1-1-2002

Image indexing and retrieval using formal concept analysis.

Taek-Sueng Jang
University of Windsor

Image Indexing and Retrieval Using Formal Concept Analysis

By

Taek-Sueng Jang

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada
2002

0-612-75793-5

962039

APPROVED BY

[illegible]

Dr. S. Suh, External Reader
Department of Economics

[illegible]

Dr. B. Boufama, Internal Reader
Computer Science

[illegible]

Dr. I. Ahmad, Advisor
Computer Science

[illegible]

Dr. A. Ngom, Chairman
Computer Science

ABSTRACT

With the advent of digital photography and advancement in digitization process, everyday a great number of digital images are produced, resulting in a rapid growth in the size of image databases. Despite advances in image data capture and storage techniques, development of methods for effective image retrieval has not kept pace with the technology of image production. The ability to effectively retrieve non-alphanumeric data is a complex issue. Research and development in recent years have focused on the retrieval of images by their content. In this thesis, based on Formal Concept Analysis (FCA), a new image indexing and retrieval technique is proposed. This technique allows us fast retrieval of image from the databases. The retrieval efficiency in this scheme depends on the number of attributes rather than the number of images in the database with dynamic support for addition of new images but requires an advanced knowledge of a specific domain.

DEDICATION

I dedicate this thesis to my parents, who live in Korea and have supported me without stint, and my lovely wife, Seo-Young, who has always encouraged me and cheered me up whenever I was depressed. In my life, I never thought I would have my own thesis, but now I have one. This great achievement in my life was possible because of my parents and wife.

I also dedicate this to my future children because they couldn't come to this beautiful world earlier while I was doing Master Program.

내 인생의 첫 논문을

나의 사랑하는

부모님.

그리고. 부인 서영에게 바칩니다.

ACKNOWLEDGEMENTS

My foremost acknowledgement goes to Dr. Imran Ahmad who is a professor in University of Windsor and my supervisor. His great help – endless reading and correcting, remarkably improved this thesis with his great knowledge.

My second acknowledgement goes to Dr. Y.G. Park who was my former supervisor and is presently a professor at Bradley University. Without his acceptance to University of Windsor, I couldn't have had this opportunity. I would also like to thank him for giving me the basic idea for this thesis.

A special acknowledgement goes to Dr. Boufama, Dr. Suh and Dr. Ngom for their kind advice for this thesis work.

I would also like to acknowledge my classmates who spent time with me during my thesis work : Mr. J.S. Kim who inspired me to study hard, Mr. B.S. Kim who helped me to improve my knowledge and Mr. H.S. Ahn who helped me relax when it was needed. I'm also grateful to C.H. Park and S.H. Mun. Mr. Park always encouraged me and gave me kind words, and Ms. Mun shared her knowledge to finish my thesis.

My final acknowledgement goes to my parents and wife who helped me in various ways and some of my friends who joined during my smoking time. In fact, all the ideas in this thesis came from these smoking times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

With the advancements in digital photography and digitization process, everyday a great number of digital images are created or produced. As a result, the size of digital image repositories is growing at a very fast pace, creating a demand for efficient techniques for effective management and organization of images and a mechanism to navigate through such repositories. Moreover, this demand has further increased by the epochal growth of the World Wide Web (WWW). Users in many fields are exploiting the advantages offered by such collections in all kinds of new and exciting ways in a number of different applications found in the areas of geographical and medical information systems, digital photo albums, sports and training, news, advertisement and multimedia applications to name a few. At the same time, such users are also discovering that the process of locating a desired image in a large and varied collection can be a source of considerable frustration. The problems of image retrieval are widely recognized and the search for solutions is an increasingly active area of research.

1.1. Image Retrieval

An Image Database (IDB) is a collection of unique images. Each image in the database represents specific individual or group of objects in the real world. There are two main approaches to address the issues of image retrieval from such databases. A more traditional approach, known as text-based retrieval approach, depends on textual descriptions of visual attributes of the image contents in the form of keyword or annotations. There are processes to enter these descriptions into the database along with the actual image and require a priori knowledge of the type of application domain and queries that can be addressed to the database. Kodak Picture Exchange System (KPX) [29], PressLink [35] and Time pictures archive collection (Time) [3] are few of the systems based on this approach. The second and most widely discussed approach depends on automatic feature extraction of visual and mathematical attributes of images such as color, texture, shape of objects, etc. and is used in some of the popular systems such as QBIC [14, 38], Virage [2], Photobook [45]. However, irrespective of the approach, considerable research is still required to produce a fast and efficient retrieval methods and make the best use of visual data.

