ELSEVIER

Artificial Intelligence 105 (1998) 77–103

Artificial Intelligence

Rough computational methods for information systems

J.W. Guan *, D.A. Bell [1]

School of Information and Software Engineering, University of Ulster at Jordanstown, Jordanstown, BT 37 OQB Northern Ireland, UK

Received 16 December 1996; received in revised form 23 December 1997

Abstract

Rough set theory is a relatively new mathematical tool for use in computer applications in circumstances which are characterized by vagueness and uncertainty. The technique called rough analysis can be applied very fruitfully in artificial intelligence and cognitive sciences.

Although this methodology has been shown to be successful in dealing with the vagueness of many real-life applications, there are still several theoretical problems to be solved, and we also need to consider practical issues if we want to apply the theory.

It is the latter set of issues we address here, in the context of handling and analysing large data sets during the knowledge representation process. Some of the associated problems (for example, the general problem of finding all "keys") have been shown to be NP-hard. Thus, it is important to seek efficient computational methods for the theory.

In rough set theory, a table called an *information system* or a *database relation* is used as a special kind of formal language to represent knowledge syntactically. Semantically, knowledge is defined as classifications of information systems. The use of rough analysis does not involve the details of rough set theory directly, but it uses the same basic classification techniques.

We discuss computational methods for the rough analysis of databases.

Keywords: Intelligent information systems; Database and knowledge base systems

Introduction

In this paper we discuss how to apply the *rough analysis* technique to databases. Rough analysis is one of the main application techniques arising from rough set theory. It provides

* Corresponding author. Email: J.guan@ulst.ac.uk.

[1] Email: DA.bell@ulst.ac.uk.

0004-3702/98/$ – see front matter
PII: S0004-3702(98)00090-3

a technique for gaining insights into properties of data, dependencies and the significance of individual attributes in databases. It has important applications to artificial intelligence and cognitive sciences, as a tool for dealing with vagueness and uncertainty of facts, and in classification. The objective of this paper is to enhance the application of the technique by designing a series of algorithms to implement the technique in a knowledge representation context. To study the theory itself, the reader is referred to [6].

A sister paper [2] complements the collection of algorithms presented here; it focuses on the use of rough set theory, *per se*, as a means of discovering knowledge which is latent in database relations (i.e., data mining or knowledge discovery in databases) in the form of rules, with particular attention being paid to decision tables.

In this paper, we concentrate upon using the *rough analysis* technique (as opposed to rough set theory itself) for the classical database problems of *checking dependencies* and *finding keys* for a conventional relation (i.e., with no distinguished decision variables) with a view to using the solutions in general knowledge discovery [1]. One or two algorithms appear in both papers for convenience—being relevant to both studies.

Our practical motivation is to use the tool for computer applications in which reasoning and learning are based on (often large) collections of sense or other data stored in computers, and perhaps managed by database management systems. For example, we have shown elsewhere [1] how evidence can be obtained for reasoning purposes by examining the properties of data as expressed in certain kinds of integrity constraints (see below). Consider an investigation [3,14] into our distant ancestors, using a set of field notes recorded on paleontological sites (e.g., near Lake Turkana in Kenya). Ultimately, we are interested in deriving evidence from this data to support various hypotheses, for example, on the classes of skulls described in the dataset. One hypothesis, based on prima facie indications, is that there are 3 *different classes of skull*—a small, finely featured one, a large, coarsely featured one, and a third class lying between these two. An alternative hypothesis (there may be several) is that there are *really two classes*—the small and medium "classes" being the clearly distinguished female and male specimens of just one class, and the large "class" being a different species. The field data can be recorded as an information system (an illustration is given in Example 3.1 below), and rough analysis can be considered as a possible approach for deriving evidence from this data to see which hypothesis is best supported by the data. There is a basic need for classification in this example, and there is also a potential practical benefit if we can decide which features, or attributes, are significant in the classification exercise (the "core"). A related problem is to find which set of attributes or features (perhaps minimal) can be used to distinguish individual skulls from one another. This is what we mean by a "key". The solution to this "classical" problem in database theory can provide important support in underpinning the reasoning and learning applications encountered in artificial intelligence—not least in classification itself. The classification of (e.g., paleontological) specimens and events, and the accumulation of evidence to support hypotheses from observation and experimental data, are clearly very important in machine learning and other reasoning exercises. The discovery of keys can also provide insights into the structure of data which are not easy to get by alternative means.

In Section 1, we introduce information systems. An information system is a generalization of a database relation. Formally, it consists of two finite sets: one universe U and

one attribute set A. The rough analysis technique is applied to find the core—the set of significant attributes, and the keys—the minimal identifying sets of attributes.

The basic tool for rough set theory is classification. In Section 2, we discuss classification for one attribute. Algorithm E with the time complexity $O(|U|^2)$ to find the corresponding classification for an attribute is given, where $|U|$ is the cardinality of U. The algorithm can be run in parallel mode to compute concurrently all corresponding classifications for many attributes. In Section 3, we discuss classification intersections between many attributes. Algorithm I with the time complexity $O(|U|^2)$ to find the corresponding classification intersection between many attributes is given.

Using classification, we analyse dependencies between two subsets of attributes. Section 4 discusses functional dependencies (FD). We present Algorithm O with the time complexity $O(|U|^2)$ to check the functional dependency of two attribute subsets. Section 5 discusses identity dependencies (ID). We present Algorithm S with the time complexity $O(|U|^2)$ to check the identity dependency of two attribute subsets. The keys of an attribute set A are the minimal ID subsets of set A. Algorithm K with the time complexity $O(2^{|A|}|A||U|^2)$ to find all keys is presented in Section 6. In order to reduce the exponential complexity, we need to investigate the ID significance of an attribute and ID significant attribute subsets.

An attribute x in X ($\subseteq A$) is significant if X and $X - \{x\}$ are not ID. An attribute set X is significant if every attribute $x \in X$ is significant. Using classification, we analyse significances. Section 7 introduces the significance measure of attribute x in X. The time complexity for computing a significance is $O(|X| \times |U|^2)$. Algorithm C with the time complexity $O(|X|^2|U|^2)$ is presented to allow us to find the core—the set of significant attributes. Section 8 discusses significant subsets of attributes. Section 9 shows that the keys are significant ID subsets. Thus, finding keys is equivalent to finding significant ID subsets. Using the significance measure, we present Algorithm A with the time complexity $O(|A|^3|U|^2)$ to find one key. The algorithm can be run in parallel mode to compute all keys concurrently.

1. Information systems and databases

An *information system* $\mathcal{I}$ is a system $\langle U, A\rangle$, where:

(1) $U = \{u_1, u_2, \ldots, u_i, \ldots, u_{|U|}\}$ is a finite non-empty set, called a *universe* or an *object space*; elements of U are called identifiers or *objects*;

(2) $A = \{a_1, a_2, \ldots, a_l, \ldots, a_{|A|}\}$ is also a finite non-empty set; elements of A are called *attributes*;

(3) for every $a \in A$ there is a mapping a from U into some space, $a : U \to a(U)$, and

$$a(U) = \{a(u) \mid u \in U\}$$

is called the *domain* of attribute a.

An information system [6] is also called a *knowledge representation system*, or an *attribute-value system*. An information system can be intuitively expressed in terms of an *information table* (see Table 1).

Table 1

$U\backslash A$	a_1	a_2	...	a_l	...	$a_{\|A\|}$
u_1	$a_1(u_1)$	$a_2(u_1)$	...	$a_l(u_1)$	...	$a_{\|A\|}(u_1)$
u_2	$a_1(u_2)$	$a_2(u_2)$	...	$a_l(u_2)$	...	$a_{\|A\|}(u_2)$
...	...	...	...	...	...	...
u_i	$a_1(u_i)$	$a_2(u_i)$	...	$a_l(u_i)$	...	$a_{\|A\|}(u_i)$
...	...	...	...	...	...	...
$u_{\|U\|}$	$a_1(u_{\|U\|})$	$a_2(u_{\|U\|})$	...	$a_l(u_{\|U\|})$	...	$a_{\|A\|}(u_{\|U\|})$

The time complexity for computing an information system $\langle U, A\rangle$ is $|U| \times |A|$ since there are $|U| \times |A|$ values $a_l(u_i)$ to be computed, where $i = 1, 2, \ldots, |U|$; $l = 1, 2, \ldots, |A|$.

The concept of information systems is a generalization of the concept of a *relation* in *databases*.

For relational databases, a relational scheme is a finite set of attributes $A = \{a_1, a_2, \ldots, a_{|A|}\}$. Each attribute a_l $(l = 1, 2, \ldots, |A|)$ has a set of values, D_l, called its domain. A relation R on a relation scheme is a subset of $D_1 \times D_2 \times \cdots \times D_{|A|}$. A member of R is called a tuple.

Unlike relations in databases, an information system may consist of **duplicate** rows (tuples): they are labelled by different objects u_l and u_m and for any attribute they have identical values [4].

Using rough set theory, we can analyse dependencies and significancies of attributes for an information system to find (1) the keys—the minimal identity dependent sets, and (2) the core—the intersection of all keys. The core is equal to the set of significant attributes (see Fig. 1).

The *rough analysis* technique for finding the core and keys is of fundamental importance to artificial intelligence and cognitive sciences, especially in the areas of machine learning, knowledge acquisition, decision analysis, knowledge discovery from databases, expert systems, decision support systems, inductive reasoning, and pattern recognition. Currently, the rough set methodology is being used, among other areas, in market research, medical data analysis, sensor data analysis for the purpose of control, and research leading to the design of new composite materials. The analysis of stock market data has confirmed some well-known market rules and has led to the discovery of some interesting new rules [7].

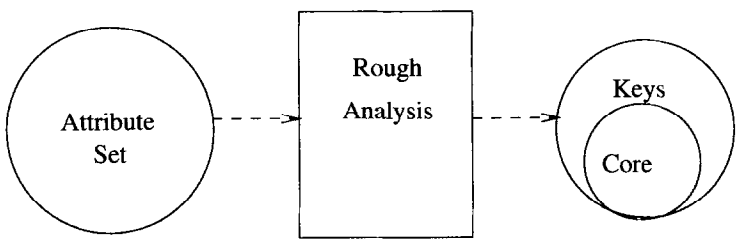

Fig. 1. Rough analysis.

This paper describes the rough analysis technique and proposes a series of algorithms to implement it.

2. Classification for one attribute

Classification is the basic tool for rough analysis.

Let $\langle U, A\rangle$ be an information system, where $U = \{u_1, u_2, \ldots, u_{|U|}\}$, a set of objects, and $A = \{a_1, a_2, \ldots, a_{|A|}\}$, a set of attributes.

