endl
        <<"Find "<< dupilcation_ <<" duplications,"<<rAndOrExpr.size()
        <<" unique AND-exprs"<<endl;
    allExprWDup_.clear();
}

ostream& operator <<( ostream &os , const OrAndExpr & rhs){
    if ( rhs.size()==0 ) {
```

```
        os<<"Could not find discern function"<<endl;
        return os;
    }
    OrExpr *  pBmp = * rhs.rbegin();
    for( OrAndExpr::const_iterator it = rhs.begin();it!= rhs.end(); it++) {
        os<<*(*it);
        if ( *it != pBmp ) os<<" ^ "<<endl;
    }
    os<< endl;
    return os;
}

OrAndExpr::~OrAndExpr(){
    for ( OrAndExpr::iterator exprIt = begin(); exprIt != end(); exprIt++ ) {
        delete (*exprIt);
    }
}

/*****************************************************************************
*                                                                            *
*                  implementation of VecDecisionFunc                             *
*                                                                            *
*****************************************************************************/
ostream& operator <<( ostream &os , const VecDecisionFunc & rhs){
    size_t matrixNum = rhs.size();
    os<<endl;
    if ( matrixNum > 1 ) {
        for ( VecDecisionFunc::const_iterator vecIt = rhs.begin();
              vecIt != rhs.end(); vecIt++ ) {
            os<<"Discern Function for scenario "<< 1+vecIt- rhs.begin()<<" = "<<endl<<**vecIt;
        }
    } else {
        os<<"Discern Function = "<<endl<<**rhs.begin();
    }
    return os;
}

void VecDecisionFunc::trans2AndOrExprs( ostream & os , VecAndOrExpr & vAndOrExprs , size_t max) {
    for ( VecDecisionFunc::iterator vecIt = begin(); vecIt != end(); vecIt++ ) {
        AndOrExpr * pNewExpr = new AndOrExpr();
        (*vecIt)->trans2AndOrExpr( os,* pNewExpr , max);
        vAndOrExprs.push_back( pNewExpr);
    }
}

VecDecisionFunc::~VecDecisionFunc(){
    for ( VecDecisionFunc::iterator vecIt = begin(); vecIt != end(); vecIt++ ) {
        delete (*vecIt);
    }
}

}

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : OrAndExpr.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/23 21:49:25
  Last Modified :
  Description   : definition of OrAndExpr. OR-AND logic expression , get
                  from discern bitmap before transform to AND-OR logic expression
  Function List :
  History       :
  1.Date        : 2012/9/23 21:49:25
    Author      : Shuang Wang
    Modification: Created file
```

```
*****************************************************************************/
#ifndef OR_AND_EXPR_H_
#define OR_AND_EXPR_H_

#include    "Bitmap.h"
#include    "AndOrExpr.h"       //for trans2AndOrExpr( AndOrExpr & rAndOrExpr)
#include    <vector>

namespace rough_set {

class   DiscernBitmap ;
class   OrExpr {
friend std::ostream& operator <<( std::ostream &os , const OrExpr & rhs);
    const DiscernBitmap  *prDscnBmp_;      // just refer to a pointer of DiscernBitmap
                                            // do not create any instance
    BitIdx_t    curBitIdx_;
    BitIdx_t    curSelectedItemNum_;
public:
    OrExpr( const DiscernBitmap * pDscnBmp):
        prDscnBmp_( pDscnBmp),
        curBitIdx_(0),
        curSelectedItemNum_(0)   {
    }
       inline BitIdx_t    getBitNum() const ;
       inline BitIdx_t    getNumOf1() const;
    inline    bool    runOutOfItems() const;

    void    resetSelectVars() {
        curBitIdx_ = 0;
        curSelectedItemNum_ = 0;
    }

       BitIdx_t    getNextItem();

};

class   OrAndExpr : public std::vector<OrExpr*> {
friend std::ostream& operator <<( std::ostream &os , const OrAndExpr & rhs);
    bool    isOrItemsTooLarge( size_t max);
    void    randSelectAndOrExpr( AndOrExpr & rAndOrExpr );
    void    bruteForceExpr( AndOrExpr & rAndOrExpr   , size_t max );
    bool    selectAndItem( size_t level, const AndExpr& tmpExpr, AndOrExpr & result );
    std::vector< AndExpr>    allExprWDup_;   // all And expression with duplication
    size_t  dupilcation_;
    size_t  maxNum_;
public:
    OrAndExpr():dupilcation_(0) {}
    ~OrAndExpr();
    void    trans2AndOrExpr( std::ostream &os, AndOrExpr & rAndOrExpr , size_t max);
};


class VecDecisionFunc :public std::vector<OrAndExpr*> {
friend std::ostream& operator <<( std::ostream &os , const VecDecisionFunc & rhs);

public:
    ~VecDecisionFunc();
    void trans2AndOrExprs( std::ostream &os,VecAndOrExpr & vAndOrExprs ,size_t max);
};      // end of  VecDecisionFunc


}
#endif /* OR_AND_EXPR_H_ */

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : rangeMatch.cpp
  Version       : Initial Draft
```

```
  Author        : Shuang Wang
  Created       : 2012/10/18 23:51:40
  Last Modified :
  Description   : implementation of rangematch functions of class InformationSystem
  Function List :
  History       :
  1.Date        : 2012/10/18 23:51:40
    Author      : Shuang Wang
    Modification: Created file

******************************************************************************/
#include    "InformationSystem.h"
#include    <vector>
#include    <iomanip>                // for setw
#include    <set>               // for recodr member of each equivalent set
#include       <sstream>                    // for ostreamstring
#include       <cassert>                    // for assert
#include    <ctime>             // for timeAndDate( ostream & os )
#include    <string>
#include    <exception>
using namespace std;

namespace rough_set {           // stuff used in rough sets peojects

size_t    InformationSystem::rangeMatch(
                const Universe& beMatched,  //the data to be matched with training data
                const AndExpr& rAttrBmp ,
                DecisionOption desicionOption   // check decision attr or not
    ) {
    AttributeSubSet discernAttr;
    bmp2AttrSet( rAttrBmp, discernAttr);
    Record::setColFilter( & discernAttr );
    matchCondition_=matchNum_= misDecision_ = misMatch_ = 0;
    for( Universe::const_iterator itR1 = beMatched.begin();
                itR1!= beMatched.end(); itR1++) {
        bool    foundSameCondition = false;
        bool    foundSameDecision  = false;
        for( Universe::const_iterator itR2 = trainingRecs_.begin();
                    itR2!= trainingRecs_.end(); itR2++) {
            if ( **itR1 == **itR2 ) {
                foundSameCondition = true;
                if ( CHECK_DECISION == desicionOption ) {
                    if (  trainingDS_.hasDecisionAttr()) {
                        const AttrIdx_t decisionAttrIdx = trainingDS_.getDecisionAttrIdx();
                        if ( (*itR1)->at(decisionAttrIdx) ==
                             (*itR2)->at(decisionAttrIdx)) {
                            foundSameDecision = true;
                            break;
                        }
                    }
                }
            }
        }
        if ( foundSameCondition ) {
            ++matchCondition_;
            if ( foundSameDecision ) {
                ++matchNum_;
            } else {
                ++misDecision_;
            }
        } else {
            ++misMatch_;
        }
    }
    return matchCondition_;
}

size_t    InformationSystem::rangeMatchTestData( ostream & os , const AndExpr& rAttrBmp ) {
    matchCondition_=matchNum_= misDecision_ = misMatch_ = 0;
    AttributeSubSet discernAttr;
    bmp2AttrSet( rAttrBmp, discernAttr);
```