Even though both of these approaches make use of some of the visual image properties but the second approach is more commonly referred as the content-based image retrieval. The earliest use of this term in the literature seems to have been by Kato [26] to describe his experiments of automatic retrieval of images from a database by color and shape feature. The term has since been widely used to describe the process of retrieving desired images from a large collection on the basis of various features such as color, texture and shape that can be automatically extracted from the images themselves. Same technique is applied to both the database images during their addition to the database and to the user provided query image to find possible matches.

As mentioned earlier, essentially contents could be either text-based or visual contents. In some systems, when an image is added into IDB along with visual attributes, descriptions of image are also extracted and stored in the IDB, thus, allowing the users to query the database using more than one approach. When IDB receives the user-query, it examines the approach specified in the user provided query, followed by mechanism to establish suitability of match and retrieves only those images which satisfy the criteria given in the user provided query.

A survey of literature indicates the existence of several different types of content-based (text or visual) image retrieval systems [1, 3, 6, 14, 22, 23, 32, 35, 36, 39, 49, 53]. There are several different types of image retrieval supported by the systems and are classified as [1, 19, 20, 39] :

- Retrieval based on textual descriptions and keywords.
- Retrieval based on attributes and existence of image objects.
- Retrieval based on similarity by:
  - color
  - texture
  - shape
  - spatial locations and geographical position
  - template through rough sketch

In some systems, one or more of the above retrieval types are combined to improve the performance of search and retrieval process. Moreover, as mentioned before, content-based queries are often combined with text and keywords to get powerful retrieval methods for image retrieval. Many popular and well known image retrieval systems such as QBIC [12, 14, 38], Virage [2, 21], RetrievalWare [4], Photobook [45, 46], VisualSEEK and WebSEEk [15, 53], Netra [34], and MARS [17, 47] use this approach to provide relatively more comprehensive retrievals.

1.2. Research Issues

Image Retrieval has been a widely studied topic since the 1970's and a number of approaches and techniques have been proposed. Image retrieval is a complex problem and requires expertise in more than one area [47]. Since an image represents a multitude of complex information, none of the retrieval techniques presented so far either address all of the problems or have been able to provide a suitable solution to them.

Because of color histogram efficiency and the fact that they are insensitive to small changes in camera viewpoint, use of color histograms for retrieval of images has been quite popular among many contemporary researchers [40, 43, 54, 55]. However, despite their advantages, color histograms lack spatial information and fail to discriminate objects of the same color but different shape. As a result, images with totally different objects can have similar histograms [43]. Similarly, images with various appearances of the same object can also have different histograms [44]. Shape feature has also been used for image retrieval. However, shapes are more complex than color and need number of parameters for explicit representation while color needs only a few parameters [37]. Moreover, the computation of various shape features and their comparisons are computationally expensive and therefore, unfit for large collection of images.

There are also uncertainty and time complexity problems. Existing image analysis techniques are inadequate to address the complexity issues associated with image contents and result in introduction of different types of uncertainties. Similarly, in text and appearance-based systems, extraction of keywords and description of images is a complicated and difficult task. In such systems, extracting and annotation of keywords is generally a manual process. Since manual descriptions are prone to error and depend on perception of extractor, unexpected results are possible. Second, time to retrieve potentially matching images is generally application and domain dependent and a function of number of images in the database. The phenomenal retrieval time for large collections of images makes many of the proposed systems prohibitive for real-life applications [13]. As mentioned earlier, a great number of digital images are generated everyday, making retrieval efficiency to be a major concern and important parameter beside accuracy and relevance of retrieved information.

1.3. Problem Statement

Although tremendous work has been done on content-based image retrieval, efficient and precise image retrieval still remains an open problem. Many keyword-based text information retrieval systems have achieved great success for indexing image collections on web sites. The two main problems requiring intensive research efforts are the effectiveness of the search, retrieval process and the time complexity to retrieve desired information. In this thesis, the use of Formal Concept Analysis techniques is proposed to catalogue descriptions or keywords associated with the image contents. The time complexity for the proposed approach depends on the number of attributes, which represent real objects in the world rather than the number of images, as is the case in all of the image retrieval systems. A novel addition method is also proposed to build a lattice structure to reduce the time for building a lattice structure, as is done in Formal

Concept Analysis.