With each attribute $a_1, a_2, \ldots, a_l, \ldots, a_{|A|}$ in A we associate an equivalence relation θ_a on U by $u\theta_a v$ if and only if $a(u) = a(v)$ for all $u, v \in U$; i.e., objects u and v have the same value for attribute a. The equivalence relation θ_a gives a *classification* (denoted by U/θ_a or U/a) $\{X_1, X_2, \ldots, X_{|U/a|}\}$ on universe U such that u and v in U are in the same class X if and only if they have the same value of attribute a, and vice versa. That is, let $u^{\theta_a} = \{v \in U \mid a(v) = a(u)\}$. Then we get a collection $\{u^{\theta_a} \mid u \in U\}$ of subsets of U. The collection $\{u^{\theta_a} \mid u \in U\}$ is a classification and $v_1, v_2 \in U$ are in the same subset (class) $u^{\theta_a} = \{v \in U \mid a(v) = a(u)\}$ for a $u \in U$ if and only if $a(v_1) = a(v_2) = a(u)$:

(i) $u^{\theta_a} \neq \emptyset$,

(ii) $u_i^{\theta_a} \cap u_j^{\theta_a} = \emptyset$ if $u_i^{\theta_a} \neq u_j^{\theta_a}$,

(iii) $\bigcup_{u \in U} u^{\theta_a} = U$.

Thus, we have classification U/a on U for an attribute a; namely, $u\theta_a v$ if and only if $a(u) = a(v)$ (see Fig. 2).

In Fig. 2, we demonstrate graphically the two classes of U based on attribute a in the first figure, and the three classes on a different attribute, b, in the second. There are two kinds of classification—*static* and *dynamic*. Static classification is classifying a set which has previously been fully input. In dynamic classification the set to be classified is made available incrementally, element by element.

An algorithm for dynamic classification has previously been proposed [2]. Here we present an algorithm for static classification.

Algorithm E. Let $\langle U, A\rangle$ be an information system, where $U = \{u_1, u_2, \ldots, u_n\}$ $(n > 0)$.

Given an attribute $a \in A$, this algorithm finds the corresponding classification U/a such that $a(u_i) = a(u_j)$ if and only if u_i and u_j are in the same class $(1 \leqslant i, j \leqslant n)$.

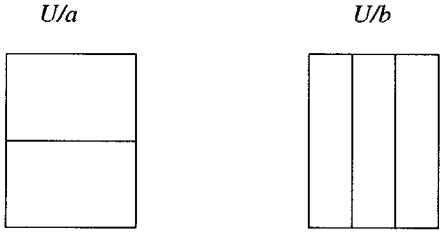

Fig. 2. Classification of a set of objects can be different for different attributes.

We use $u_i^{\theta} \ni u_j$ to represent the fact that u_j is classified into class $u_i^{\theta_a}$. We check $u_1, u_2, \ldots, u_n$ from the first object u_1 to the last object u_n. That is, we check object u_i for $i = 1, 2, \ldots, n$.

To establish class $u_i^{\theta_a}$, the idea is to check u_j for $j = i, i+1, \ldots, n$. This classification can be speeded up as follows. When we *first* meet a j such that $a(u_j) \neq a(u_i)$ and u_j is not classified, we set this j as the next u_i (by $I \leftarrow j, i \leftarrow I$) to be checked (to establish class u_i^{θ}). Thus, we need an auxiliary variable I to remember the first j.

In the step following this (see (E2) below) we establish class $u_i^{\theta_a}$. So we set $I = i$ (> 0) and then set $I \leftarrow 0$ when we enter this step.

(E1) [*Start from* $u_1^{\theta_a}$]. Set $I \leftarrow 1$ (to check u_1 and to establish class $u_1^{\theta_a}$).

(E2) [*Start to set* $u_i^{\theta_a}$]. Set $i \leftarrow I, I \leftarrow 0$. Set $j \leftarrow i$ (to check u_i for $j = i, i+1, i+2, \ldots, n$). Classify $u_i^{\theta_a} \ni u_j$. (First of all, classify $u_i^{\theta_a} \ni u_i$.)

(E3) [$j = n$?]. (Is class $u_i^{\theta_a}$ complete?) If $j < n$ (then class $u_i^{\theta_a}$ is not complete) go to step (E5) (to check next u_j). If $j = n$ then **class** $u_i^{\theta_a}$ **is complete**, go to (E4).

(E4) [$i = n$?]. (Now $j = n$, class $u_i^{\theta_a}$ is complete; i.e., the classification for $u_i^{\theta_a}$ is complete.) If $i < n$ and $I > 0$ (we have ever met a u_j such that $a(u_j) \neq a(u_i)$ and u_j is not classified and we have already set $I \leftarrow j$ at (E9)) go to (E2) (to set next $u_i^{\theta_a}$ by $i \leftarrow I$).

If $i < n$ and $I = 0$ (we have never met a u_j such that both $a(u_j) \neq a(u_i)$ and u_j is not classified; i.e., we only meet such u_j that either (1) $a(u_j) = a(u_i)$, or (2) $a(u_j) \neq a(u_i)$ and u_j is classified) the classification for θ_a is **completely finished**.

If $i = n$ (now $i = j = n$) the classification for θ_a is **completely finished**.

(E5) [*Increase* j]. $j \leftarrow j + 1$.

(E6) [$a(u_i) = a(u_j)$?]. If $a(u_i) = a(u_j)$ go to (E10). Otherwise go to (E7).

(E7) [$I > 0$?]. (Now $a(u_j) \neq a(u_i)$. Is this the first j?). If $I > 0$ (then this is not the first j so we simply ignore this j and) go to (E3) (to check next j for $u_i^{\theta_a}$). Otherwise go to (E8).

(E8) [*Is* u_j *classified*?] (Now $I = 0$.) If u_j is classified then go to (E3) (to check next j for $u_i^{\theta_a}$). If u_j is not classified then go to (E9).

(E9) [*Set* $I \leftarrow j$]. (Now $I = 0$ and u_j is not classified.) Set $I \leftarrow j$ and go to (E3) (to check next j for $u_i^{\theta_a}$).

(E10) [*Classify* $u_i^{\theta_a} \ni u_j$]. Classify $u_i^{\theta_a} \ni u_j$. Go to (E3) (to check if class $u_i^{\theta_a}$ is complete).

The time complexity of Algorithm E to classify U is $O(|U|^2)$. In the worst case, we need to check objects as follows.

(1) To establish $u_1^{\theta_a}$, to check $|U| - 1$ objects: $u_2, u_3, \ldots, u_{|U|}$.

(2) To establish $u_2^{\theta_a}$, to check $|U| - 2$ objects: $u_3, u_4, \ldots, u_{|U|}$. And so on.

(3) To establish $u_{|U|-1}^{\theta_a}$, to check 1 object $u_{|U|}$.

Table 2
Skull information system, where x_1—teeth-size, x_2—type (a crude prima facie classification), x_3—location, x_4—morphology, x_5—skull-size (100 cc), x_6—sex

Specimen #	x_1	x_2	x_3	x_4	x_5	x_6
# 45	4	1	K	X	8.36	M
# 92	3	2	J	Y	5.14	F
#163	4	1	L	X	8.38	M
#167	3	1	K	X	8.29	F
#181	4	2	J	Y	5.27	M

So the time complexity of Algorithm E to classify U is

$$(|U|-1)+(|U|-2)+\cdots+1=\frac{|U|(|U|-1)}{2}=\mathrm{O}(|U|^2).$$

The algorithm can be run in parallel mode to compute concurrently all corresponding classifications from many attributes.

Example 2.1. In a paleontological investigation we might have the following information system (see Table 2).

Here we have $n=|U|=5$: $U=\{u_1,u_2,u_3,u_4,u_5\}$, where $u_1=\#45, u_2=\#92, u_3=\#163, u_4=\#167, u_5=\#181$, and $A=\{x_1,x_2,x_3,x_4,x_5,x_6\}$.

For every attribute $a\in\{x_1,x_2,x_3,x_4,x_5,x_6\}$ in A we can introduce a classification $U/\theta_a: U/\theta_{x_1}, U/\theta_{x_2}, U/\theta_{x_3}, U/\theta_{x_4}, U/\theta_{x_5}, U/\theta_{x_6}$, in universe U as follows: two objects $u,v\in U$ are in the same class if and only if $a(u)=a(v)$.

We can find all classifications for this skull information system by using Algorithm E:

- $U/\theta_{x_1}=\{V_{11},V_{12}\}$,where

  $V_{11}=\{\#45,\#163,\#181\}$, $V_{12}=\{\#92,\#167\}$;

- $U/\theta_{x_2}=\{V_{21},V_{22}\}$, where

  $V_{21}=\{\#45,\#163,\#167\}$, $V_{22}=\{\#92,\#181\}$;

- $U/\theta_{x_3}=\{V_{31},V_{32},V_{33}\}$, where

  $V_{31}=\{\#45,\#167\}$, $V_{32}=\{\#92,\#181\}$, $V_{33}=\{\#163\}$;

- $U/\theta_{x_4}=\{V_{41},V_{42}\}$, where

  $V_{41}=\{\#45,\#163,\#167\}$, $V_{42}=\{\#92,\#181\}$;

- $U/\theta_{x_5}=\{V_{51},V_{52},V_{53},V_{54},V_{55}\}$, where

  $V_{51}=\{\#45\}$, $V_{52}=\{\#92\}$, $V_{53}=\{\#163\}$, $V_{54}=\{\#167\}$, $V_{55}=\{\#181\}$;

- $U/\theta_{x_6}=\{V_{61},V_{62}\}$, where

  $V_{61}=\{\#45,\#163,\#181\}$, $V_{62}=\{\#92,\#167\}$.

Notice that $U/\theta_{x_1}=U/\theta_{x_6}$ and that $U/\theta_{x_2}=U/\theta_{x_4}$.

3. Classification intersections between many attributes

Before discussing classification intersections between many attributes, we need to introduce the intersection operation on classifications.

3.1. The intersection operation on classifications

Let θ_1, θ_2 be two equivalence relations on U. The *intersection* $\theta_1 \cap \theta_2$ of two equivalence relations θ_1 and θ_2 is defined as follows: $u(\theta_1 \cap \theta_2)v$ if and only if $u\theta_1 v$ and $u\theta_2 v$.

For an equivalence θ, denote the corresponding classification by U/θ. For a $u \in U$, denote $u^\theta = \{v \in U \mid v\theta u\}$, the class in U/θ containing u. Then $U/\theta = \{u^\theta \mid u \in U\}$, i.e., classification U/θ comprises different classes u^θ for all $u \in U$.

For the intersection operation, we have $u^{\theta_1 \cap \theta_2} = u^{\theta_1} \cap u^{\theta_2}$.

Let θ_1, θ_2 be two equivalence relations on U. We say that θ_1 is *included* in θ_2, denoted by $\theta_1 \subseteq \theta_2$, if $\theta_1 \cap \theta_2 = \theta_1$. We say that θ_1 is *strictly included* in θ_2, denoted by $\theta_1 \subset \theta_2$, if $\theta_1 \cap \theta_2 = \theta_1$ and $\theta_1 \neq \theta_2$.

Now, $\theta_1 \subseteq \theta_2$ can be described alternatively by stating that relation θ_1 is *stronger* than relation θ_2: $u\theta_1 v$ *implies* $u\theta_2 v$ for all $u, v \in U$.

We denote $\theta_1 \subseteq \theta_2$ by $\theta_1 \models \theta_2$; denote the identity equivalence by ε: $u\varepsilon v$ if and only if $u = v$ for $u, v \in U$; and denote the universal equivalence by δ: $u\delta v$ for all $u, v \in U$ [11].

Then, ε is the least equivalence in the following sense: $\varepsilon \subseteq \theta$ for every equivalence θ; δ is the greatest equivalence: $\delta \supseteq \theta$ for every equivalence θ.