```
    Record::setColFilter( & discernAttr );
       AttrIdx_t decisionAttrIdx;
    int noDecesionCollision = 0;
    if (  trainingDS_.hasDecisionAttr()) {
               decisionAttrIdx = trainingDS_.getDecisionAttrIdx();
       } else {
           throw runtime_error(string("No decision attribute, can not do
classify")+__FUNCTION__);
       }
    for( Universe::const_iterator itR1 = testRecs_.begin();
                   itR1!= testRecs_.end(); itR1++) {
        bool    foundSameCondition = false;
        DecisionValueSet  curDecisionValueSet;
        for( Universe::const_iterator itR2 = trainingRecs_.begin();
                   itR2!= trainingRecs_.end(); itR2++) {
            if ( **itR1 == **itR2 ) {
                foundSameCondition = true;
                curDecisionValueSet.insertUnique( (*itR2)->at(decisionAttrIdx));
            }
        }
        if ( foundSameCondition ) {
            ++matchCondition_;
            os<<**itR1<<" ---> "<< curDecisionValueSet<<endl;
            if ( curDecisionValueSet.has((*itR1)->at(decisionAttrIdx))){
                ++matchNum_;
                if ( curDecisionValueSet.size()==1 ) {
                    ++noDecesionCollision;
                }
            } else {
                ++misDecision_;
            }
        } else {
            os<<**itR1<<" ---> NULL"<<endl;
            ++misMatch_;
        }
    }
    os  <<"Mathched "<<matchCondition_<<" records condition, "
        <<"matched "<<matchNum_<< " decisions, decision match rate="
        <<static_cast<double>(matchNum_)*100.0/testRecs_.size()
        <<"% "<<endl
        <<noDecesionCollision<<" no collision matches, rate="
        <<static_cast<double>(noDecesionCollision)*100.0/testRecs_.size()<<endl
        <<misMatch_<<" mismatch records"<<endl;
    return  matchCondition_;
}

void    InformationSystem::rangeClassify1DecisionFunc(   ostream & os, const AndOrExpr & df ) {
    size_t  maxMatchNum = 0 ,maxMatchIdx;
    size_t  curMatchNum;
    size_t  curIdx= 0;
    AndOrExpr::const_iterator maxIt = df.end();
    for ( AndOrExpr::const_iterator it= df.begin();  it!=df.end(); it++ , curIdx++) {
        curMatchNum = rangeMatch( testRecs_,*it, IGNORE_DECISION );
        os<<"AND expr idx:"<< curIdx <<", match = "<<curMatchNum<<endl;
        if ( curMatchNum > maxMatchNum ) {
            maxMatchNum = curMatchNum;
            maxMatchIdx = curIdx;
            maxIt = it;
        }
    }
    os<<"--------------- Classify result -----------------"<<endl;
    if ( maxIt != df.end() ) {
        curMatchNum = rangeMatchTestData( os, *maxIt );
        if( curMatchNum != maxMatchNum ) {
            throw runtime_error(string("Internal error @")+__FUNCTION__);
        }
        os<<"Match "<< curMatchNum <<" records in test data , AND expr idx = "<<
maxMatchIdx<<endl;
        os<<"Decision Attr is: "<< *maxIt <<endl;
    } else {
        os<<"No rules match any records in test data"<<endl;
```

```
    }
}


}
/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : Record.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/2 22:01:53
  Last Modified :
  Description   : implementation of class Record
  Function List :
  History       :
  1.Date        : 2012/7/2 22:01:53
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#include    "Record.h"
#include       "Attribute.h"
#include    "AttributeSubSet.h" // for AttributeSubSet
#include       <iostream>
#include    <iomanip>
#include       <string>
#include    <exception>

using namespace std;
namespace rough_set {           // stuff used in rough sets peojects

const AttributeSubSet*  Record::pColFilter_ = NULL;  // the pointer is not const
const Attributes * Record::pAttr_ = NULL;            // the content it points to is content
Record::~Record() {
    for( Record::iterator   recIt = begin(); recIt!= end(); recIt++ ) {
#ifdef XDEBUG
    cout<<"will delete obj which holds "<<**recIt<<endl;
#endif
        delete(*recIt) ;    //delete the obj of DataType<T>
    }
}

bool Record::operator == ( const Record & rhs){
    int index = 0;
//    for( Record::iterator   recIt = begin(); recIt!= end(); recIt++ ) {
    for ( AttributeSubSet::iterator filterIt = pColFilter_->begin();
            filterIt != pColFilter_->end(); filterIt++ ){
        index = *filterIt;
        if ( !pAttr_ ) {
            throw runtime_error(string("Internal error")+__FUNCTION__);
        }
        if (! (this->at(index))->match( *(rhs.at(index)), pAttr_->at(index))) {
            return false;
        }
    }
    return true;
}

void    Record::calcDiscernAttrBitmap( const Record & rRecord2,
            const AttributeSubSet & rAttrSubSet,DiscernEntry * pDiscernEntry) const {
    for ( AttributeSubSet::const_iterator colIt = rAttrSubSet.begin();
                colIt != rAttrSubSet.end(); colIt++ ){
        int colIndex = *colIt;
        DataTypeBase& rec1ColValue = *( this->at(colIndex));
        DataTypeBase& rec2ColValue = *( rRecord2.at(colIndex));
        /*
        if( rec1ColValue != rec2ColValue ) {
            pDiscernEntry->setBit( colIndex );
```

```
        }*/
        if (!( rec1ColValue.match( rec2ColValue, pAttr_->at( colIndex )))) {
            pDiscernEntry->setBit( colIndex );
        }
    }
}

ostream& operator << ( ostream & os, const Record & rhs ){
    for( Record::const_iterator recIt = rhs.begin(); recIt!= rhs.end(); recIt++ ) {
               int width = (rhs.pAttr_->at(recIt- rhs.begin())).getFieldLen();
        os<<setw(width)<< *(*recIt) ;
    }
       //os<<endl;
       return os;
}

}

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : Record.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/2 21:39:49
  Last Modified :
  Description   : definition of class record which is Value( Xi) for each
------------------Attributej
  Function List :
  History       :
  1.Date        : 2012/7/2 21:39:49
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#ifndef RS_RECORD_H_
#define RS_RECORD_H_

#include    "DataType.h"
//#include    "AttributeSubSet.h" // for AttributeSubSet
#include    "DiscernMatrix.h"   //for DiscernEntry typedef
#include    <vector>

namespace rough_set {           // stuff used in rough sets peojects

class AttributeSubSet;          // for not include AttributeSubSet.h
class Attributes;               // for not include Attributes.h

class Record  : public std::vector<DataTypeBase*> {
friend std::ostream& operator << ( std::ostream & os, const Record & rhs );
    bool    isAlreadyInOtherEqSet;
    static  const AttributeSubSet   *  pColFilter_;  // Which column/field will be considered
when compare 2 records
                                                     // when check two rows/records equivalent
    static  const Attributes * pAttr_;  // for get column width

public:
    Record ():isAlreadyInOtherEqSet(false),
        std::vector<DataTypeBase*>() {
    }
    ~Record ();

    bool    operator == ( const Record & rhs);

    bool    alreadyInOtherEqSet(){
        return isAlreadyInOtherEqSet;
    }

    void    setAlreadyInEqset() {
```

```
        isAlreadyInOtherEqSet = true;
    }

    void    resetAlreadyInEqset() {
        isAlreadyInOtherEqSet = false;
    }

    static void setColFilter( const AttributeSubSet * iFilter) {
        pColFilter_ = iFilter;           // copy iFilter to colFilter_
    }

    static void associateAttribute( const Attributes * ipAttr) {
        pAttr_ = ipAttr;           // copy iFilter to colFilter_
    }

    void    calcDiscernAttrBitmap( const Record & rRecord2,
            const AttributeSubSet & rAttrSubSet,DiscernEntry * pDiscernEntry) const;
};