Remainder of this thesis is organized as follows. Section 2 contains a review of some of the related work. Section 3 describes methodologies used in this thesis to build a lattice structure using formal concept analysis for image retrieval. Section 4 discusses and analyzes issues of the time complexity for retrieval of images in this system. Section 5 present experimental results and implementation details whereas Section 6 contains the conclusion and directions for future research work.

2. Related Work

This section briefly explains some of the related work. Some of the concepts described in this chapter are used in subsequent chapters to describe this system and experiments. As mentioned in Chapter 1, many popular and well-known image retrieval systems exist. We first briefly introduce the three most popular image retrieval systems followed by a more detailed explanation of the most widely talked about related techniques.

2.1. Photobook

Photobook [45, 46] was developed at MIT Media Lab and is a tool to perform queries on image databases and to retrieve images based on their contents. It works by comparing various image features rather than images themselves. Commonly used features are the image or object shape, texture, and its color. These features in turn serve as parameter values for a particular model fitted to each image and are compared using one of the many matching algorithms such as Euclidean, mahalanobis, divergence, vector space angle, historgram, etc. that Photobook provides. It also provides the capability to perform searches on the basis of a user-defined matching algorithm via dynamic code loading. Photobook also employs an interactive learning agent, FourEyes, which selects and combines models based on examples from the user, thus, allowing users to directly address their intent. The system has been successfully used in a number of applications, involving retrieval of image textures, shapes, and human faces, each using feature based on a different model of the image. Further information about Photobook can be found in [45] and at the URL: http://www-white.media.mit.edu/vismod/demos/photobook.

2.2. QBIC (Query By Image Content)

QBIC [12, 14, 38, 46] is the first commercial image retrieval system. It allows a user to pose queries on large image databases based on visual image content, i.e., color texture, etc. Such queries use the visual properties of images, so that user can match colors, textures and their positions in the image without explicitly describing them in words. However, content-based queries are often combined with text and keyword predicates to get powerful retrieval methods for image and multimedia databases. For color, it uses a K-element color histogram for each object and scene. For texture, the tamura texture representation, combinations of coarseness, contrast and directionality are used. For shape, it consists of shape area, circularity, eccentricity, major axis orientation and a set of algebraic moments invariants. A few systems take into account the high dimensional feature indexing. QBIC is one of the systems. For its indexing subsystem, KLT is used to perform dimension reduction and later R*- tree is used as the multi-dimensional indexing structure. As mentioned above, text-based keyword search can be combined with content-based similarity search in its new system. An online QBIC demo and further information about the system can be found at their web site: http://www.qbic.almaden.ibm.com.

2.3. Virage

Virage [2, 21], developed at Virage Inc, is based on image contents such as color, texture, composition (color layout) and structure (object boundary information) with visual queries. In this respect it is similar to QBIC's visual queries but differs from QBIC because of its supports for arbitrary combinations of the above mentioned four atomic properties (color, texture, composition and structure) in queries. Users can emphasize on any of these atomic features. It is available as a series of independent modules, which system developers can build into their own programs. This makes it easy to extend the system by building in new types of query interface, or additional customized modules to process specialized collections of images such as trademarks. Alternatively, the system is available as an add-on to existing database management systems such as Oracle or Informix. A high-profile application of Virage technology is AltaVista's AV Photo Finder (http://image.altavista.com/cgi-bin/avncgi), allowing Web surfers to search for images by content similarity. Virage technology has also been extended to the management of video data. This allows content owners to efficiently digitize, locate and manage video and distribute it across the Internet or for viewing on any device. Further information about Virage can be found at the web site: http://www.virage.com.

2.4. Text-Based Image Retrieval

Text-based image retrieval has been used in a number of proposed image retrieval systems such as Kodak Picture Exchange System (KPX) [29], PressLink [35] and Time pictures archive collection (Time) [3]. It has the ability to represent general and specific illustration of objects on images. Before images could be digitized, librarians, curators and archivists through text descriptors or classification codes had provided the access to image collections. This manual procedure, though time consuming, costly and suffering from low term agreement across indexers, and between indexers and user queries [18], tends to be more useful and more practical automatic feature-based IRS [33].