Let U/θ_1 and U/θ_2 be two classifications on U with the respective equivalence relations θ_1 and θ_2 on U. The *intersection* $\cap$ between two classifications U/θ_1 and U/θ_2 is defined as follows: $U/\theta_1 \cap U/\theta_2 = U/\theta_1 \cap \theta_2$ (also called classification "U/θ_1 *AND* U/θ_2").

Let U/θ_1 and U/θ_2 be two classifications corresponding to equivalence relations θ_1, θ_2 on U, respectively. We say that U/θ_1 is *included* in U/θ_2, denoted by $U/\theta_1 \subseteq U/\theta_2$, if $\theta_1 \subseteq \theta_2$. We say that U/θ_1 is *strictly included* in U/θ_2, denoted by $U/\theta_1 \subset U/\theta_2$, if $\theta_1 \subseteq \theta_2$ and $\theta_1 \neq \theta_2$.

Now, $U/\theta_1 \subseteq U/\theta_2$ if and only if $\theta_1 \cap \theta_2 = \theta_1$; i.e., $u^{\theta_1} = u^{\theta_1} \cap u^{\theta_2}$ for all $u \in U$; i.e., $u^{\theta_1} \subseteq u^{\theta_2}$ for all $u \in U$.

So $U/\theta_1 \subseteq U/\theta_2$ if and only if U/θ_1 is *finer* than U/θ_2; i.e., every class X_1 in classification U/θ_1 is *included* in a class X_2 in classification U/θ_2: $X_1 \subseteq X_2$.

If $U/\theta_1 \subseteq U/\theta_2$ then $|U/\theta_1| \geqslant |U/\theta_2|$.

We also know that $U/\varepsilon = \{\{u\} \mid u \in U\}$ and $|U/\varepsilon| = |U|$ and that $U/\delta = \{U\}$ and $|U/\delta| = 1$.

3.2. Intersection for many attribute classifications

Now, consider a subset $X \subseteq A$. There is an equivalence $\bigcap_{a \in X} \theta_a$ corresponding to X as follows: two objects $u, v \in U$ are equivalent if and only if $a(u) = a(v)$ for every $a \in X$. Let us denote $\theta_X = \bigcap_{a \in X} \theta_a$. Also, remember that U/θ_X can be denoted by U/X (see Fig. 3).

For the empty set $\emptyset$, we take $\theta_\emptyset = \delta$.

Theorem 3.1 (Grzymala-Busse [4], Pawlak and Rauszer [8]). *Let $\langle U, A\rangle$ be an information system. Then, for $X, Y \subseteq A$ we have $\theta_X \cap \theta_Y = \theta_{X \cup Y}$.*

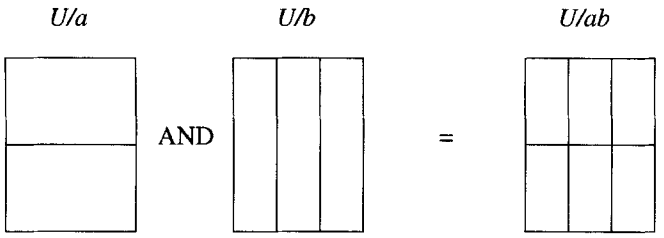

Fig. 3. Classification intersection.

Proof. We find that

$$\begin{aligned}\theta_X \cap \theta_Y &= \Big(\bigcap_{a\in X}\theta_a\Big)\cap\Big(\bigcap_{a\in Y}\theta_a\Big)\\ &= \Big(\Big(\bigcap_{a\in X-X\cap Y}\theta_a\Big)\cap\Big(\bigcap_{a\in X\cap Y}\theta_a\Big)\Big)\cap\Big(\Big(\bigcap_{a\in Y-X\cap Y}\theta_a\Big)\cap\Big(\bigcap_{a\in X\cap Y}\theta_a\Big)\Big)\\ &= \Big(\bigcap_{a\in X-X\cap Y}\theta_a\Big)\cap\Big(\bigcap_{a\in Y-X\cap Y}\theta_a\Big)\cap\Big(\Big(\bigcap_{a\in X\cap Y}\theta_a\Big)\cap\Big(\bigcap_{a\in X\cap Y}\theta_a\Big)\Big)\\ &= \Big(\bigcap_{a\in X-X\cap Y}\theta_a\Big)\cap\Big(\bigcap_{a\in Y-X\cap Y}\theta_a\Big)\cap\Big(\bigcap_{a\in X\cap Y}\theta_a\Big) = \bigcap_{a\in X\cup Y}\theta_a. \qquad \square\end{aligned}$$

Notice that subsets of attributes are often discussed in databases, and XY is a conventional shorthand for $X \cup Y$, where $X, Y \subseteq A$. So, this theorem can be rewritten as $\theta_X \cap \theta_Y = \theta_{XY}$, and we have the following algorithm for computing intersection θ_{XY}.

Algorithm I. Let $U/\theta_X = \{X_1, X_2, \ldots, X_i, \ldots, X_s\}, 1 \leqslant s \leqslant |U|$; $U/\theta_Y = \{Y_1, Y_2, \ldots, Y_j, \ldots, Y_t\}, 1 \leqslant t \leqslant |U|$.

This algorithm gives classification $U/\theta_{XY} = \{V_1, V_2, \ldots, V_k, \ldots, V_r\}, 1 \leqslant r \leqslant st$.

We use the following pointers:

$i = 1, 2, \ldots, s$ points to X_i,

$j = 1, 2, \ldots, t$ points to Y_j,

r records that we have found r classes $V_1, V_2, \ldots, V_r$ of $U/\theta_1 \cap \theta_2$.

For every i and every j, we check whether or not $X_i \cap Y_j = \emptyset$. If $X_i \cap Y_j = \emptyset$ then we simply ignore it. Otherwise, we establish a new class: $r \leftarrow r + 1$, $V_r = X_i \cap Y_j$.

(I1) [*Initialize*]. Set $i \leftarrow 1, j \leftarrow 1, r \leftarrow 0$.

(I2) [$X_i \cap Y_j = \emptyset$?]. If intersection $X_i \cap Y_j = \emptyset$ then go to (I3) to check the next intersection.
Otherwise, set $r \leftarrow r + 1$ and establish a new class $V_r = X_i \cap Y_j$ for $U/\theta_1 \cap \theta_2$. Go to (I3) to check the next intersection.

(I3) [$j = t$?]. If $j = t$ then go to (I5) to check next i.
Otherwise, go to next step (I4) to see next j.

(I4) [*Increase* j]. Set $j \leftarrow j + 1$. Go to (I2).

(I5) [$i = s$?]. If $i = s$ then **the classification is completed**, and we have $U/\theta_{XY} = \{V_1, V_2, \ldots, V_r\}$. Otherwise, go to (I6).
(I6) [*Increase* i]. Set $i \leftarrow i + 1$, $j \leftarrow 1$. Go to (I2).

We know that the time complexity of Algorithm I is $O(|U|^2)$ since there are $s \times t$ ($\leqslant |U| \times |U|$) intersections $X_i \cap Y_j$ to be calculated. From the definition of intersection and inclusion, we have the following.

(1) For $X, Y \subseteq A$ we have $\theta_{X\cup Y} = \theta_{XY} \subseteq \theta_X, \theta_Y$.
(2) (*Reflexivity*). For $X, Y \subseteq A$, if $X \supseteq Y$ then $\theta_X \subseteq \theta_Y$.
(3) (*Augmentation*). For $X, Y \subseteq A$, if $\theta_X \subseteq \theta_Y$ then $\theta_{X\cup Z} \subseteq \theta_{Y\cup Z}$ for $Z \subseteq A$. (Thus, if $\theta_X = \theta_Y$ then $\theta_{X\cup Z} = \theta_{Y\cup Z}$ for $Z \subseteq A$.)

Example 3.1. From Example 2.1, the following results are obtained for the skull information system.

(1) $\theta_{x_1} = \theta_{x_6}$ and $\theta_{x_2} = \theta_{x_4}$. So we take $A = \{x_1, x_2, x_3, x_5\}$. For convenience, we rewrite $A = \{y_1, y_2, y_3, y_4\}$, where $y_1 = x_1$, $y_2 = x_2$, $y_3 = x_3$, $y_4 = x_5$.
(2) Also, notice that $\theta_{y_4} = \varepsilon$, the identity equivalence.
So is $\theta_{\{y_1,y_4\}} = \theta_{\{y_2,y_4\}} = \theta_{\{y_3,y_4\}} = \varepsilon$.
(3) $U/\theta_{y_1} = \{V_{11}, V_{12}\}$, where

$$V_{11} = \{\#45, \#163, \#181\}, \quad V_{12} = \{\#92, \#167\}.$$

(4) $U/\theta_{y_2} = \{V_{21}, V_{22}\}$, where

$$V_{21} = \{\#45, \#163, \#167\}, \quad V_{22} = \{\#92, \#181\}.$$

(5) $U/\theta_{y_3} = \{V_{31}, V_{32}, V_{33}\}$, where

$$V_{31} = \{\#45, \#167\}, \quad V_{32} = \{\#92, \#181\}, \quad V_{33} = \{\#163\}.$$

(6) $U/\theta_{\{y_1,y_2\}} = \{V_{51} \cup V_{53}, V_{52}, V_{54}, V_{55}\}$.
(7) $U/\theta_{\{y_1,y_3\}} = \{V_{51}, V_{52}, V_{53}, V_{54}, V_{55}\}$. That is, $\theta_{\{y_1,y_3\}}$ is ε. So is $\theta_{\{y_1,y_2,y_3\}} = \varepsilon$.
(8) $U/\theta_{\{y_2,y_3\}} = \{V_{51} \cup V_{54}, V_{52} \cup V_{55}, V_{53}\}$.

4. Functional dependencies

Using classification, we can analyze dependencies between two subsets of attributes. For an information system $\mathcal{I} = \langle U, A\rangle$, let $u, v \in U$ and $X \subseteq A$. We denote $X(u) = X(v)$ if and only if $x(u) = x(v)$ for all $x \in X$.

Definition 4.1. A *functional dependency* (FD) between two subsets $X, Y \subseteq A$ of attributes in an information system $\mathcal{I} = \langle U, A\rangle$ is a statement, denoted by $X \to Y$, which holds in the information system $\mathcal{I}$, if and only if, for every pair $u, v \in U$ we have that $X(u) = X(v)$ implies $Y(u) = Y(v)$ (see Fig. 4).

Obviously, $X \supseteq Y$ implies $X \to Y$ (see Fig. 5).

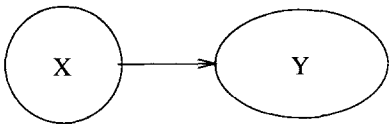

Fig. 4. Functional dependency.

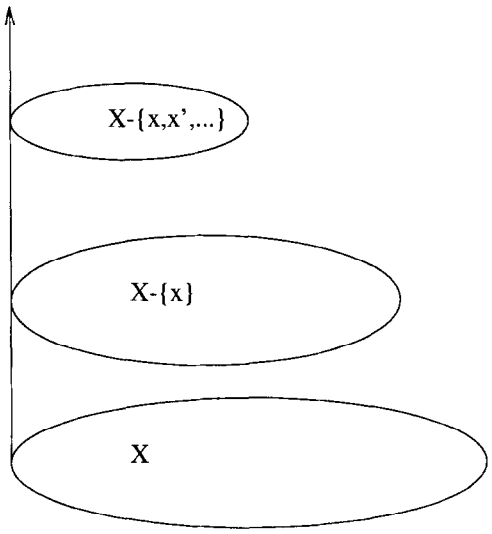

Fig. 5. Functional dependency for subsets.