}

#endif /* RS_RECORD_H_ */

/*************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ************************************************************************
  File Name     : DataTypeUtest.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/6/26 23:37:53
  Last Modified :
  Description   : unit test for Object Factory
  Function List :
              initDbDataStrut
              main
  History       :
  1.Date        : 2012/6/26 23:37:53
    Author      : Shuang Wang
    Modification: Created file

*************************************************************************/
//#include "DataTypeFactories.h"    //has been included in Attribute.h
//#include "Attribute.h"
#include       "InformationSystem.h"
#include    <iostream>
#include    <vector>
#include    <iomanip>       //for setw
#include    <cstdlib>       // for srand
#include    <ctime>         // for time
using namespace std;
using namespace rough_set;

inline void    dispUsage() {
    cout<< "Usage:"<<endl<<"RS <DataFileName.txt>"<<endl;
}

int main( int argc , char * argv[]) {
    if ( argc<2 ) {
        dispUsage();
        return -1;
    }
        try {
        srand(static_cast<unsigned int>(time(NULL)));
               InformationSystem myIs;
               myIs.getTrainingData(argv[1]);
        myIs.doCalculate(cout);
        } catch ( runtime_error & runErr) {
               cout<<runErr.what()<<endl;
               exit(-1);
```

```
        }
}
/*************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *************************************************************************
  File Name     : Universe.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/6 22:04:40
  Last Modified :
  Description   : implementation of class Universe
  Function List :
  History       :
  1.Date        : 2012/7/6 22:04:40
    Author      : Shuang Wang
    Modification: Created file

*************************************************************************/

#include "Record.h"
#include "Universe.h"
using namespace std;

namespace rough_set {

ostream & operator << ( ostream & os, const Universe & rhs ) {
    for( Universe::const_iterator univIt = rhs.begin();
                            univIt!= rhs.end();   univIt++ ) {
        const Record & curRow = *(*univIt);
        os<<curRow<<endl;
    }
       os<<endl;
    return os;
}
Universe::~Universe() {
    for( Universe::iterator univIt = begin();
                 univIt!= end();   univIt++ ) {
        delete (*univIt);   // delete the obj of Record, this delete will trigger
                            // Record's destructor which will delete all the DataType<T> in this
Record
                            // and vector<>'s destructor which will delete vector<DataTypeBase *>
itself
                            // because Record derived from vector<DataTypeBase*>

    }
}

void    Universe::resetEachRecordState(){
    for( UnivIt univIt = begin(); univIt!= end();   univIt++ ) {
        (*univIt)->resetAlreadyInEqset();
    }
}

void    Universe::calcIndRelation( VecIndRelation & rIndRelations  ) {
    IndRelation *pCurInd = new IndRelation();
    rIndRelations.push_back( pCurInd);
    for( UnivIt univIt = begin(); univIt!= end();   univIt++ ) {
        Record * pCurRow = *univIt;
        if ( !pCurRow->alreadyInOtherEqSet()) {
            IndRecs * pNewEqSet = new IndRecs();
            pCurRow->setAlreadyInEqset();
            pNewEqSet->insert(pCurRow);   //this record is the  representative member of this Ind
            for ( UnivIt    univIt2 = univIt+1;
                            univIt2!= end();   univIt2++ ){
                if ( !(*univIt2)->alreadyInOtherEqSet()
                      && ( *pCurRow == **univIt2 )) {
                    pNewEqSet->insert(*univIt2);
                    (*univIt2)->setAlreadyInEqset();
                }
```

```
            }
            pCurInd->push_back(pNewEqSet);
        }
    }
        resetEachRecordState();
#ifdef  XDEBUG
    cout<<"get EC#="<< pCurInd->size()<<endl;
#endif
}

}


/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : Universe.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/7/6 21:50:41
  Last Modified :
  Description   : definition of class Universe which is consisted of Records
------------------
  Function List :
  History       :
  1.Date        : 2012/7/6 21:50:41
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#ifndef UNIVERSE_H_
#define UNIVERSE_H_

#include    "IndRelations.h"
#include    <vector>
#include    <iostream>
namespace rough_set {           // stuff used in rough sets peojects

class Record;                   // for Record * , do not need to include Record.h

//typedef std::vector<Record*> Universe;
class   Universe : public std::vector<Record *> {
friend std::ostream& operator << ( std::ostream & os, const Universe & rhs );
    void    resetEachRecordState();

public:
    ~Universe() ;
    void    calcIndRelation( VecIndRelation & rIndRelations  );

} ;

typedef Universe::iterator   UnivIt;

}

#endif /* UNIVERSE_H_ */

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : DecisionEntry.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/2 18:07:12
  Last Modified :
  Description   : implementation of DecisionEntry
  Function List :
```

```
  History       :
  1.Date        : 2012/9/2 18:07:12
    Author      : Shuang Wang
    Modification: Created file

*******************************************************************************/
#include "DecisionEntry.h"

using namespace std;

namespace rough_set {

DecisionEntry::DecisionEntry( RowIdx_t r1, RowIdx_t r2, BitIdx_t bitNum) :
    row1(r1), row2(r2),
    pBitmap( new Bitmap( bitNum ) ) {
}

DecisionEntry::~DecisionEntry() {
    delete pBitmap;
    pBitmap = 0;
}

ostream& operator<< ( ostream & os, const DecisionEntry& aEntry){
    os<<"( "<<aEntry.row1<<", "<<aEntry.row2<<")= "<<*(aEntry.pBitmap);
    return os;
}


}

/*******************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : DecisionEntry.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/2 17:47:55
  Last Modified :
  Description   : DecisionEntry: the element in decision table
  Function List :
  History       :
  1.Date        : 2012/9/2 17:47:55
    Author      : Shuang Wang
    Modification: Created file

*******************************************************************************/
#ifndef DECISION_ENTRY_H_
#define DECISION_ENTRY_H_

#include    "Bitmap.h"

namespace rough_set {
typedef     size_t  RowIdx_t;
class DecisionEntry {
        friend  class PopuEntry;
    friend  std::ostream& operator<< ( std::ostream & os, const DecisionEntry& aEntry);
    RowIdx_t     row1;
    RowIdx_t     row2;
    Bitmap  * pBitmap;  // we do not know the number of bits now,
                        // so we only can define the bitmap as a pointer
public :
    DecisionEntry( RowIdx_t r1,RowIdx_t r2, BitIdx_t bitNum );
    ~DecisionEntry();
    void setBit ( BitIdx_t bitIndex)    {
        pBitmap->setBit( bitIndex);
    }
    void    resetBit( BitIdx_t bitIndex ) {
        pBitmap->resetBit( bitIndex);
    }
```

```
    BitIdx_t    getBitNum() {
        pBitmap->getBitNum();
    }
    bool     isNullBmp() {
        return  (!pBitmap->getNumOf1());
    }
    // bmp1 < bmp2 means bmp1 included in bmp2
    bool    operator < ( const DecisionEntry & longerEntry) const {
        return ( *pBitmap< *(longerEntry.pBitmap ));
    }
};

}
#endif /* DECISION_ENTRY_H_ */




/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : DecisionTable.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/2 21:09:54
  Last Modified :
  Description   : DecisionTable.cpp
  Function List :
  History       :
  1.Date        : 2012/9/2 21:09:54
    Author      : Shuang Wang
    Modification: Created file