Text-based image retrieval is still common practice. Zheng (1999) and Goodrum & Martin (1997) have recently reported on the hybridization of multiple schemas for classifying collections of historic costume collections. Hourihane (1989) has also reviewed a number of unique systems for image classification. Automatic annotation of textual attributes has been guided using captions from still images, and transcripts, close captioning, or verbal description for the blind, that accompany many videos by Turner (1994). These automatic annotation approaches greatly reduce the labor work in manual annotation work. However, there are many images without accompanying text [18] and that should be remembered.

2.5. Two-Dimensional Strings

In this thesis, 1-D String is used for attributes set, and a lattice structure of FCA is used for the data structure to contain concepts represented as an attribute set and an object set. An explanation of how 1-D string of 2-D strings and FCA are adapted to this scheme is described in Chapter 3.

2-D string is a representational structure and its technique is a representation of spatial information of image properties. To derive the 2-D string representation of a given image, first the image is segmented and then the locations of all objects are projected along the x- and y-axis. The location of each object is at center of mass of each object. The two one-dimensional strings are derived forming the 2-D string representation by checking the objects from left to right and from below and above. A 2-D string represents relationships ("left/right" and "below/above") between image objects (a pair of two objects).

S.K. Chang et. al. [8] provides a technique for a simple and compact representation of spatial image properties in the form of two one-dimensional strings. 2-D strings [8] is one of a few representation structures originally designed for use in an IDB environment. 2-D strings can be used to resolve queries based on image contents. For a query, an example image or icon is provided. However, they used an exhaustive search : to retrieve images, all of 2-D string representations corresponding to all stored images are compared with a 2-D string representation of a given user query image or icon. C.C. Chang and S.Y. Lee [5] proposed a technique for the indexing of 2-D strings. In other words, 2-D strings are indexed based on representations corresponding to all pairs of objects and each pair of objects is assigned an index. After that, each pair of objects is saved into a hash table.

Basic notions of 2-D String

Let O be a finite set of symbols representing real objects in the world, given as O = {a, b, c, ...} and let R be a set of symbols representing spatial relationships between two objects, given as R = {<, =, :}. An explanation of symbols "<", "=", and ":" is given in next paragraph. A 1-D string is a string formed by $o_1r_1\ o_2r_2 \ldots o_{i-1}r_{i-1}o_i \ldots\ r_{n-1}o_n$, where $n \geq 0$, $o_i \in O$, $r_i \in R$, $0 \leq i \leq n$. A 2-D string is a (u, v), where

u is related to X-axis

v is related to Y-axis.

A symbol "<" represents "left/right" in u, or "below/above" in v. For example, if an object A is left of an object B, then the relationship is represented as "A<B". If the object

A is below object B, then the relation is also represented as "A<B". A symbol "=" represents the same projection along the X-axis or the Y-axis. For example, if object A and B are at the same position in u, but at different positions in v then the relationship is represented as "A=B". Likewise, if object A and B are at the same position in v but at different positions in u then the relationship is represented as "A=B". A symbol ":" represents the same position between two objects in O. For example, if object A and B are at the same position in both u and v then the relationship is represented as "A:B".

Example of 2-D String

Subsequent paragraphs contain a detailed but simple example of 2-D string. With this example, the process of translation of physical image given in Figure 1-a to a symbolic image Figure 1-b is demonstrated followed by a description of a process to obtain 2-D string from a symbolic image representation.

Let the actual image Figure 1-a contains the following four different objects :

Cup, Pen, Notebook, Pencil.

These objects can be simply represented by symbols "A", "B", "C", and "D" respectively, as shown in Figure 1-b. The symbolic image obtained can be represented by a 2-D string as follows :

$$(u, v) = (A < B < C : D, A = C : D < B)$$

From Figure 1-b, u = (A<B<C:D), which is related to X-axis is obtained and is explained as :

- A is on the left of B, $\therefore$ A < B

- B is on the left C and D, ∴ $B < C : D$
- C is at the same position as D, ∴ $C : D$.