Theorem 4.1 (Grzymala-Busse [4], Pawlak [6]). *A functional dependency* $X \to Y$ *can be defined as* $\theta_X \subseteq \theta_Y$; *i.e.,* $\bigcap_{a\in X} \theta_a \subseteq \bigcap_{a\in Y} \theta_a$.

Proof. By Definition 4.1 we know that $X \to Y$, if and only if, for every pair $u, v \in U$ we have that $a(u) = a(v)$ for $a \in X$ implies $a(u) = a(v)$ for $a \in Y$; i.e., $u\theta_X v$ implies $u\theta_Y v$; i.e., $u(\bigcap_{a\in X} \theta_a)v$ implies $u(\bigcap_{a\in Y} \theta_a)v$; i.e., $(\bigcap_{a\in X} \theta_a) \models (\bigcap_{a\in Y} \theta_a)$ by definition; i.e., $(\bigcap_{a\in X} \theta_a) \subseteq (\bigcap_{a\in Y} \theta_a)$. □

Thus, we have the following algorithm for checking whether or not $X \to Y$ for $X, Y \subseteq A$.

Algorithm O. Let $\langle U, A\rangle$ be an information system. Let $X, Y \subseteq A$.

Suppose that

$$U/\theta_X = \{X_{11}, X_{12}, \ldots, X_{1i}, \ldots, X_{1|\pi_1|}\}, |\pi_1| \leqslant |U|;$$
$$U/\theta_Y = \{X_{21}, X_{22}, \ldots, X_{2j}, \ldots, X_{2|\pi_2|}\}, |\pi_2| \leqslant |U|,$$

this algorithm checks whether or not $X \to Y$.

(O1) [*Initialize*]. Set $i \leftarrow 1$, $j \leftarrow 1$.

(O2) [$X_{1i} \subseteq X_{2j}$?]. If $X_{1i} \subseteq X_{2j}$ then (for this i we have already found a j such that $X_{1i} \subseteq X_{2j}$ so) go to (O3) (to check next i).
If $X_{i1} \not\subseteq X_{2j}$ then go to (O5) (to check next j).

(O3) [$i = |\pi_1|$?]. If $i = |\pi_1|$ then the algorithm is completed with the **answer:** $X \to Y$.
If $i < |\pi_1|$ then go to (O4).

(O4) [*Increase* i]. Set $i \leftarrow i + 1$, go to (O2).

(O5) [$j = |\pi_2|$?]. If $j = |\pi_2|$ (now we have an i such that $X_{1i} \not\subseteq X_{2j}$ for $j = 1, 2, \ldots, |\pi_2|$) then the algorithm is completed with the **answer:** $X \not\to Y$.
If $j < |\pi_2|$ then go to (O6).

(O6) [*Increase* j]. Set $j \leftarrow j + 1$, go to (O2).

Algorithm O should make $|\pi_1| \times |\pi_2| \leqslant |U|^2$ comparisons in step (O2). So its time complexity is $O(|U|^2)$.

Example 4.1. For the skull information system, we have the following:

$x_1 \to x_6, x_2 \to x_4$; i.e.,
$\theta_{x_1} \subseteq \theta_{x_6}, \theta_{x_2} \subseteq \theta_{x_4}$; i.e.,
$x_1(u) = x_1(v)$ implies $x_6(u) = x_6(v)$, and
$x_2(u) = x_2(v)$ implies $x_4(u) = x_4(v)$ for every $u, v \in U$.

5. Identity dependencies and keys

Definition 5.1. An *identity dependency* (ID) between two subsets $X, Y \subseteq A$ of attributes in an information system $\mathcal{I} = \langle U, A \rangle$ is a statement, denoted by $X \leftrightarrow Y$, which holds in the information system $\mathcal{I}$, if and only if, both $X \to Y$ and $Y \to X$ hold (see Fig. 6).

From Theorem 4.1, an identity dependency $X \leftrightarrow Y$ can be defined as $\theta_X = \theta_Y$; i.e., $\bigcap_{a \in X} \theta_a = \bigcap_{a \in Y} \theta_a$.

Example 5.1. For the skull information system: $x_1 \leftrightarrow x_6, x_2 \leftrightarrow x_4$, and $\{x_1, x_3\} \leftrightarrow \{x_5\} \leftrightarrow \{x_1, x_2, x_3, x_5\}$, where x_1—teeth-size, x_2—type, x_3—location, x_4—morphology, x_5—skull-size and x_6—sex.

Algorithm S. Let $\langle U, A \rangle$ be an information system. Let $X, Y \subseteq A$. Suppose that

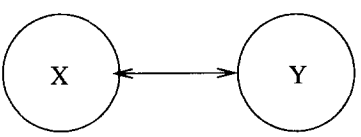

Fig. 6. Identity dependency.

$$U/\theta_X = \{X_{11}, X_{12}, \ldots, X_{1i}, \ldots, X_{1|\pi_1|}\}, |\pi_1| \leqslant |U|;$$
$$U/\theta_Y = \{X_{21}, X_{22}, \ldots, X_{2j}, \ldots, X_{2|\pi_2|}\}, |\pi_2| \leqslant |U|.$$

This algorithm checks whether or not $X \leftrightarrow Y$.

(S1) [*Initialize*]. Set $i \leftarrow 1$, $j \leftarrow 1$.

(S2) [$X_{1i} = X_{2j}$?]. If $X_{1i} = X_{2j}$ then (for this i we have already found a j such that $X_{1i} = X_{2j}$ so) go to (S3) (to check next i).
If $X_{i1} \neq X_{2j}$ then go to (S5) (to check next j).

(S3) [$i = |\pi_1|$?]. If $i = |\pi_1|$ then the algorithm is completed with the **answer:** $X \leftrightarrow Y$.
If $i < |\pi_1|$ then go to (S4).

(S4) [*Increase* i]. Set $i \leftarrow i + 1$, go to (S2).

(S5) [$j = |\pi_2|$?]. If $j = |\pi_2|$ (now we have an i such that $X_{1i} \neq X_{2j}$ for $j = 1, 2, \ldots, |\pi_2|$) then the algorithm is completed with the **answer:** $X \not\leftrightarrow Y$.
If $j < |\pi_2|$ then go to (S6).

(S6) [*Increase* j]. Set $j \leftarrow j + 1$, go to (S2).

Algorithm S should make $|\pi_1| \times |\pi_2| \leqslant |U|^2$ comparisons in step (S2). So its time complexity is $O(|U|^2)$.

Now we introduce keys in an information system.

Definition 5.2. Let X be a (non-empty or empty) subset $X \subseteq A$ of A. A subset X_0 of X is said to be a *key* of X if X_0 satisfies:

(1) ID: $\theta_{X_0} = \theta_X$; i.e., $X_0 \leftrightarrow X$;

(2) Minimal: if $X' \subset X_0$ then $\theta_X \subset \theta_{X'}$; i.e., if $X' \subset X_0$ then $X' \not\leftrightarrow X$.

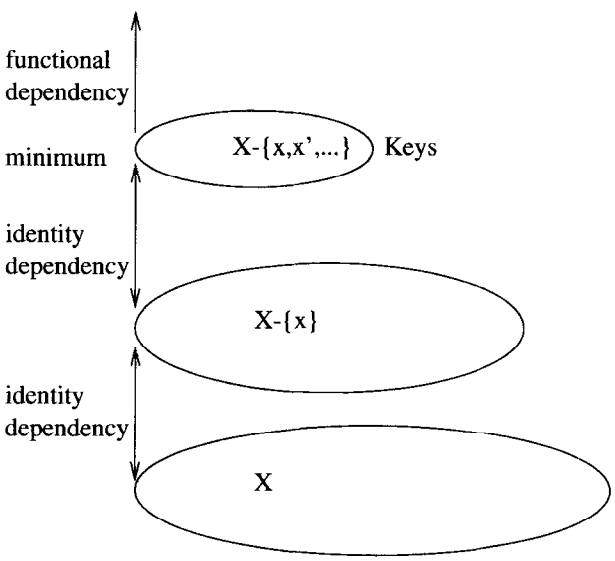

Fig. 7. Keys: minimal ID subsets.

The empty subset $\emptyset$ has key $\emptyset$ (see Example 5.2 below). That is, X_0 is a minimal identity dependent subset of X, i.e., we have

$$X \leftrightarrow \cdots \leftrightarrow X_0 \rightarrow X' \rightarrow \cdots$$

for

$$X \supseteq \cdots \supseteq X_0 \supset X' \supseteq \cdots$$

(see Fig. 7).

Example 5.2. Let $X = \emptyset$. Then X has the unique key $X_0 = X = \emptyset$. Indeed, subset X_0 of X satisfies:

(1) $\theta_{X_0} = \theta_X = \theta_\emptyset = \delta$;
(2) there are no $X' \subset X_0$ since $X_0 = \emptyset$.

6. Finding all keys for a database

Let $\mathcal{I} = \langle U, A \rangle$ be an information system, where U is universe and A is the set of attributes. We want to find all its keys—that is, all subsets A_0: $A_{01}, A_{02}, \ldots, A_{0s}$ of A such that:

(1) $\theta_{A_0} = \theta_A$; i.e., $A_0 \leftrightarrow A$; and
(2) if $A' \subset A_0$ then $\theta_A \subset \theta_{A'}$; i.e., if $A' \subset A_0$ then $A' \not\leftrightarrow A$.

Algorithm K (Grzymala-Busse [4]). This algorithm finds all keys of A by searching from singletons to A.

If $A = \emptyset$ then $\emptyset$ is the unique key of A. Let $A = \{a_1, a_2, \ldots, a_j, \ldots, a_{|A|}\}$, $|A| > 0$. We need to check all the subsets of A.

Let us denote the binomial coefficients by $C_k^l = l!/k!(l-k)!$.