******************************************************************************/

#include       "DecisionTable.h"
#include       "Record.h"            // for comparision of attributes of Record-pair in function
calcDecisionTable()
#include       "Attribute.h"  // for DecisionTable::calcDecisionTable(
using namespace std;

namespace rough_set {
DecisionTable ::~DecisionTable() {
    for (    DecisionTable::iterator it= begin(); it!=end(); it++) {
        delete *it;
    }
}

void DecisionTable::calcDecisionTable(
        const SubAttrs &  subAttrs,
        const Attributes & rAttributes, //range match need attributes which holds ranges
        //AttrIdx_t      attributeNum,
        const Universe &  universe ,
        BitIdx_t decisionIdx         // index of decision Attribute
    ){
    AttrIdx_t subAttrNum = subAttrs.size();
    for ( Universe::const_iterator it1 = universe.begin(); it1!=universe.end(); it1++) {
        for( Universe::const_iterator it2 = it1+1; it2!=universe.end(); it2++ ) {
            if ( *((*it1)->at(decisionIdx) )!= *( (*it2)->at(decisionIdx))) {
                RowIdx_t    row1 = it1 - universe.begin();
                RowIdx_t    row2 = it2 - universe.begin();
                DecisionEntry * pNewEntry = new DecisionEntry( row1,row2, subAttrNum );
                push_back( pNewEntry );     // add pointer of this bitmap into this desicion
table
                for ( SubAttrs::const_iterator pAttrIdx = subAttrs.begin();
                        pAttrIdx != subAttrs.end(); pAttrIdx++ ) {
                    BitIdx_t  subAttrIdx = ( pAttrIdx- subAttrs.begin() );
                    /*
```

```
                    if (*((*it1)->at(*pAttrIdx) ) != *( (*it2)->at(*pAttrIdx))){
                        pNewEntry->setBit( subAttrIdx);
                    }*/
                    if ( !((*it1)->at(*pAttrIdx)->match( *( (*it2)->at(*pAttrIdx)),
rAttributes.at(*pAttrIdx)))) {
                        pNewEntry->setBit( subAttrIdx);
                    }
                }
            }
        }
    }
}

ostream& operator<< ( ostream & os, const DecisionTable& tbl){
    for (    DecisionTable::const_iterator it= tbl.begin(); it!=tbl.end(); it++) {
        os<<'\t'<<it-tbl.begin()<<"  "<<**it<<endl;
    }
       return os;
}

////////////////////////////////////////////////////////////////////////////
//
//            implementation of class  VecDecisionTable
//
////////////////////////////////////////////////////////////////////////////
VecDecisionTable::~VecDecisionTable() {
    for (    VecDecisionTable::iterator it= begin(); it!=end(); it++) {
        delete *it;
    }
}


}


/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : DecisionTable.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/2 19:57:18
  Last Modified :
  Description   : definition of Desicion Table for Genetic Algo
  Function List :
  History       :
  1.Date        : 2012/9/2 19:57:18
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/

#ifndef DECISION_TABLE_H_
#define DECISION_TABLE_H_

#include    "DecisionEntry.h"
#include    "Universe.h"                //for function calcDecisionTable
#include    "AttributeSubSet.h"

#include    <vector>

namespace  rough_set {

typedef     std::vector<AttrIdx_t>  SubAttrs;
typedef     std::vector<SubAttrs * >  VecSubAttrs;
class  Attributes;
class  DecisionTable : public std::vector<DecisionEntry*> {
friend  std::ostream& operator<< ( std::ostream & os, const DecisionTable& aEntry);
    //the * of DecisionEntry* avoid defining the copy constructor for Decision entry
```

```
public:
    ~DecisionTable();
    void    calcDecisionTable(const SubAttrs& atrr, /*AttrIdx_t      attributeNum,*/
        const Attributes & rAttributes, //range match need attributes which holds ranges
        const Universe& universe, BitIdx_t decisionIdx );
};

// each scenario has its own Decision Table, so we have a vector of DecisionTable *
// the * of DecisionTable* avoid vector copy
class   VecDecisionTable: public std::vector<DecisionTable *> {
public:
    ~VecDecisionTable();
};

}
#endif /* DECISION_TABLE_H_ */

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : GeneAlgo.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/2 21:41:08
  Last Modified :
  Description   : implementation of class GeneAlgo
  Function List :
  History       :
  1.Date        : 2012/9/2 21:41:08
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#include    "GeneAlgo.h"
#include    <cstdlib>       // for rand
#include       <algorithm>           // for random_shuffle and sort
using namespace std;
namespace rough_set {

GeneAlgo::GeneAlgo(): popuNum_(10), eliteRate_(20),
        crossOverRate_(70),
        mutationRate_(5),
        crossOverStartIdx_(popuNum_* eliteRate_ /100),
        loopMax_(2000),
        epslon_(1e-6),
        //topScoreKeepTime_(0),     //initialized in run4EachScenario()
        //loopNum_(0),
        topScoreStraightTime_(4) {
}

GeneAlgo::~GeneAlgo() {
    for ( VecSubAttrs::const_iterator it = vecSubAttrs_.begin();
                  it !=vecSubAttrs_.end() ;it++) {
        delete * it;
    }
}
void    GeneAlgo::copySubAttr( const VecAttrSubSets &    vecSubAttrSet ) {
    totAttrNum_ = AttributeSubSet::getTotAttrNum();
    for ( VecAttrSubSets::const_iterator it = vecSubAttrSet.begin();
            it !=vecSubAttrSet.end() ; it++) {
        AttributeSubSet * pCurAttrSubSet = *it;
        SubAttrs * pNewSubAttrs = new SubAttrs;
        vecSubAttrs_.push_back( pNewSubAttrs);
        for ( AttributeSubSet::const_iterator it2 = pCurAttrSubSet->begin();
            it2!= pCurAttrSubSet->end(); it2++) {
            pNewSubAttrs->push_back( *it2);
        }
        //sort( pNewSubAttrs->begin(),pNewSubAttrs->end());  set is already sorted
    }
```

```
}

void    GeneAlgo::calcDecisionTables(    const Universe & universe ,
            const Attributes & rAttributes,
            int decisionIdx ) {
    m_ = universe.size();
    for ( VecSubAttrs::const_iterator it = vecSubAttrs_.begin();
                    it !=vecSubAttrs_.end() ;it++) {
        DecisionTable * pNewDTbl = new DecisionTable;   // each scenario has its own Decision
Table
        vecDecisionTbl_.push_back( pNewDTbl);
        pNewDTbl->calcDecisionTable(**it, rAttributes, universe , decisionIdx );
    }
}

void    GeneAlgo::outDecisionTables( std::ostream & os ){
    for ( VecDecisionTable::const_iterator it = vecDecisionTbl_.begin();
                    it !=vecDecisionTbl_.end() ;it++) {
        cout<<"Decision Table for scenario "<< it- vecDecisionTbl_.begin() <<" is :"<<endl
<<**it<<endl;
    }
}

void    GeneAlgo::start( ostream & os, const Attributes & rAttribute , VecAndOrExpr & dfs) {
    for ( VecDecisionTable::const_iterator it = vecDecisionTbl_.begin();
                    it !=vecDecisionTbl_.end() ;it++) {
        size_t  scenarioIdx = it- vecDecisionTbl_.begin() ;
        AndOrExpr * pNewAOE = new AndOrExpr();
        dfs.push_back( pNewAOE);
        run4EachScenario( os,*(vecSubAttrs_.at(scenarioIdx)),*(vecDecisionTbl_.at(scenarioIdx)),
*pNewAOE);
        outPopuWithName( os, rAttribute,*(vecSubAttrs_.at(scenarioIdx)) );
        population_.topPopus( rAttribute, *(vecSubAttrs_.at(scenarioIdx)),*pNewAOE );
        population_.clearAll();        // prepare for next scenario
    }
}

void    GeneAlgo::run4EachScenario( ostream & os ,
                    const SubAttrs& subAttrs,
                    const DecisionTable& decisionTbl ,
                    AndOrExpr & df ) {
    topScoreKeepTime_ = loopNum_ = 0;       // initialize loop control variable for each scenario
    generate1stPopulation(decisionTbl);
    evaluation( m_, subAttrs, decisionTbl );
    os<<population_<<endl;
    while( ! reachTermination()) {
        selectNReproduction();
        crossOver(subAttrs);
        mutation(subAttrs);
        evaluation(m_, subAttrs, decisionTbl);
    }
    os<<population_<<endl;
}