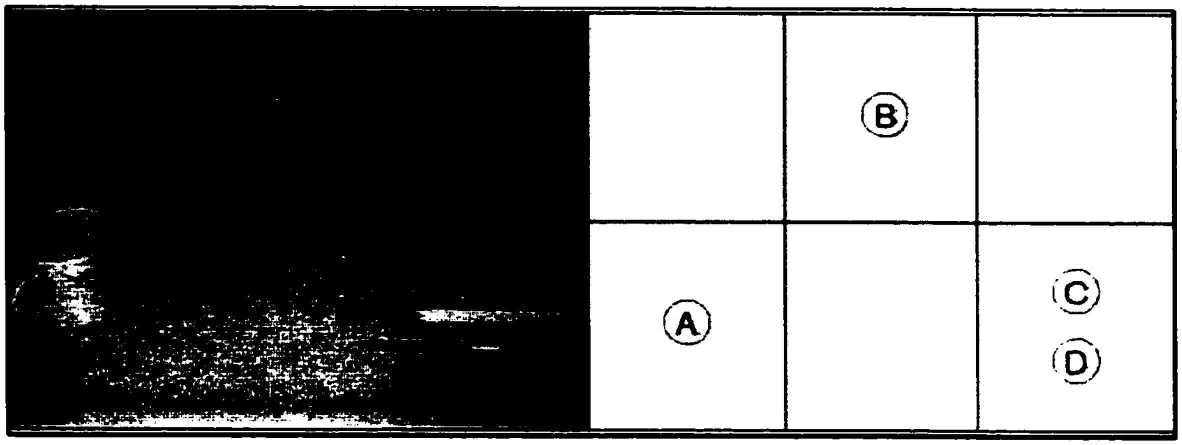


Figure 1. (a) A physical image (left image). (b) A symbolic image (right image)

The v is described as $A = C : D < B$ and is explained as :

- A is at the same projection as C and D in Y-axis, but not in X-axis,

  ∴ $A = C : D$
- C and D are at the same position, ∴ $C : D$
- C and D is below B, ∴ $C : D < B$

2.6. Formal Concept Analysis (FCA)

Formal Concept Analysis [16, 58] proposed by Rudolf Wille in 1982 is based on the mathematical order theory and formalization of the philosophical understanding of a concept. It provides graph-based visualizations of tabular data and has been successfully applied to a number of different fields, such as Text data mining, Social sciences, and

Software engineering [41, 42, 50]. FCA provides a way to identify sensible groupings of objects that have common attributes and gives a technique to create the concept lattice. The central idea of formal concept analysis is the understanding that a fundamental unit of thought is a concept and a context. A concept consists of two parts :

- Extent that contains all objects common to all attributes
- Intent that contains all attributes common to all objects

2.6.1. Basic Notions of FCA

A **Formal Context** is a triple (O, A, R) consisting of two sets O and A and a relation R, where :

- O is a set of objects
- A is a set of attributes
- R is a binary relation between O and A

A Formal Context is expressed as **oRa** or **(o, a)$\in$R** where $o\in O$, $a\in A$, and is read as "The object o has the attribute a".

A Formal Concept of the Formal Context (O, A, R) is a pair (E, I) with the following conditions :

$E\subseteq O$, $I\subseteq A$, $E'=I$ and $I'=E$

where :

- E is a set of objects (Extent)
- I is a set of attributes (Intent)
- E' is a subset of A satisfying oRa for all $o\in E$

- I' is a subset of O satisfying oRa for all $a \in I$

The set of all concepts of (O,A,R) ordered by the relation($\leq$) is called the **Concept Lattice** of (O,A,R). The relation($\leq$) is called the hierarchical order or simply order of the concepts and is defined as :

$(E_1, I_1) \leq (E_2, I_2)$

iff $E_1 \subseteq E_2$

or *iff* $I_2 \subseteq I_1$

2.6.2. FCA Example

An example given in [16] is used to describe a general idea about FCA and how it can be applied to a given application. This example is about an educational file "Living Beings and Water". First, a set of objects and attributes are presented and then with the two sets, demonstration for the processes of obtaining Formal context, Formal concept and Concept lattice is presented.