(1) Let us denote $C_1^{|A|} = |A|$ singletons, one-attribute subsets, by

$$A_{11} = \{a_1\}, \quad A_{12} = \{a_2\}, \quad \ldots, \quad A_{1j} = \{a_j\}, \quad \ldots,$$
$$A_{1C_1^{|A|}} = \{a_{|A|}\}.$$

We can compute

$$\theta_{A_{11}} = \bigcap_{a \in A_{11}} \theta_a, \quad \theta_{A_{12}} = \bigcap_{a \in A_{12}} \theta_a, \quad \ldots,$$
$$\theta_{A_{1j}} = \bigcap_{a \in A_{1j}} \theta_a, \quad \ldots, \quad \theta_{A_{1C_1^{|A|}}} = \bigcap_{a \in A_{1C_1^{|A|}}} \theta_a.$$

(2) Let us denote $C_2^{|A|} = |A|(|A|-1)/2!$ two-attribute subsets by

$$A_{21} = \{a_1, a_2\}, \quad A_{22} = \{a_1, a_3\}, \quad \ldots,$$
$$A_{2j} = \{a_1, a_j\}, \quad \ldots, \quad A_{2C_2^{|A|}} = \{a_{|A|-1}, a_{|A|}\}.$$

We can compute

$$\theta_{A_{21}} = \bigcap_{a\in A_{21}} \theta_a, \quad \theta_{A_{22}} = \bigcap_{a\in A_{22}} \theta_a, \quad \ldots,$$
$$\theta_{A_{2j}} = \bigcap_{a\in A_{2j}} \theta_a, \quad \ldots, \quad \theta_{A_{2C_2^{|A|}}} = \bigcap_{a\in A_{2C_2^{|A|}}} \theta_a.$$

(3) Generally, let us denote $C_t^{|A|} = |A|!/t!(|A|-t)!$ t-attribute subsets by

$$A_{t1} = \{a_1, a_2, \ldots, a_t\}, \quad \ldots, \quad A_{tj}, \quad \ldots,$$
$$A_{tC_t^{|A|}} = \{a_{|A|-t+1}, \quad \ldots, \quad a_{|A|-1}, a_{|A|}\}.$$

We can compute

$$\theta_{A_{t1}} = \bigcap_{a\in A_{t1}} \theta_a, \quad \theta_{A_{t2}} = \bigcap_{a\in A_{t2}} \theta_a, \quad \ldots,$$
$$\theta_{A_{tj}} = \bigcap_{a\in A_{tj}} \theta_a, \quad \ldots, \quad \theta_{A_{tC_t^{|A|}}} = \bigcap_{a\in A_{tC_t^{|A|}}} \theta_a.$$

(4) Notice that $C_{|A|}^{|A|} = 1$ (i.e., the unique) $|A|$-attribute subset is $A_{|A|1} = \{a_1, a_2, \ldots, a_j, \ldots, a_{|A|}\} = A$, and $\theta_A = \bigcap_{a\in A_{|A|1}} \theta_a = \bigcap_{a\in A} \theta_a$.

The algorithm is to search subsets of A as follows: singletons, two-attribute subsets, ..., t-attribute subsets, and so on. Continue up to the unique $|A|$-attribute subset A itself.

Suppose that we have already found s keys $A_{01}, A_{02}, \ldots, A_{0k}, \ldots, A_{0s}$ so far. Then, a t-attribute subset A_{tj} ($j = 1, 2, \ldots, C_t^{|A|}$) is the next key $A_{0,s+1}$ if:

(1) $A_{tj} \not\supset A_{0k}$ for $k = 1, 2, \ldots, s$. (Otherwise $A_{tj} \supset A_{0k}$ implies that A_{tj} has a proper subset A_{0k} such that $\bigcap_{a\in A} \theta_a = \bigcap_{a\in A_{0k}} \theta_a$; i.e., $\theta_A = \theta_{A_{0k}}$. So A_{tj} is not a key.)

(2) $\bigcap_{a\in A} \theta_a = \bigcap_{a\in A_{tj}} \theta_a$; i.e., $\theta_A = \theta_{A_{tj}}$.

Therefore, our algorithm includes the following steps, where:

(1) Steps (K2)–(K5) check $A_{tj} \not\supset A_{0k}$ for all $k = 1, 2, \ldots, s$;

(2) Step (K10) check $\bigcap_{a\in A} \theta_a = \bigcap_{a\in A_{tj}} \theta_a$; i.e., $\theta_A = \theta_{A_{tj}}$;

(3) Steps (K6)–(K9) set the next subset A_{tj}.

In the successful case (detected by (K2) and (K3)) we go to step (K10) to check $\bigcap_{a\in A} \theta_a = \bigcap_{a\in A_{tj}} \theta_a$ for this A_{tj}; i.e., $\theta_A = \theta_{A_{tj}}$. Otherwise (detected by (K5) and (K10)) we go to steps (K6)–(K9) to set the next subset A_{tj} (to check whether or not $\bigcap_{a\in A} \theta_a = \bigcap_{a\in A_{tj}} \theta_a$; i.e., $\theta_A = \theta_{A_{tj}}$).

We use the following variables:

- s—the number of keys we have already found,
- k—counting from 1 to s,
- t—we are currently searching t-attribute subset A_{tj},
- j—we are currently searching the jth subset A_{tj} in all t-attribute subsets

$$A_{t1}, \ldots, A_{tj}, \ldots, A_{tC_t^{|A|}}.$$

(K1) [*Initialize*]. Set $j \leftarrow 1, s \leftarrow 0, t \leftarrow 1$. Compute $\theta_A = \bigcap_{a\in A} \theta_a$.

(K2) [$s = 0$?]. If $s = 0$ go to (K10). If $s > 0$ then set $k \leftarrow 1$ and go to (K5).

(K3) [$k = s$?]. If $k = s$ (so $A_{tj} \not\supset A_{0k}$ for all $k = 1, 2, \ldots, s$, where

$$A_{01}, \quad A_{02}, \quad \ldots, \quad A_{0k}, \quad \ldots, \quad A_{0s}$$

are all keys we have found so far) go to (K10). If $k < s$ go to (K4).

(K4) [*Increase* k]. Set $k \leftarrow k + 1$. (Go to (K5) to check next A_{0k}.)

(K5) [$A_{tj} \supset A_{0k}$?]. If $A_{tj} \supset A_{0k}$ (a key found previously), then (it is impossible for A_{tj} to be a key, so ignore it and) go to (K6) (to check the next subset A_{tj}). If $A_{tj} \not\supset A_{0k}$ go to (K3) (to check next A_{0k}).

(K6) [$j = C_t^{|A|}$?]. If $j = C_t^{|A|}$ go to (K8). If $j < C_t^{|A|}$ go to (K7).

(K7) [*Increase* j]. Set $j \leftarrow j + 1$. Go to (K2) (to check the next subset A_{tj}: whether or not $A_{tj} \supset A_{0k}$ for $k = 1, 2, \ldots, s$).

(K8) [$t = |A|$?]. If $t = |A|$ then (searching is complete and **all keys** are obtained: $A_{01}, A_{02}, \ldots, A_{0k}, \ldots, A_{0s}$. So) **the algorithm terminates**. If $t < |A|$ go to (K9).

(K9) [*Increase* t]. Set $t \leftarrow t + 1$, $j \leftarrow 1$. Go to (K2) (to check the next subset A_{tj}: whether or not $A_{tj} \supset A_{0k}$ for $k = 1, 2, \ldots, s$).

(K10) [$\theta_A = \theta_{A_{tj}}$?]. If $\bigcap_{a \in A} \theta_a = \bigcap_{a \in A_{tj}} \theta_a$ (then A_{tj} is a key) go to (K11). If $\bigcap_{a \in A} \theta_a \subset \bigcap_{a \in A_{tj}} \theta_a$ (then A_{tj} is not a key) go to (K6) (to check the next subset A_{tj}: whether or not $\bigcap_{a \in A} \theta_a = \bigcap_{a \in A_{tj}} \theta_a$; i.e., $\theta_A = \theta_{A_{tj}}$).

(K11) [*Found an* A_0]. **Found a key:** set $s \leftarrow s + 1$ and $A_{0s} \leftarrow A_{tj}$. Go to (K6) (to check the next subset A_{tj}: whether or not $\bigcap_{a \in A} \theta_a = \bigcap_{a \in A_{tj}} \theta_a$; i.e., $\theta_A = \theta_{A_{tj}}$).

Example 6.1. Consider the skull information system in Example 2.1. For convenience (see Example 3.1), we rewrite $A = \{y_1, y_2, y_3, y_4\}$, where $y_1 = x_1$, $y_2 = x_2$, $y_3 = x_3$, $y_4 = x_5$.

We can find all keys $A_{01} = \{y_4\}$, $A_{02} = \{y_1, y_3\}$ for the skull information system by using Algorithm K.

This tells us that the skull size y_4 can be used as an identifier of specimens, and that a combination of teeth size y_1 and location y_3 could serve the same purpose. The latter fact is probably more interesting—it says that there are no two specimens at a given location which have the same teeth size. This may not mean much to the paleontologists—but it is easy to envisage applications where such information can give insights into the ontology being studied.

The time complexity of Algorithm K for finding all keys of A is exponential since the algorithm checks all subsets in 2^A, and $|2^A| = 2^{|A|}$. Moreover, when the algorithm checks one subset $A_{tj} \subseteq A$ to see if $\theta_A = \theta_{A_{tj}}$ in (K10), we need to compute equivalence $\theta_{A_{tj}} = \bigcap_{a \in A_{tj}} \theta_a$.

We know that the time complexity of Algorithm E for finding the corresponding classification of one θ_a is $|U|^2$. So the price to find $|A_{tj}| \leqslant |A|$ equivalence relations θ_a is $|U|^2|A|$.

To compute $\bigcap_{a \in A_{tj}} \theta_a$, we need $|A_{tj}| - 1$ intersections. And the time complexity to compute one intersection is $O(|U|^2)$. So the price to find $|A_{tj}| - 1$ intersections is $(|A_{tj}| - 1) \times O(|U|^2) = O(|U|^2|A|)$.

Thus, the time complexity of Algorithm K is

$$2^{|A|} \times O(|U|^2|A| + |U|^2|A|) = O(2^{|A|}|A||U|^2).$$

7. Significance and core

In order to reduce the exponential complexity of rough analyses, we investigate the ID significance of an attribute x in an attribute set X ($\subseteq A$): attribute x is significant if X and $X - \{x\}$ are not ID, and x is not significant if X and $X - \{x\}$ are ID.

Intuitively, some attributes are not significant in a representation, in the sense that their removal has no real impact on the value of the representation of objects. If it is not significant, we can simply remove an attribute from further consideration.

Using classification, we can analyze the significance of every attribute.

Definition 7.1. Let $\mathcal{I} = \langle U, A\rangle$ be an information system. Let X be a non-empty subset of A: $\emptyset \subset X \subseteq A$. Given an attribute $x \in X$, we say that x is *significant* in X if $\theta_X \subset \theta_{X-\{x\}}$; and that x is *not significant* or *nonsignificant* in X if $\theta_X = \theta_{X-\{x\}}$.

That is, $x \in X$ is significant in X if and only if $X \not\leftrightarrow X - \{x\}$; $x \in X$ is not significant in X if and only if $X \leftrightarrow X - \{x\}$.

We can introduce a quantitative measure for significance as follows.

Definition 7.2. Let X be a non-empty subset of A: $\emptyset \subset X \subseteq A$. Given an attribute $x \in X$, we define the *significance* of x in X as

$$sig_{X-\{x\}}(x) = \frac{|U/\theta_X| - |U/\theta_{X-\{x\}}|}{|U|}.$$

In the special case where X is a singleton, $X = \{x\}$, we also denote $sig_{\emptyset}(x)$ by $sig(x)$:

$$sig(x) = sig_{\emptyset}(x) = \frac{|U/\theta_x| - |U/\delta|}{|U|} = \frac{|U/\theta_x| - |\{U\}|}{|U|} = \frac{|U/\theta_x| - 1}{|U|}.$$

So we always have $sig(x) > 0$ unless $\theta_x = \delta$.

To compute a significance $sig_{X-\{x\}}(x)$, the following computations are required.