Index_t GeneAlgo::getUniqueIdx( const     DecisionTable& decisionTbl ) {
    Index_t tentativeIdx;
    int loop = 0;
    int maxLoop = popuIdxMax_*5;
    do {
        tentativeIdx = rand()% popuIdxMax_;
        //cout<<tentativeIdx<<" ";
        loop++;
    }   while ( (pGeneratedIdx_->getBit(tentativeIdx)||         //already generated this idx
                decisionTbl.at(tentativeIdx)->isNullBmp())&&     // a NULL bitmap
                loop< maxLoop );
    if ( loop >= maxLoop ) {
        cout<<endl<<"Loop time= "<< loop<<", maxLoop= "<< maxLoop<<endl;
        throw   runtime_error("Not enough unique index for entries in 1st generation");
    }
    pGeneratedIdx_->setBit(tentativeIdx);
```

```
    return tentativeIdx;
}

void    GeneAlgo::generate1stPopulation( const     DecisionTable& decisionTbl ){
    popuIdxMax_ = decisionTbl.size();
    int curPopuNum = 0;
    pGeneratedIdx_ = new Bitmap( popuIdxMax_);
    while( curPopuNum < popuNum_ ) {
        Index_t popuIdx = getUniqueIdx(decisionTbl );
        population_.push_back(new PopuEntry(popuIdx, *( decisionTbl.at(popuIdx))));
        curPopuNum++;
    }
    delete pGeneratedIdx_;
}

void    GeneAlgo::evaluation( size_t m,const SubAttrs& subAttrs, const DecisionTable& decisionTbl
){
    size_t  N = subAttrs.size();
    //size_t  K = m*(m-1)/2;
    for ( Population::iterator it = population_.begin(); it!= population_.end(); it++) {
        (*it)->calcFitnessValue( N,m,subAttrs, decisionTbl );
    }
}

bool comp2Popu( PopuEntry * pP1, PopuEntry * pP2) {
    return *pP1< *pP2;
}

bool    GeneAlgo::reachTermination(){
    loopNum_++;
    sort(population_.begin(), population_.end(),comp2Popu);
    cout<<"After sorted...."<<endl<<population_<<endl;
    double newTopScore = (*population_.begin())->getFitnessValue();
    double scoreDiff = newTopScore - topScore_;
    if ( abs(scoreDiff < epslon_ ) ) {
        ++topScoreKeepTime_;
    } else {
        topScoreKeepTime_ = 0;
    }
    cout<<"Top score keep "<<topScoreKeepTime_<<endl;
    //if ( scoreDiff > 0.0 ) {
        topScore_ = newTopScore;
    //}
    if( (loopNum_>= loopMax_ ) ||( topScoreKeepTime_>=topScoreStraightTime_)) {
       return true;
    } else {
        return false;
    }
}

void    GeneAlgo::selectNReproduction(){
    // random shuffle preparing for select crossover pairs randomly
    random_shuffle( population_.begin()+crossOverStartIdx_ , population_.end());
}

void    GeneAlgo::crossOver( const SubAttrs& subAttrs ){
    int crossOverNum = popuNum_ * crossOverRate_ /100;
    int crossOverEnd = crossOverStartIdx_+crossOverNum;
    if ( crossOverNum %2 ) {
        ++crossOverNum ;
    }
    for ( int i=crossOverStartIdx_; i<crossOverEnd; i+=2 ) {
        BitIdx_t    crossOverIdx = rand()%subAttrs.size();
        population_.at(i)->crossOver( * population_.at(i+1) , crossOverIdx );
    }
}

void    GeneAlgo::mutation( const SubAttrs& subAttrs ){
    double  mutationNum = static_cast<double>( popuNum_ * mutationRate_ )/100.0;
    int     mutationLoop = 0;
    if ( mutationNum < 1.0 ) {
```

```
        double curDo = static_cast<double> (rand()%100)/100.0;
        if ( curDo < mutationNum ) {
            mutationLoop =1;
        }
    } else {
        mutationLoop = static_cast<int>( mutationNum );
    }

    for ( int i = 0; i< mutationLoop; i++ ) {
        int mutationIdxRange = popuNum_ - crossOverStartIdx_ ;
        int mutationIdx = crossOverStartIdx_+ rand()% mutationIdxRange;
        BitIdx_t mutationBitPos = rand()% subAttrs.size();
        population_[mutationIdx]->reverse( mutationBitPos );
    }
}

void    GeneAlgo::outPopuWithName( ostream & os,const Attributes & rAttribute ,const SubAttrs&
subAttrs){
    // output the decision function with attribute name of the top score population
    population_.at(0)->trans2name(os, rAttribute,subAttrs);
}

}
/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : GeneAlgo.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/2 20:23:59
  Last Modified :
  Description   : GeneAlgo : genetic algorithm to calculate decision function
  Function List :
  History       :
  1.Date        : 2012/9/2 20:23:59
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#ifndef GENE_ALGO_H_
#define GENE_ALGO_H_

#include    "DecisionTable.h"
#include    "Universe.h"                //for function calcDecisionTables
#include    "AttributeSubSet.h"
#include    "Population.h"
#include    "AndOrExpr.h"               // for start( .. )
#include    <vector>
namespace rough_set {
class GeneAlgo {
    size_t              m_;             // # of enties in Universe
    int                 popuNum_;
    int                 eliteRate_;     // top eliteRate_% will not change to next generation
    int                 crossOverRate_; // %
    int                 mutationRate_;  // %
    int                 crossOverStartIdx_;
    int                 loopMax_;               // algo will end after loopMax_ loops
    int                 topScoreStraightTime_;  // top score keep same for this times will
terminate the loop
    int                 loopNum_;
    double              topScore_;
    int                 topScoreKeepTime_;
    const double        epslon_;
    AttrIdx_t           totAttrNum_;
    VecDecisionTable    vecDecisionTbl_;
    VecSubAttrs         vecSubAttrs_;
    Population          population_;
    Index_t             popuIdxMax_;    // random max for generate 1st population randomly
```

```
    Bitmap * pGeneratedIdx_;            // used for generate unique idx which will set
corresponding bit in
                                        //  this Bitmap to 1;
public:
        GeneAlgo();
    ~GeneAlgo();
    void    calcDecisionTables(    const Universe & universe ,const Attributes & rAttributes, int
decisionIdx );
    void    outDecisionTables( std::ostream & os );
    void    copySubAttr( const VecAttrSubSets & scenario );
    void    setPopuNum( int iPopuNum ) {
        popuNum_ = iPopuNum ;
        std::cout<<"The amount of Population is "<< popuNum_<<std::endl;
    }
    void    start( std::ostream & os,const Attributes & rAttribute , VecAndOrExpr & dfs  );
    void    outPopuWithName( std::ostream & os,const Attributes & rAttribute ,  const SubAttrs&
subAttrs);
private :
    void    run4EachScenario( std::ostream & os,const SubAttrs& subAttrs, const DecisionTable&
decisionTbl , AndOrExpr & df );
    Index_t getUniqueIdx( const     DecisionTable& decisionTbl );
    void    generate1stPopulation( const DecisionTable& decisionTbl );
    void    evaluation( size_t m, const SubAttrs& subAttrs, const DecisionTable& decisionTbl );
    bool    reachTermination();
    void    selectNReproduction();
    void    crossOver( const SubAttrs& subAttrs );
    void    mutation( const SubAttrs& subAttrs );