2.6.2.1. Object Set

Elements of object set could be anything such as people, animals, and human being and their creation but all should have common properties among them. For example, Floppy disk and CD have different properties but share the common properties such as shape, storage capability, etc. In this example, as elements of an object set, the following eight living creatures are used :

1. Leech
2. Bream
3. Frog
4. Dog
5. Spike-weed
6. Reed
7. Bean
8. Maize

2.6.2.2. Attributes Set

Elements of attribute set explain some relationship among elements of the object set. For example, let an attribute set has 3 elements {an item can store electronic data, an item is bendable, an item has round shape} and an object set has 2 elements {Floppy disk, CD}. The elements of attribute set explain a relationship between a floppy disk and a CD. Both of them can store electronic data and have round shape (common properties). A floppy disk is bendable but a CD is not (different property). As elements of attribute set in this example, the following nine attributes are used :

a : needs water to live

b : lives in water

c : lives on land

d : needs chlorophyll to produce food

e : two seed leaves

f : one seed leaf

g : can move around

h : has limbs

i : suckles its offspring

In order to simplify representation of attributes, the above nine attributes are recognized only by the alphabets (a – i).

2.6.2.3. Formal Context

From the above two sets (object and attribute sets), the Formal Context can be derived. Usually, Formal Context is represented by a cross table as given in Figure 2. The first row represents all elements of the attribute set whereas the first column represents all elements of the object set. For convenience, numbers (1, 2, ..., 9) and alphabets (a, b, ..., i) are used to represent all elements of object set and attribute set respectively.

		a	b	c	d	e	f	g	h	i
1	Leech	X	X					X		
2	Bream	X	X					X	X	
3	Frog	X	X	X				X	X	
4	Dog	X		X				X	X	X
5	Spike-weed	X	X		X		X			
6	Reed	X	X	X	X		X			
7	Bean	X		X	X	X				
8	Maize	X		X	X		X			

Figure 2. Cross Table.

"X" in a cell of the cross table implies that the particular object posses that attribute. For example, Leech has marked attributes a, b, and g meaning that leech needs water to live, lives in water, and has two seed leaves.

2.6.2.4. Formal Concept

A Formal Concept is a pair comprising of an object set and an attribute set. From the cross table in Figure 2, followings the definition of a Formal Concept, 19 Formal Concepts are derived. Each of these Formal Concepts according to the above definition is given as :

- Concept 1 : ({1,2,3,4,5,6,7,8,}, {a})
- Concept 2 : ({1,2,3,5,6}, {a,b})
- Concept 3 : ({3,4,6,7,8}, {a,c})
- Concept 4 : ({1,2,3,4,}, {a,g})
- Concept 5 : ({5,6,7,8}, {a,d})
- Concept 6 : ({1,2,3}, {a,b,g})
- Concept 7 : ({2,3,4}, {a,g,h})
- Concept 8 : ({5,6,8}, {a,d,f})
- Concept 9 : ({6,7,8}, {a,c,d})
- Concept 10 : ({2,3}, {a,b,g,h})
- Concept 11 : ({3,4}, {a,c,g,h})
- Concept 12 : ({3,6}, {a,b,c})
- Concept 13 : ({5,6}, {a,b,d,f})
- Concept 14 : ({6,8}, {a,c,d,f})

- Concept 15 : ({3}, {a,b,c,g,h})
- Concept 16 : ({4}, {a,c,g,h,i})
- Concept 17 : ({6}, {a,b,c,d,f})
- Concept 18 : ({7}, {a,c,d,e})
- Concept 19 : ({}, {a,b,c,d,e,f,g,h,i})

The first set consists of numbers representing an object set in which each number represents each object described in Section 2.6.2.1. The second set consists of alphabets representing an attribute set such that alphabet represents an attribute in Section 2.6.2.2. For example, the [concept 12] contains "3" and "6" as elements of object set, and "a", "b", and "c" as elements of attribute set and represents that "Frog and Reed need water to live, live in water and live on land".