(1) Compute $|X|$ partitions U/θ_x for all $x \in X$. The time complexity for computing each partition is $O(|U|^2)$. So the time complexity for computing $|X|$ partitions is $O(|X| \times |U|^2)$.

(2) To compute U/θ_X and $U/\theta_{X-\{x\}}$, $|X| - 1$ and $|X| - 2$ intersections are needed. The time complexity for computing an intersection is $O(|U|^2)$. So the time complexity for computing these intersections is $(|X| - 1 + |X| - 2) \times O(|U|^2) = O(|X| \times |U|^2)$.

Summarizing, the time complexity for computing a significance is $O(|X| \times |U|^2)$.

We know the following:

(1) $0 \leqslant sig_{X-\{x\}}(x) \leqslant 1 - 1/|U|$.

(2) attribute x is *significant* in X if and only if $sig_{X-\{x\}}(x) > 0$.

Example 7.1. For the skull information system in Example 2.1, we have $A = \{y_1, y_2, y_3, y_4\}$ (see Example 3.1), where $y_1 = x_1$, $y_2 = x_2$, $y_3 = x_3$, $y_4 = x_5$. Also:

(1) y_1 is not significant in $\{y_1, y_2, y_3, y_4\}$ since $\theta_{\{y_1,y_2,y_3,y_4\}} = \theta_{\{y_2,y_3,y_4\}} = \varepsilon$ and $sig_{\{y_2,y_3,y_4\}}(y_1) = 0$.

(2) y_2 is not significant in $\{y_1, y_2, y_3, y_4\}$ since $\theta_{\{y_1,y_2,y_3,y_4\}} = \theta_{\{y_1,y_3,y_4\}} = \varepsilon$ and $sig_{\{y_1,y_3,y_4\}}(y_2) = 0$.

(3) y_3 is not significant in $\{y_1, y_2, y_3, y_4\}$ since $\theta_{\{y_1,y_2,y_3,y_4\}} = \theta_{\{y_1,y_2,y_4\}} = \varepsilon$ and $sig_{\{y_1,y_2,y_4\}}(y_3) = 0$.

(4) y_4 is not significant in $\{y_1, y_2, y_3, y_4\}$ since $\theta_{\{y_1,y_2,y_3,y_4\}} = \theta_{\{y_1,y_2,y_3\}} = \varepsilon$ and $sig_{\{y_1,y_2,y_3\}}(y_4) = 0$.

Example 7.2. Let $X = \{x\}$, a singleton in 2^A. Notice that $\theta_X = \bigcap_{x\in X} \theta_x = \theta_x, \theta_{X-\{x\}} = \theta_\emptyset = \bigcap_{\theta\in\emptyset} \theta = \delta$. So

(1) x is significant in X if $\theta_x \neq \delta$.
Indeed, $sig_{X-\{x\}}(x) = sig_\emptyset(x) > 0$ if $\theta_x \neq \delta$.

(2) x is not significant in X if $\theta_x = \delta$.
Indeed, $sig_{X-\{x\}}(x) = sig_\emptyset(x) = 0$ if $\theta_x = \delta$.

Definition 7.3. Let X be a non-empty subset of A: $\emptyset \subset X \subseteq A$. The set of all attributes $x \in X$ which are *significant* in X is called the *core* of X, denoted by C_X. That is, $C_X = \{x \in X \mid sig_{X-\{x\}}(x) > 0\}$.

Also, we define $C_\emptyset = \emptyset$ (see Fig. 8).

Example 7.3. For the skull information system in Example 2.1 (see Example 7.1): $C_{\{y_1,y_2,y_3,y_4\}} = \emptyset$ since y_1, y_2, y_3, y_4 are not significant in $\{y_1, y_2, y_3, y_4\}$.

Example 7.4. Let $X = \{x\}$, a singleton in 2^A. From Example 7.2, we have the following.

(1) $C_{\{x\}} = \{x\}$ if $\theta_x \neq \delta$ since x is significant in X.

(2) $C_{\{x\}} = \emptyset$ if $\theta_x = \delta$ since x is not significant in X.

Knowing what the core is can be useful for efficiency. If one of the attributes had been significant in A, it would have been the focus of our attention in rough analysis. Here we present an algorithm for computing the core.

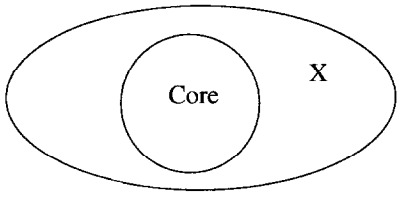

Fig. 8. The core for X.

Algorithm C. Let $\langle U, A\rangle$ be an information system, where $U = \{u_1, u_2, \ldots, u_{|U|}\}$ is the universe, and A is the set of attributes. Let X be a non-empty subset of A: $\emptyset \subset X \subseteq A$. This algorithm computes the core C_X of X.

Let $X = \{x_1, x_2, \ldots, x_{|X|}\}$. The notation $C_X \ni x$ represents the fact that x is included in C_X (i.e., x is significant in X).

(C1) [*Initialize*]. Set $i \leftarrow 1$.

(C2) [$sig_{X-\{x_i\}}(x_i) > 0$?].
If $sig_{X-\{x_i\}}(x_i) > 0$ go to (C3).
If $sig_{X-\{x_i\}}(x_i) = 0$ go to (C4).

(C3) [*Set* $C_X \ni x_i$]. Set $C_X \ni x_i$ go to (C4).

(C4) [$i = |X|$?]. If $i = |X|$ then the algorithm is finished and C_X is the core of X.
If $i < |X|$ then go to (C5).

(C5) [*Increase* i]. Set $i \leftarrow i + 1$. Go to (C2).

This algorithm computes $|X|$ significancies $sig_{X-\{x_i\}}(x_i)$ for $i = 1, 2, \ldots, |X|$. The time complexity for computing one significance is $O(|X||U|^2)$. So the time complexity of Algorithm C is $O(|X|^2|U|^2)$.

Example 7.5. Using Algorithm C, for the skull information system we find that

$$C_{\{y_1, y_2, y_3, y_4\}} = \emptyset.$$

A subset X of A may have many keys X_0. Let us denote them by $X_{01}, X_{02}, \ldots, X_{0s}$; $s \geqslant 1$.

Now we prove that the intersection of all keys is equal to the core.

Theorem 7.1 (Pawlak [6]). *Let* $\langle U, A\rangle$ *be an information system. Let* $X \subseteq A$. *Then* $C_X = \bigcap_{i=1}^{s} X_{0i}$, *where* $X_{01}, X_{02}, \ldots, X_{0i}, \ldots, X_{0s}$ *are all keys of* X (*see Fig.* 9).

Proof.

(i) $C_X \subseteq \bigcap_{i=1}^{s} X_{0i}$. Suppose that $x \in C_X$. We want to prove that $x \in \bigcap_{i=1}^{s} X_{0i}$. Assume that $x \notin X_{0i}$ for some i. Then $X_{0i} \subseteq X - \{x\} \subset X$ and so $\theta_{X_{0i}} \supseteq \theta_{X-\{x\}} \supseteq \theta_X$.

On the other hand, $\theta_X = \theta_{X_{0i}}$ since X_{0i} is a key of X. Thus, we find that $\theta_{X-\{x\}} = \theta_X$; i.e., x is nonsignificant in X; i.e., $x \notin C_X$. This is a contradiction.

Summarizing, if $x \in C_X$ then $x \in X_{0i}$ for all $i = 1, 2, \ldots, s$; i.e., $x \in \bigcap_{i=1}^{s} X_{0i}$.

(ii) $C_X \supseteq \bigcap_{i=1}^{s} X_{0i}$. Suppose that $x \in \bigcap_{i=1}^{s} X_{0i}$. We want to prove that $x \in C_X$. Assume that $x \notin C_X$, where $x \in X$. Then x is nonsignificant in X and we have $\theta_X = \theta_{X-\{x\}}$. On the other hand, since keys always exist, we know that $X - \{x\}$ has a key X_0. From $X_0 \subseteq X - \{x\}$ we know that $x \notin X_0$. Also, key X_0 of $X - \{x\}$ satisfies:

(1) $\theta_{X_0} = \theta_{X-\{x\}}$;

(2) if $X' \subset X_0$ then $\theta_{X-\{x\}} \subset \theta_{X'}$.

These two conditions can be rewritten as:

(1) $\theta_{X_0} = \theta_X$;

(2) if $X' \subset X_0$ then $\theta_X \subset \theta_{X'}$.

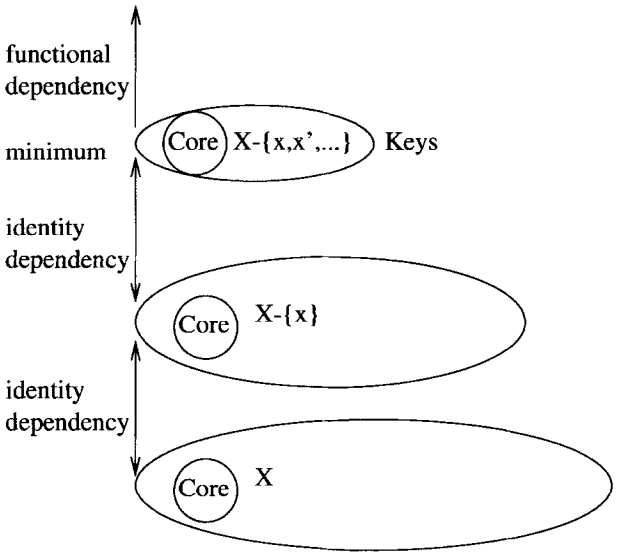

Fig. 9. Keys and core.

Noting that $X_0 \subseteq X - \{x\} \subset X$, we know that X_0 is a key of X as well. So $X_0 = X_{0i}$ for some i, and $x \notin X_0$. Then, from $x \notin X_{0i}$ for some i we have $x \notin \bigcap_{i=1}^{s} X_{0i}$. This is a contradiction. Summarizing, if $x \in \bigcap_{i=1}^{s} X_{0i}$ then $x \in C_X$. □

Example 7.6. For the skull information system (see Example 6.1), the set $A = \{y_1, y_2, y_3, y_4\}$ of all attributes has two keys $\{y_4\}, \{y_1, y_3\}$. So it has core $C_A = \{y_4\} \cap \{y_1, y_3\} = \emptyset$.

8. Significant subsets of attributes

The time complexity for finding keys can be reduced by analysing the *significance* of attributes.

Let $\langle U, A\rangle$ be an information system, where U is the universe, and A is the set of attributes. A subset X ($\subseteq A$) is significant if its every attribute is significant.

Definition 8.1. Let X be a non-empty subset of A: $\emptyset \subset X \subseteq A$. The non-empty subset X is said to be *significant* or *independent* [6] if each $x \in X$ is significant in X; otherwise X is *nonsignificant*.

An empty set $\emptyset$ is said to be *significant*.

Thus, an attribute set X is significant if and only if X is equal to its core: $X = C_X$.

Let $X \subseteq A$ be a (non-empty or empty) subset of A. It is easy to verify the following assertions.

(1) If X is nonsignificant, then every superset $X \cup \{x_1, x_2, \ldots, x_l\}$ of X is nonsignificant, where $x_1, x_2, \ldots, x_l \in A - X$.