};

}

#endif /* GENE_ALGO_H_ */

/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : Population.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/9 19:37:53
  Last Modified :
  Description   : implementation of class Population & PopuEntry
  Function List :
  History       :
  1.Date        : 2012/9/9 19:37:53
    Author      : Shuang Wang
    Modification: Created file

******************************************************************************/
#include    "Population.h"
//#include     "DecisionTable.h"   //already in "Population.h"
#include    <iomanip>
#include       <string>
#include    <cassert>
using namespace std;
namespace rough_set {
PopuEntry::PopuEntry( Index_t idx, const DecisionEntry&    src ) :
                idxInDecistionTbl_ ( idx),
                fitnessValue_(0.0),
                pBitmap( new Bitmap( *src.pBitmap)){
}

PopuEntry::~PopuEntry() {
    //cout<<"POpuEntry was called"<<endl;
    delete pBitmap;
}

size_t  PopuEntry::calcCr(   const SubAttrs& subAttrs,
```

```
                const DecisionTable& decisionTbl  ) {

    if ( ! pBitmap->getNumOf1() ) {
        return 0;
    }
    // generate mask for each bit and add it into a vector
    // it is coincident that the data type of this vector is also DecisionTable
    DecisionTable  tmpMasks;
    for ( SubAttrs::const_iterator it = subAttrs.begin(); it!= subAttrs.end(); it++){
        BitIdx_t curIdx = it- subAttrs.begin();
        if (pBitmap->getBit( curIdx ) ) {         // getBit(*it)
            DecisionEntry * pMask = new DecisionEntry( 0, 0, pBitmap->getBitNum());
            pMask->setBit( curIdx );
            tmpMasks.push_back( pMask);
        }
    }

    if ( tmpMasks.size() !=  pBitmap->getNumOf1()) {
        throw runtime_error(string("Different number of 1 @ ")+__FILE__+" "+__FUNCTION__);
    }

    // scan decision table
    size_t Cr = 0;
    for ( DecisionTable::const_iterator itDesicionEntry = decisionTbl.begin();
                itDesicionEntry != decisionTbl.end();    itDesicionEntry++ ) {
        for ( DecisionTable::iterator itMask= tmpMasks.begin();
                    itMask!= tmpMasks.end();   itMask++) {
            if ( **itMask < **itDesicionEntry ) {
                ++Cr;
                break;  // force to next decision entry once any mask is covered by this decision
entry
            }
        }
    }
    return Cr;
}

void PopuEntry::calcFitnessValue(   size_t N,size_t m,
                        const SubAttrs& subAttrs,
                        const DecisionTable& decisionTbl ){
    size_t     Cr = calcCr( subAttrs, decisionTbl);
       double  bonus= static_cast<double>(Cr)/m/2.0;
    size_t  K = m*(m-1)/2;
    // F(r)= (N-Lr)/N + Cr/K
    if( pBitmap->getNumOf1()> N ) {
        throw runtime_error(string("wrong number of 1 @ ")+__FILE__+" "+__FUNCTION__);
    }

    fitnessValue_ = static_cast<double>(N- pBitmap->getNumOf1())/ N +
                    static_cast<double>(Cr)/K + bonus;

}

void    PopuEntry::trans2name(std::ostream & os,const Attributes & rAttribute ,    const
SubAttrs& subAttrs){
    os  <<" # "<<setw(3)<<idxInDecistionTbl_<<" : "<< "f(r)= "<<fitnessValue_
        <<"   < ";
    size_t loopTime = pBitmap->getBitNum();
    size_t  commaPos =pBitmap->getNumOf1();
    for ( size_t i=0; i<loopTime ; i++ ) {
        if ( pBitmap->getBit(i) ) {
            int attrIdx = subAttrs.at(i);
            os<< rAttribute.at(attrIdx).getName();
            if ( --commaPos ) {
                os<<" , ";
            }
        }
    }
    os<<" >"<<endl;
}
```

```
void    PopuEntry::conv2AttrBmp(     const SubAttrs& subAttrs, AndExpr & rAE){
    size_t loopTime = pBitmap->getBitNum();
    for ( size_t i=0; i<loopTime ; i++ ) {
        if ( pBitmap->getBit(i) ) {
            int attrIdx = subAttrs.at(i);
            assert(attrIdx<rAE.getBitNum());
            rAE.setBit(attrIdx );
        }
    }

}

ostream& operator<< ( ostream & os, const PopuEntry& aEntry){
    os  <<" # "<<setw(3)<<aEntry.idxInDecistionTbl_<<" : "<< "f(r)= "<<aEntry.fitnessValue_
        <<"   < "<<*(aEntry.pBitmap)<<" >"<<endl;
    return os;
}

//////////////////////////////////////////////////////////////////////
// implementation of class Population
//////////////////////////////////////////////////////////////////////
Population::~Population() {
    delAllPopuEntry();
}

void    Population::topPopus(const Attributes & rAttribute, const SubAttrs& subAttrs, AndOrExpr &
df) {
    AndExpr tmp1 = AndExpr(rAttribute.size());
    this->at(0)->conv2AttrBmp( subAttrs, tmp1);
    AndExpr tmp2 = AndExpr(rAttribute.size());
    this->at(1)->conv2AttrBmp( subAttrs, tmp2);
    df.insert(tmp1);
    df.insert(tmp2);
}

void Population::delAllPopuEntry() {
    for ( Population::iterator it = begin(); it!= end(); it++) {
        delete (*it);
    }
}

void Population::clearAll(){
    delAllPopuEntry();      // release the memory of pointers
    clear();                // clear the vector
}

ostream& operator<< ( ostream & os, const Population& tbl){
    cout<<"Population is:"<<endl;
    for (    Population::const_iterator it= tbl.begin(); it!=tbl.end(); it++) {
        os<<'\t'<<it-tbl.begin()<<"  "<<**it<<endl;
    }
       return os;
}

}

/*************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *************************************************************************
  File Name     : Population.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/9/9 14:25:05
  Last Modified :
  Description   : definition of class PopuEntry (population entry) and
                  class Population for genetic algorithm
  Function List :
  History       :
  1.Date        : 2012/9/9 14:25:05
```

```
    Author      : Shuang Wang
    Modification: Created file

*********************************************************************************/
#ifndef POPULATION_H_
#define POPULATION_H_

//#include    "DecisionEntry.h"           //already include Bitmap.h
#include    "DecisionTable.h"           //for SubAttrs
#include    "Attribute.h"               //for trans2name()
#include    "AndOrExpr.h"
#include    <vector>

namespace rough_set {
typedef size_t  Index_t;
typedef double  Fitness_t;
class PopuEntry {
friend std::ostream& operator<< ( std::ostream & os, const PopuEntry& aEntry);
    Index_t     idxInDecistionTbl_;
    Fitness_t   fitnessValue_;
    Bitmap      *pBitmap;
public:
    PopuEntry( Index_t idx, const DecisionEntry & src );
    void calcFitnessValue(size_t N,size_t K,  const SubAttrs& subAttrs,const DecisionTable&
decisionTbl );      //will set fitnessValue_
    ~PopuEntry();
    bool operator < ( const PopuEntry & rhs) {
        return fitnessValue_ > rhs.fitnessValue_;       // !!! for descending order
    }
    double  getFitnessValue() { return fitnessValue_; }
    void    reverse( BitIdx_t mutationBitPos ) {
        pBitmap->reverseBit( mutationBitPos );
    }
    void    crossOver( PopuEntry& op2, BitIdx_t crossPoint) {
        pBitmap->crossOver( *(op2.pBitmap), crossPoint );
    }
    const   Bitmap& getBitmap()  { return * pBitmap ; }
    void    trans2name(std::ostream & os,const Attributes & rAttribute ,    const SubAttrs&
subAttrs);
    void    conv2AttrBmp(     const SubAttrs& subAttrs, AndExpr & rAE );
private:
    size_t  calcCr(  const SubAttrs& subAttrs,const DecisionTable& decisionTbl  );
};

class Population:public std::vector<PopuEntry *> {  // PopuEnty * avoid copy
    friend std::ostream& operator<< ( std::ostream & os, const Population& aTbl);
    void    delAllPopuEntry();
public:
    ~Population();
    void clearAll();
    void    topPopus(const Attributes & rAttribute,const SubAttrs& subAttrs,  AndOrExpr & df);