2.6.2.5. Concept Lattice

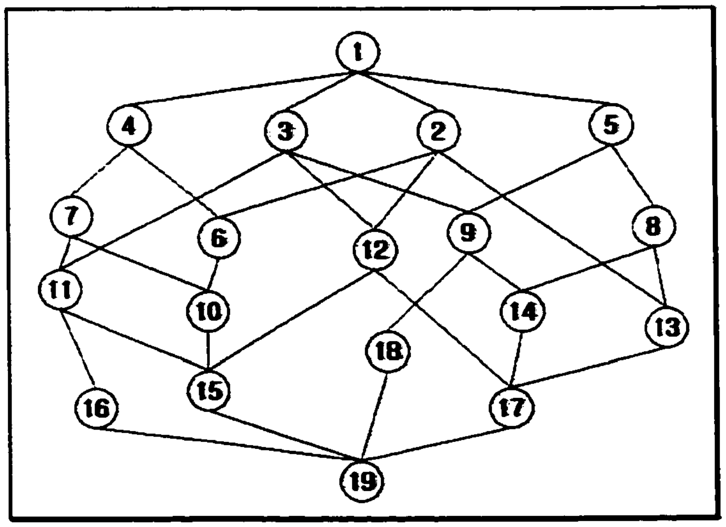


Figure 3. Concept Lattice

Concept Lattice looks like a hierarchical tree structure. In Concept Lattice, all Formal Concepts are linked to each other by the definition of the relation($\leq$). From above Formal Concept in this figure, a Concept Lattice structure as shown in Figure 3 is obtained. A circle represents a concept and the number in a concept represents concept numbers.

3. Thesis Approach to Image Retrieval System

In this thesis, two methodologies are introduced in order to add images into a lattice structure. The first methodology is a bit set structure representing an attribute set structure based on the 1-D string of the 2-D string. The attribute set is one of sets in concept nodes and represents real object in the world. The second is an Addition Method for rebuilding a lattice structure. This addition method rebuilds only a part of a lattice structure when an image is added into the lattice structure.

3.1. Attribute Set Structure

Let O be a finite set of symbols representing some real world objects given as :

$$O = \{a, b, c, \ldots\}$$

where a, b, c, ... are the real objects.

Like 1-D string as mentioned in Chapter 2, a representation of an attribute set forms $x_1x_2 \ldots\ldots x_n$, where x_i is an object in real world. The difference between attribute set and 1-D string is that spatial information is provided in 1-D string but is not provided in an attribute set. For example, $(A < B < C : D)$ was described as the representation of 1-D string of u in Section 2.5. This 1-D string contains both object (A, B, C, and D) and spatial (< and :) information. However, (ABCD) is the representation of attribute set containing only object information. Then a bit set structure is used to represent each attribute set.

3.1.1. Bit Set Structure

To represent image attributes, the use of bit set structure, also known as bit vector, is proposed. A bit set structure is composed of a fixed number of bits in which each bit represents a real object appearing in an image. If the universal set U contains N items, an N-bit vector can represent any subset S of U. Bit 'i' will be 1 if $i \in S$, otherwise bit 'i' is set to 0. Therefore, it is initialized to '0' to indicate an empty set. This scheme requires use of only 1 bit per element. Therefore, an advanced knowledge of the size of universe is required. However, it is a very space efficient even for large values of U. Element insertion or deletion simply requires flipping the appropriate bit. Intersection can be performed simply by performing the AND operation.

Use of bit set structure allows for efficiently finding one or more attributes. For example, suppose there is five attributes (a, b, c, d, e) as elements of an attribute set without using a bit set structure, and an "e" in the attribute set is considered as a query attribute. In worse case, five comparisons are needed to find "e". The query character "e" is compared with "a" to "e" until "e" in the attribute set is found. Therefore, five comparisons are needed as shown in the Figure 4-Case1. If the query attribute set contains "d" and "e" then nine comparisons are needed in worst case. First, "d" is compared with "a", "b", "c", "e", and "d" in worst case then five comparisons are needed to find "d" in the attribute set. Second, "e" is compared with "a", "b", "c" and "e" except for "d" in worst case then four comparisons are needed to find "e" in the attribute set shown in Figure 4-Case2. However, if the bit set structure is used, only two comparisons are needed in both cases because "AND" and "EQUALITY" operations are used.