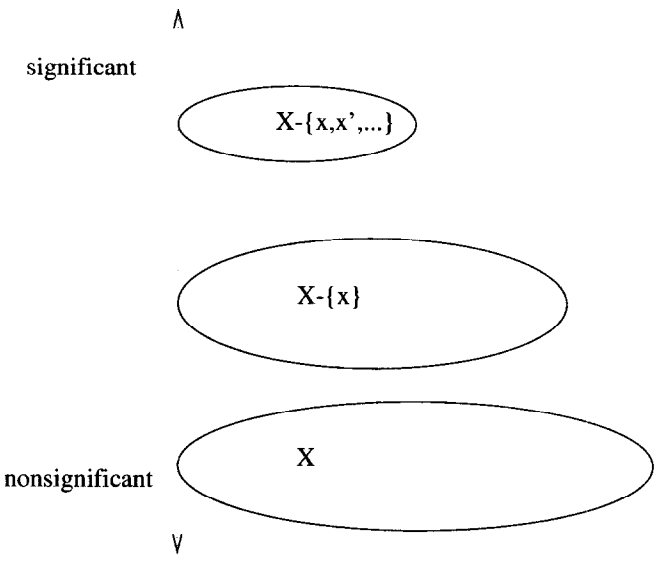

Fig. 10. Significant subsets of attributes.

(2) If X is significant, then every subset $X - \{x_1, x_2, \ldots, x_s\}$ of X is significant, where $x_1, x_2, \ldots, x_s \in X$ (see Fig. 10).

Example 8.1. For the skull information system (see Examples 2.1 and 3.1), significance can be calculated for subsets of $A = \{y_1, y_2, y_3, y_4\}$ as follows.

(1) $\emptyset$ is significant.

(2) All singletons $\{y_1\}, \{y_2\}, \{y_3\}, \{y_4\}$ are significant since $\theta_{y_1}, \theta_{y_2}, \theta_{y_3}, \theta_{y_4}$ are different from δ.

(3) $A = \{y_1, y_2, y_3, y_4\}$ is nonsignificant since y_1, y_2, y_3, y_4 are not significant in A. Now, notice that $|U/\theta_{y_1}| = |\{V_{11}, V_{12}\}| = 2$, where

$$V_{11} = \{\#45, \#163, \#181\}, \quad V_{12} = \{\#92, \#167\}.$$

$|U/\theta_{y_2}| = |\{V_{21}, V_{22}\}| = 2$, where

$$V_{21} = \{\#45, \#163, \#167\}, \quad V_{22} = \{\#92, \#181\}.$$

$|U/\theta_{y_3}| = |\{V_{31}, V_{32}, V_{33}\}| = 3$, where

$$V_{31} = \{\#45, \#167\}, \quad V_{32} = \{\#92, \#181\}, \quad V_{33} = \{\#163\}.$$

$|U/\theta_{\{y_4\}}| = |\{V_{51}, V_{52}, V_{53}, V_{54}, V_{55}\}| = 5$, where

$$V_{51} = \{\#45\}, \quad V_{52} = \{\#92\},$$
$$V_{53} = \{\#163\}, \quad V_{54} = \{\#167\}, \quad V_{55} = \{\#181\}.$$

$$|U/\theta_{\{y_1,y_2\}}| = |\{V_{51} \cup V_{53}, V_{52}, V_{54}, V_{55}\}| = 4.$$

$$|U/\theta_{\{y_1,y_3\}}| = |\{V_{51}, V_{52}, V_{53}, V_{54}, V_{55}\}| = 5.$$

$$|U/\theta_{\{y_2,y_3\}}| = |\{V_{51} \cup V_{54}, V_{52} \cup V_{55}, V_{53}\}| = 3.$$

(4) $\{y_1, y_2\}$ is significant since y_1, y_2 are significant in $\{y_1, y_2\}$: $sig_{\{y_2\}}(y_1) = 2/5 > 0$; $sig_{\{y_1\}}(y_2) = 2/5 > 0$.

(5) $\{y_2, y_3\}$ is nonsignificant since y_2 is not significant in $\{y_2, y_3\}$ even if y_3 is significant in $\{y_2, y_3\}$: $sig_{\{y_3\}}(y_2) = 0$; $sig_{\{y_2\}}(y_3) = 1/5 > 0$.
So its supersets $\{y_1, y_2, y_3\}, \{y_2, y_3, y_4\}$ are nonsignificant.

(6) $\{y_3, y_4\}$ is nonsignificant since y_3 is not significant in $\{y_3, y_4\}$ even if y_4 is significant in $\{y_3, y_4\}$: $sig_{\{y_4\}}(y_3) = 0$; $sig_{\{y_3\}}(y_4) = 2/5 > 0$.
Again, its supersets $\{y_2, y_3, y_4\}, \{y_1, y_3, y_4\}$ are nonsignificant.
Similarly, $\{y_1, y_4\}$ is nonsignificant since y_1 is not significant in $\{y_1, y_4\}$ even if y_4 is significant in $\{y_1, y_4\}$: $sig_{\{y_4\}}(y_1) = 0$; $sig_{\{y_1\}}(y_4) = 3/5 > 0$.
Also, its supersets $\{y_1, y_2, y_4\}, \{y_1, y_3, y_4\}$ are nonsignificant.
Also, $\{y_2, y_4\}$ is nonsignificant since y_2 is not significant in $\{y_2, y_4\}$ even if y_4 is significant in $\{y_2, y_4\}$: $sig_{\{y_4\}}(y_2) = 0$; $sig_{\{y_2\}}(y_4) = 3/5 > 0$.
Its supersets $\{y_2, y_3, y_4\}, \{y_1, y_2, y_4\}$ are nonsignificant.

(7) $\{y_1, y_3\}$ is significant since y_1, y_3 are significant in $\{y_1, y_3\}$: $sig_{\{y_3\}}(y_1) = 2/5 > 0$; $sig_{\{y_1\}}(y_3) = 3/5 > 0$.

Example 8.2. Let $X = \{x\}$, a singleton in 2^A. We have the following.
Case A. $\theta_x \neq \delta$. X is significant since x is significant in X.
Case B. $\theta_x = \delta$. X is nonsignificant since x is not significant.

9. Finding one key

According to the concept of significance, we have the following fact about keys. It suggests finding the *significant* ID subsets in order to find the *minimal* ID subsets (see Fig. 11).

First of all, it is easy to show the following.

Let X be a (non-empty or empty) subset of A: $X \subseteq A$. A subset X_0 of X is a *key* of X if and only if X_0 satisfies:

(1) ID: $\theta_{X_0} = \theta_X$;

(2′) significant: X_0 is significant.

Thus, from Theorem 7.1, we know that core C_X for every $X \subseteq A$ is significant. Also,

(1) If $X_0 \subseteq X$ is a key of X then X_0 is significant.

(2) If X is significant then X is the unique key of X.

Example 9.1. For the skull information system (see Example 8.1), the keys of $A = \{y_1, y_2, y_3, y_4\}$ are as follows.

(1) $\emptyset$ is significant. It is not a key of A since $\theta_A = \varepsilon$ and so $\theta_A \neq \theta_\emptyset = \delta$.

(2) All singletons $\{y_1\}, \{y_2\}, \{y_3\}, \{y_4\}$ are significant. Among them $\{y_1\}, \{y_2\}, \{y_3\}$ are not a key of A since $\theta_{y_1}, \theta_{y_2}, \theta_{y_3} \neq \theta_A = \varepsilon$, but $\{y_4\}$ is a key of A since $\theta_{y_4} = \varepsilon$.

(3) $A = \{y_1, y_2, y_3, y_4\}$ is not a key of A since it is nonsignificant.

(4) $\{y_1, y_2\}$ is significant but it is not a key of A since $\theta_{\{y_1,y_2\}} \neq \varepsilon$.

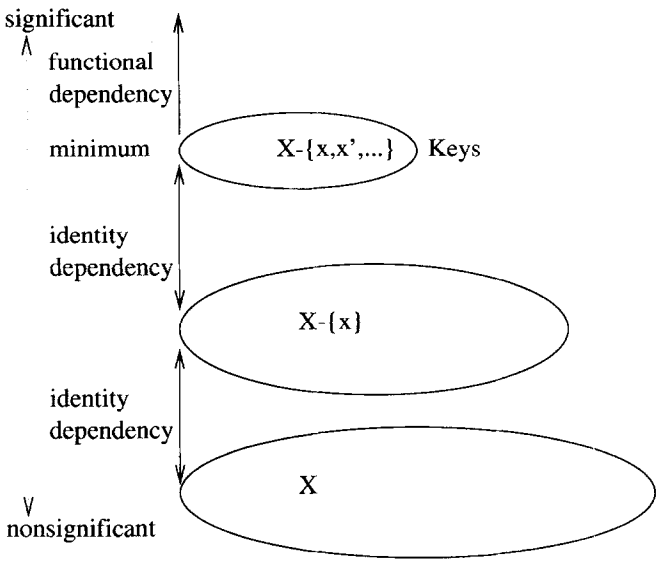

Fig. 11. Keys: significant ID subsets.

(5) $\{y_2, y_3\}$ is nonsignificant. So it is not a key of A. Its supersets $\{y_1, y_2, y_3\}$, $\{y_2, y_3, y_4\}$ are nonsignificant, and so they not keys.
(6) $\{y_3, y_4\}$ is nonsignificant. It is not a key of A. Its supersets $\{y_2, y_3, y_4\}$, $\{y_1, y_3, y_4\}$ are nonsignificant, and so they not keys.
Similarly, $\{y_1, y_4\}$ is nonsignificant—it is not a key of A, and supersets $\{y_1, y_2, y_4\}$, $\{y_1, y_3, y_4\}$ are nonsignificant, so they not keys.
Also, $\{y_2, y_4\}$ is nonsignificant—it is not a key of A, and supersets $\{y_2, y_3, y_4\}$, $\{y_1, y_2, y_4\}$ are nonsignificant, so they not keys.
(7) $\{y_1, y_3\}$ is significant and is a key of A since $\theta_{\{y_1,y_3\}} = \varepsilon$.

Example 9.2. Let $X = \{x\}$, a singleton in 2^A.
Case A. $\theta_x \neq \delta$. $X_0 = X$ is the unique key of X since
(1) $\theta_{X_0} = \theta_X$;
(2) if $X' \subset X_0$ then $X' = \emptyset$ and $\theta_X = \theta_x, \subset \theta_{X'} = \theta_\emptyset = \delta$.
Notice that $X_0 = \{x\}$ $(\theta_x \neq \delta)$ is significant.
Case B. $\theta_x = \delta$. $X_0 = \emptyset$ is the unique key of X since
(1) $\theta_{X_0} = \theta_\emptyset = \delta, = \theta_X = \theta_x = \delta$;
(2) there are no $X' \subset X_0$ since $X_0 = \emptyset$.
Notice that $X_0 = \emptyset$ is significant.

We can find **one** key X_0 for every (non-empty or empty) subset $X \subseteq A$ as follows. Use mathematical induction on $|X|$. We can find one key when $|X| = 0, 1$. Suppose that we can find one key for $|X| - 1$. We want to find one key for subset X. There are two cases depending on whether X is significant or not.
Case A. X is significant. Then X is the unique key of X.