};

}
#endif /* POPULATION_H_ */
```

```
/**********************************Convert Part*****************************************/


#include    "DtFileParser.h"
#include    <iostream>
#include    <vector>
#include    <string>       // for srand
#include    <cstdlib>
#include    <exception>

using namespace std;

inline void    dispUsage() {
    cout<< "Usage:"<<endl<<"convert <InputTrainingFileName> <InputTestingFileName>
<outputFileName> <c/n>"<<endl;
    cout<< "\tc:convert float to Integer"<<endl;
    cout<< "\tn:do not convert float to Integer,copy directly"<<endl;
}

int main( int argc , char * argv[]) {
    if ( argc<5 ) {
        dispUsage();
        return -1;
    }
    bool convert = false;
    if ( argv[4][0]=='c') {
        convert = true;
    }
       try {
        DtFileParser dtParser(argv[1],argv[2],argv[3]);
        dtParser.run( convert );
        return 0;
       } catch ( runtime_error & runErr) {
              cout<<runErr.what()<<endl;
              exit(-1);
       }
}


/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 ******************************************************************************
  File Name     : DtFileParser.cpp
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/10/13 23:38:52
  Last Modified :
  Description   : implementation of DtFileParser
  Function List :
  History       :
  1.Date        : 2012/10/13 23:38:52
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/

#include    "DtFileParser.h"
#include    <string>
#include    <sstream>
#include    <iostream>
#include    <iomanip>       //for setw
#include    <exception>

using namespace std;
DtFileParser::DtFileParser(     const char * inTrainFileName,
```

```
                                const char * inTstFileName,
                                const char * outFileName ) :
        /*curDataLine_(0),*/ curState_(PS_FILE_START) ,
        dataStart_(0),maxDigitN_(0),padWidth_(0),lineNum_(0) {
    inTrainFile_.open( inTrainFileName ,ios_base::in | ios_base::binary  );
    inTstFile_.open( inTstFileName ,ios_base::in | ios_base::binary  );
    outDtFile_.open( outFileName);
    if ( inTrainFile_.fail() ) {
        throw runtime_error(string("File open failed : ")+string( inTrainFileName));
    }
    if ( inTstFile_.fail() ) {
        throw runtime_error(string("File open failed : ")+string( inTstFileName));
    }
    if ( outDtFile_.fail() ) {
        throw runtime_error(string("File open failed : ")+string( outFileName));
    }
}

DtFileParser::~DtFileParser(){
    inTrainFile_.close();
    inTstFile_.close();
    outDtFile_.close();
}

void    DtFileParser::read1Attr( istringstream& lineStr){
    string token;
    lineStr>>token;
    if(token!="@attribute"){
        curState_ = PS_ATTR_END;
        return;
    }
    string  attrName;
    lineStr>>attrName;
    AttrInfo    tmp;
    tmp.name_ = attrName;
    string  attrType;
    lineStr>>attrType;
    if ( attrType == "real" ) {
        tmp.type_ = DT_FLOAT;
    } else if ( ( attrType =="integer" ) || ( attrType =="int" )){
        tmp.type_ = DT_INT;
    } else {
        // find {MIT,... or/{0.. }
        bool    foundAlpha = false;
        bool    foundDigit = false;
        bool    foundMinus = false;
        for ( int i=0; i<attrType.size(); i++ ) {
            if ( isalpha(attrType[i])){
                foundAlpha = true;
            }else if ( isdigit(attrType[i])){
                foundDigit = true;
            }else if ( '-'== attrType[i]) {
                foundMinus = true;
            }
        }
        if ( foundAlpha )    tmp.type_ = DT_STRING;
        else if ( foundMinus )    tmp.type_ = DT_STRING;
        else if ( foundDigit )    tmp.type_ = DT_INT;
        else { tmp.newType_ = tmp.type_ = DT_UNKOWN; }
        if ( tmp.type_ == DT_UNKOWN ) {
            throw runtime_error(string("wrong type for attr:")+ tmp.name_ );
        }
    }
    // curState is still  PS_ATTR_START
    if ( tmp.type_ != DT_UNKOWN ) {
        tmp.newType_ = tmp.type_;
        attributes_.push_back( tmp );
    }
}
```

```
void    DtFileParser::findDataPos(ifstream &  inFile) {
    bool    foundData = false;
    string line;
    while (getline(inFile, line)) {
        istringstream   lineStr(line);
        string token;
        lineStr>>token;
        if(token=="@data"){
            foundData = true;
            dataStart_ = inFile.tellg();
            break;
        }
    }
    if( foundData ) {
        cout<<"data from "<<dataStart_<<endl;
        curState_ = PS_FIRST_DATA_LINE;
    } else {
        throw runtime_error(string("can not find @data segment"));
    }
}

void    DtFileParser::findRecNumAndMaxDigit( ifstream &  inFile, bool convert) {
    string line;
    int curDigitNum = 0;
    int cur0TailNum = 0;
    lineNum_ = 0;
    while (getline(inFile, line)) {
        if ( !line.size()) continue;
        ++lineNum_;
        if ( !convert ) continue;
        for( int i=0; i<line.size(); i++ ) {
            switch ( curState_ ) {
                case    PS_FIRST_DATA_LINE :
                case    PS_END_DIGIT :
                    if ( line[i]=='.' ) {
                        curDigitNum = 0;
                        cur0TailNum = 0;
                        curState_ = PS_START_DIGIT;
                    }
                    break;
                case    PS_START_DIGIT:
                    if( isdigit(line[i])) {
                        ++curDigitNum;
                        if ( line[i]=='0' ) {
                            ++ cur0TailNum;
                        }
                    } else {
                        curState_ = PS_END_DIGIT;
                        if ( cur0TailNum == curDigitNum ) {
                            curDigitNum = 0;        // if .00 then ignore it.
                        }
                        if ( curDigitNum> maxDigitN_ ){
                            maxDigitN_ = curDigitNum ;
                        }
                    }
                    break;
                default:
                    cout<< "curState = "<<curState_<<endl;
                    throw runtime_error(string("Wrong state in")+__FUNCTION__);
                    break;
            }
        }
    }
    cout<<"Max digit # = "<< maxDigitN_<<endl;
    cout<<"Data lines = "<< lineNum_;
    int lineNum = lineNum_;
    padWidth_ = 0;
    while ( lineNum !=0 ) {
        ++padWidth_;
        lineNum/=10;
    }
```