Case B. X is nonsignificant. In this case, there exists a nonsignificant $x_1 \in X$: $\theta_X = \theta_{X-\{x_1\}}$ for this $x_1 \in X$. Let $X_1 = X - \{x_1\}$. Then $\theta_X = \theta_{X_1}$ and $|X_1| = |X| - 1$. So we can find one key for X_1 by the induction hypothesis. That is, X_1 has a key X_0 such that:

(1) $\theta_{X_0} = \theta_{X_1}$;

(2) X_0 is significant.

Notice that $\theta_X = \theta_{X_1}$. So,

(1) $\theta_{X_0} = \theta_X$;

(2) X_0 is significant.

That is, X has also the key X_0. The mathematical induction is completed. An algorithm to find **one** key is the following, accordingly.

Algorithm A. Let $\langle U, A\rangle$ be an information system. Let $X = \{x_1, x_2, \ldots, x_j, \ldots, x_{|X|}\}$ be a subset of A. This algorithm finds one key of X.

Step 1. Compute U/θ_{x_j} and $sig_{X-\{x_j\}}(x_j)$ for $j = 1, 2, \ldots, |X|$. Choose x_{j_1} such that

$$sig_{X-\{x_{j_1}\}}(x_{j_1}) = \min_{j=1,2,\ldots,|X|} \big(sig_{X-\{x_j\}}(x_j)\big).$$

If $sig_{X-\{x_{j_1}\}}(x_{j_1}) > 0$ then X is **one key** of X and the algorithm is completed. Otherwise, go to Step 2.

Step 2. Compute $sig_{X-\{x_{j_1},x_j\}}(x_j)$ for $j = 1, 2, \ldots, |X|$; $j \neq j_1$. Choose x_{j_2} such that

$$sig_{X-\{x_{j_1},x_{j_2}\}}(x_{j_2}) = \min_{j=1,2,\ldots,|X|;\, j\neq j_1} \big(sig_{X-\{x_{j_1},x_j\}}(x_j)\big).$$

If $sig_{X-\{x_{j_1},x_{j_2}\}}(x_{j_2}) > 0$ then $X - \{x_{j_1}\}$ is **one key** of X and the algorithm is completed. Otherwise, go to Step 3.

Step 3. Compute $sig_{X-\{x_{j_1},x_{j_2},x_j\}}(x_j)$ for $j = 1, 2, \ldots, |X|$; $j \neq j_1, j_2$. Choose x_{j_3} such that

$$sig_{X-\{x_{j_1},x_{j_2},x_{j_3}\}}(x_{j_3}) = \min_{j=1,2,\ldots,|X|;\, j\neq j_1,j_2} \big(sig_{X-\{x_{j_1},x_{j_2},x_j\}}(x_j)\big).$$

If $sig_{X-\{x_{j_1},x_{j_2},x_{j_3}\}}(x_{j_3}) > 0$ then $X - \{x_{j_1}, x_{j_2}\}$ is **one key** of X and the algorithm is completed. Otherwise, go to Step 4.

And so on.

Step $|X|$. Compute $sig_{X-\{x_{j_1},x_{j_2},\ldots,x_{j_{|X|-1}},x_j\}}(x_j) = sig(x_j)$ for $j = 1, 2, \ldots, |X|$; $j \neq j_1, j_2, \ldots, j_{|X|-1}$.

If $sig(x_j) > 0$ then $\{x_j\}$ is **one key** of X and the algorithm is completed.

If $sig(x_j) = 0$ then the empty set $\emptyset$ is **one key** of X and the algorithm is completed.

By using this algorithm, the time complexity to find **one** key is polynomial.

At the first step, we check if X is significant; i.e., if every $x \in X$ is significant in X: $\theta_X \subset \theta_{X-\{x\}}$? We need to compute $|X|$ significancies $sig_{X-\{x_j\}}(x_j)$ for $j = 1, 2, \ldots, |X|$. The time complexity to compute one significance is $O(|X||U|^2)$. So the price of the first step is $|X| \times O(|X||U|^2)$.

At Step 2, we need to compute $|X| - 1$ significancies $sig_{X-\{x_{j_1},x_j\}}(x_j)$. The time complexity to compute one significance is $O(|X||U|^2)$. So the price of Step 2 is $(|X| - 1) \times O(|X||U|^2)$.

At Step 3, we need to compute $|X|-2$ significancies $sig_{X-\{x_{j_1},x_{j_2},x_j\}}(x_j)$. So the price of Step 3 is $(|X|-2)\times O(|X||U|^2)$.

And so on.

At Step $|X|$, we need to compute 1 significance $sig(x_j)$. So the price of step $|X|$ is $1\times O(|X||U|^2)$.

So the total price is

$$(|X|+(|X|-1)+(|X|-2)+\cdots+1)\times O(|X||U|^2)$$
$$=\frac{|X|(|X|+1)}{2}\times O(|X||U|^2)=O(|X|^3|U|^2).$$

Notice that Algorithm A can be run in parallel mode to compute all keys concurrently. The following is an example.

Example 9.3. For the skull information system, from Example 8.2, keys for $A=\{y_1,y_2,y_3,y_4\}$ can be found.

Step 1. A is nonsignificant, and

$$sig_{\{y_1,y_2,y_3\}}(y_4)=sig_{\{y_2,y_3,y_4\}}(y_1)=sig_{\{y_1,y_3,y_4\}}(y_2)=sig_{\{y_1,y_2,y_4\}}(y_3)=0.$$

Step 2A. Choose $y_{j_1}=y_1$,

$$sig_{\{y_2,y_3\}}(y_4)=2/5,\quad sig_{\{y_3,y_4\}}(y_2)=0,\quad sig_{\{y_2,y_4\}}(y_3)=0.$$

Step 3AA. Choose $y_{j_1}=y_1, y_{j_2}=y_2$,

$$sig_{\{y_3\}}(y_4)=2/5,\quad sig_{\{y_4\}}(y_3)=0.$$

Step 4AA. Choose $y_{j_1}=y_1, y_{j_2}=y_2, y_{j_3}=y_3$,

$$sig(y_4)=4/5>0.$$

So $\{y_4\}$ is **one key**. Go to Step 3AB.

Step 3AB. Choose $y_{j_1}=y_1, y_{j_2}=y_3$,

$$sig_{\{y_2\}}(y_4)=2/5,\quad sig_{\{y_4\}}(y_2)=0.$$

So $y_{j_3}=y_2$, and $\{y_{j_1},y_{j_2},y_{j_3}\}=\{y_1,y_2,y_3\}$ is the same as that in Step 4AA. Go to Step 2B.

Step 2B. Choose $y_{j_1}=y_2$,

$$sig_{\{y_1,y_3\}}(y_4)=0,\quad sig_{\{y_3,y_4\}}(y_1)=0,\quad sig_{\{y_1,y_4\}}(y_3)=0.$$

Step 3BA. Choose $y_{j_1}=y_2, y_{j_2}=y_4$,

$$sig_{\{y_1\}}(y_3)=3/5>0,\quad sig_{\{y_3\}}(y_1)=2/5>0.$$

So $\{y_1,y_3\}$ is **one key**. Go to Step 2C.

Step 2C. Choose $y_{j_1}=y_3$,

$$sig_{\{y_1,y_2\}}(y_4)=1/5>0,\quad sig_{\{y_1,y_4\}}(y_2)=0,\quad sig_{\{y_2,y_4\}}(y_1)=0.$$

Step 3CA. Choose $y_{j_1}=y_3, y_{j_2}=y_2$,

$$sig_{\{y_1\}}(y_4)=3/5>0,\quad sig_{\{y_4\}}(y_1)=0.$$

So $y_{j_3} = y_1$, and $\{y_{j_1}, y_{j_2}, y_{j_3}\} = \{y_1, y_2, y_3\}$ is the same as that in Step 4AA. Go to Step 2D.

Step 2D. Choose $y_{j_1} = y_4$,

$$sig_{\{y_2,y_3\}}(y_1) = 2/5 > 0, \quad sig_{\{y_1,y_3\}}(y_2) = 0, \quad sig_{\{y_1,y_2\}}(y_3) = 1/5 > 0.$$

So $y_{j_2} = y_2$, and $\{y_{j_1}, y_{j_2}\} = \{y_2, y_4\}$ is the same as that in Step 3BA.

The algorithm is completed and the keys $\{y_4\}$ and $\{y_1, y_3\}$ are output. So either the skull size or the combination of teeth size and location can be used to identify specimens.

10. Conclusions

In this paper we have presented an investigation of computational methods for rough analysis. We suggest a series of algorithms for use in such analysis of databases as a step in the discovery of knowledge in the form of classification. In particular, we suggest the use of a significance measure to obtain dependencies and keys in data collections. The use of the significance measure allows us to reduce the computational cost of discovery by the rough analysis method. The use of these algorithms might produce evidence and insights into states of affairs in the world being modelled that are not readily available in other ways. A running example of a paleontological investigation has been used throughout, and the keys and dependencies that are discovered from the corresponding data collection can be used as evidence to choose between hypotheses in that study.

Acknowledgement

The authors wish to thank the anonymous reviewers for their constructive comments. Also, we would like to thank D. Ruan for useful comments on this study.

References

[1] D.A. Bell, From data properties to evidence, IEEE Transactions on Knowledge and Data Engineering 5 (6) (1993) 965–968.

[2] D.A. Bell, J.W. Guan, Computational methods for rough classification and discovery, J. Amer. Soc. Inform. Sci., Special Topic Issue on Data Mining (to appear).

[3] P.R. Cohen, Heuristic Reasoning about Uncertainty: An Artificial Intelligent Approach, Pitman Advanced Publishing Program, 1985.

[4] J.W. Grzymala-Busse, Managing Uncertainty in Expert Systems, Kluwer Academic, Dordrecht, 1991.

[5] T.Y. Lin, N. Cercone, Rough Sets and Data Mining: Analysis of Imprecise Data, Kluwer Academic, Dordrecht, 1996.

[6] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning about Data, Kluwer, Dordrecht, Netherlands, 1991.

[7] Z. Pawlak, J.W. Grzymala-Busse, R. Slowinski, W. Ziarko, Rough sets, Comm. ACM 38 (11) (1995) 89–95.

[8] Z. Pawlak, C. Rauszer, Dependency of attributes in information systems, Bull. Polish Acad. Sci. Math. 33 (1985) 551–559.

[9] G. Piatetsky-Shapiro, W.J. Frawley (Eds.), Knowledge Discovery in Databases, AAAI/MIT Press, Cambridge, MA, 1991.

[10] R. Ruan (Ed.), Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks and Genetic Algorithms, Kluwer Academic, Dordrecht, 1997.
[11] R.R. Stoll, Sets, Logic and Axiomatic Theories, W.H. Freeman, San Francisco, CA, 1961.
[12] J. Ullman, Principles of Database Systems, Prentice-Hall, Englewood Cliffs, NJ, 1986.
[13] W. Ziarko, The discovery, analysis, and representation of data dependencies in databases, in: G. Piatetsky-Shapiro, W.J. Frawley (Eds.), Knowledge Discovery in Databases, AAAI Press/MIT Press, Cambridge, MA, 1991, pp. 177–195.
[14] A. Walker, R.E.F. Leakey, The hominids of East Turkana, Scientific American (August 1978) 54–66.