```
    cout<<",padding width= "<<padWidth_<<endl;
}

void    DtFileParser::scanData( std::ifstream &  inFile, bool convert ) {
    findDataPos( inFile );
    findRecNumAndMaxDigit( inFile , convert );
}

void    DtFileParser::readHeader( ifstream &  inFile ) {
    string line;
    // read line at time until end-of-file
    while (getline(inFile, line)) {
        switch ( curState_ ) {
            case PS_FILE_START : {
                istringstream   lineStr(line);
                string token;
                lineStr>>token;
                if(token!="@relation"){
                    throw runtime_error(string("wrong data file format near\'")+token+"\'\n");
                } else {
                    curState_ = PS_ATTR_START;
                }
                break;
            }
            case PS_ATTR_START: {
                for(int i=0;i<line.size();i++) {
                    if(line[i]=='[') line[i]=' ';
                }
                istringstream   lineStr(line);
                read1Attr(lineStr);
                break;
            }
            case PS_ATTR_END: {
                return;
            }
            default:
                throw runtime_error(string("Wrong state in")+__FUNCTION__);
                break;
        }
    }
}

bool    DtFileParser::AttrInfo::outNewType_ = false;
void    DtFileParser::showHeader( ostream & os, bool newType ) {
    AttrInfo::setOutFlag( newType);
    os<<"struct { string recName ;"<<endl;
    for( Attributes::iterator it= attributes_.begin(); it!=attributes_.end(); it++) {
        os<< '\t'<<*it<<" ;"<<endl;
    }
    os<<" }"<<endl;

}

void    DtFileParser::convertTypeFloat2Int() {
    for( Attributes::iterator it= attributes_.begin(); it!=attributes_.end(); it++) {
        if (it->type_ == DT_FLOAT ) {
            it->newType_ = DT_INT;
        } else {
            it->newType_ = it->type_;
        }
    }
}

void    DtFileParser::readNWriteData( ifstream &  inFile,ostream & os     ) {
    int multipler = 1;
    //ostream&    os = cout;
    for ( int i=0; i <maxDigitN_ ; i++ ) {
        multipler*= 10;
    }
    cout<<"each data * "<<multipler<<endl;
    cout<<"file @"<<inFile.tellg()<<endl;
```

```
    inFile.clear();              // clear fail flag
    inFile.seekg( dataStart_ );
    cout<<"after seek, file @"<<inFile.tellg()<<endl;
    string line;
    int lineNum=0;
    os<<"recordNum = "<<lineNum_<<endl<<endl;
    while (getline(inFile, line)) {
        if (!line.size())
            continue;
        ++lineNum;
        os<<'P'<<setfill('0')<<setw(padWidth_)<<lineNum<<"  ";
        for( int i=0; i<line.size(); i++ ) {
            if ( ','==line[i] ) {
                line[i] = ' ';
            }
        }
        istringstream   lineStr(line);
        cout<<line<<endl;
        //cout<<'.';
        for( Attributes::iterator it= attributes_.begin(); it!=attributes_.end(); it++) {
            switch (it->type_) {
                case DT_FLOAT :{
                    double  tmpData;
                    lineStr>>tmpData;
                    tmpData *= multipler;
                    os<<tmpData<<" ";
                    break;
                }
                case DT_INT:{
                    int tmpData;
                    lineStr>>tmpData;
                    os<<tmpData<<" ";
                    break;
                }
                case DT_STRING:{
                    string  tmpData;
                    lineStr>>tmpData;
                    os<<tmpData<<" ";
                    break;
                }
                default:{
                    cout<< "type_ = "<<it->type_<<endl;
                    throw runtime_error(string("Wrong state in")+__FUNCTION__);
                    break;
                }
            }
        }
        os<<endl;
    }
}

void    DtFileParser::OtherSegments(){
    outDtFile_ <<"subAttributes {     "<<endl<<"\t{  ";
    for( Attributes::iterator it= attributes_.begin(); it!=attributes_.end(); it++) {
        outDtFile_<<it->name_<<" , ";
    }
    outDtFile_<<" } }"<<endl<<endl;
    outDtFile_ <<"DecisionAttr = "<<endl;
    outDtFile_<<"MaxDecisionFuncORItems = 2000000"<<endl;
    outDtFile_<<"DoVerification = yes"<<endl;
    outDtFile_<<"PopuNum = "<<endl<<endl;
}

void    DtFileParser::processFile( std::ifstream &  inFile, OpMode mode, bool convert ){
    if ( TRAINING == mode ) {
        readHeader( inFile );
    }
    showHeader(cout);
    scanData( inFile, convert );
    //if( maxDigitN_>0) {
    if ( convert )  {
```

```
        convertTypeFloat2Int();     //always set newType_
    }
    //}
    if ( TRAINING == mode ) {
        showHeader(cout,true);
        showHeader(outDtFile_,true);       //output new type
        OtherSegments();
    } else {
        outDtFile_<<endl<<"DoClassify = yes"<<endl;
    }
    readNWriteData( inFile,outDtFile_ );
}

void    DtFileParser::run( bool convert ){
    processFile( inTrainFile_,TRAINING,convert);
    curState_ = PS_FILE_START;
    processFile( inTstFile_,TEST, convert);
}


/*****************************************************************************

                  Copyright (C), 2011-2021, Shuang Wang

 *****************************************************************************
  File Name     : DtFileParser.h
  Version       : Initial Draft
  Author        : Shuang Wang
  Created       : 2012/10/13 23:36:22
  Last Modified :
  Description   : definition of DtFileParser (data file parser )
  Function List :
  History       :
  1.Date        : 2012/10/13 23:36:22
    Author      : Shuang Wang
    Modification: Created file

*****************************************************************************/
#ifndef DATA_FILE_PARSER_H_
#define DATA_FILE_PARSER_H_

#include    <fstream>
#include    <string>
#include    <vector>
#include    <iostream>
enum    ParserState {
    PS_FILE_START,
    PS_ATTR_START,
    PS_ATTR_END,
    PS_FIRST_DATA_LINE,
    PS_START_DIGIT,
    PS_END_DIGIT,
    PS_NUM
};

enum    DataType {
    DT_INT,
    DT_FLOAT,
    DT_STRING,
    DT_UNKOWN,
    DT_NUM
};

enum    OpMode {
    TRAINING,
    TEST
};

class   DtFileParser {
    std::ifstream   inTrainFile_;    // input training data file
    std::ifstream   inTstFile_;      // input test data file
```

```
    std::ofstream   outDtFile_;      // output data file
    ParserState     curState_;
    //int     curDataLine_;
    int     maxDigitN_;
    int     padWidth_;
    int     lineNum_;
    struct  AttrInfo {
        static   bool   outNewType_;
        std::string     name_;
        DataType        type_;
        DataType        newType_;
        AttrInfo():type_(DT_UNKOWN) {}
        static void setOutFlag( bool isNewType_ ) { outNewType_ = isNewType_; }
    };
    friend std::ostream & operator<< ( std::ostream & os, const DtFileParser::AttrInfo& rhs);
    typedef std::vector<AttrInfo>   Attributes;
    Attributes      attributes_;
    std::streamoff  dataStart_;
public:
    DtFileParser( const char * inTrainName, const char *inTstName, const char * outFileName);
    ~DtFileParser();
    void    readHeader( std::ifstream &  inFile );
    void    showHeader( std::ostream & os, bool newType = false );
    void    scanData(std::ifstream &  inFile, bool convert);
    void    read1Attr(std::istringstream& lineStr);
    void    findDataPos( std::ifstream &  inFile );
    void    findRecNumAndMaxDigit( std::ifstream &  inFile , bool convert);
    void    convertTypeFloat2Int();
    void    readNWriteData(std::ifstream &  inFile, std::ostream & os );
    void    OtherSegments();
    void    processFile( std::ifstream &  inFile_,OpMode mode, bool convert );
    void    run( bool convert =false);
};

inline std::ostream & operator<< ( std::ostream & os, const DtFileParser::AttrInfo& rhs) {
    std::string typeName[]={"int","double","string"};
    os<<((rhs.outNewType_)?typeName[rhs.newType_]:typeName[rhs.type_])
        <<"  "<<rhs.name_;
    return os;
}

#endif /* DATA_FILE_PARSER_H_ */
```