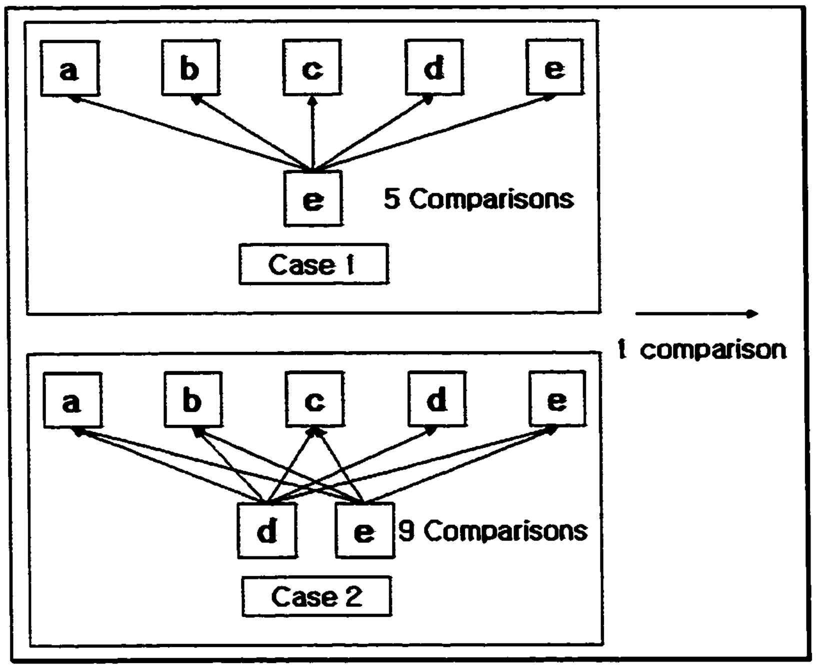


Figure 4. Finding attributes in general

By using AND operation (intersection), an intersected-attribute set is obtained then by using EQUALITY operation, the intersected-attribute set is compared with the query attribute set. Similarity of two attribute sets implies that an attribute set satisfying the query attribute set is found. Formally, the above explanation is described as follow :

$$a_g \cap a_q = a_i$$
$$\textit{if} \quad a_q = a_i \quad \textit{then} \quad a_g \quad \textit{is} \quad \textit{the} \quad \textit{attribute} \quad \textit{set}$$
$$\textit{if} \quad a_q \neq a_i \quad \textit{then} \quad a_g \quad \textit{is} \quad \textit{not}$$

, where

a_g is a given attribute set

a_q is a query attribute set

a_i is a intersected attribute set.

For example, in Figure 5, the first comparison occurs when an attribute set is intersected with the query set by using AND operation. The second comparison occurs when the equality of the intersected attribute set and query set is checked. Therefore, a procedure that finds an attribute set satisfying a query attribute set needs two comparisons if a bit set structure is used.

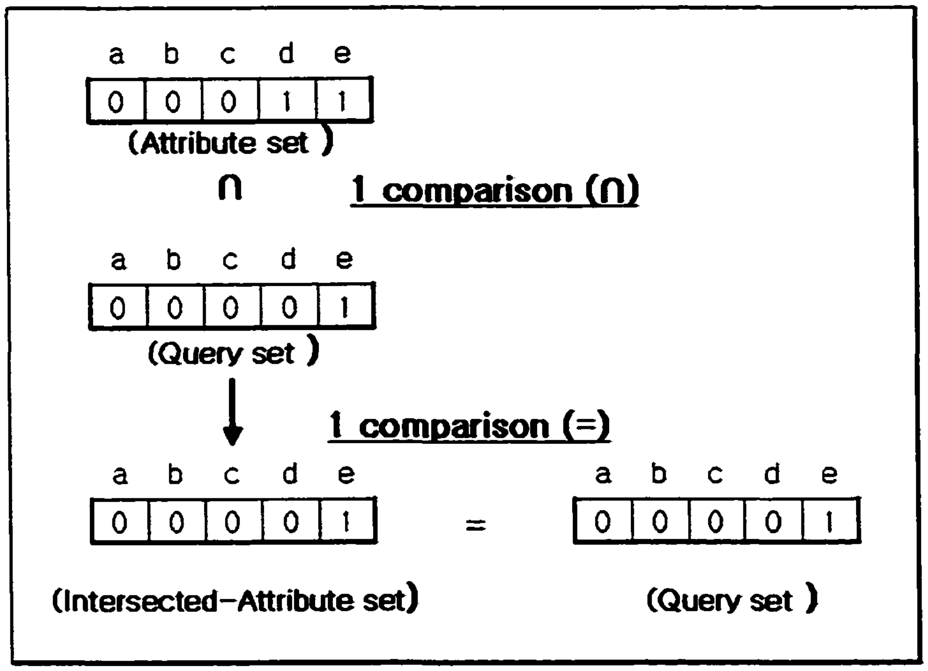


Figure 5. Finding attributes with an bit set structure

3.1.2. Example of a Bit Set Structure

In order to demonstrate use of a bit set structure, consider the image shown in Figure 6 with attributes Bridge (a), Parking Lot (e), Car (g), Streetlight (o), and Tree (p). Suppose that the size of bit set structure is 16 bits. It is capable of representing 16 different attributes marked a through p.

Objects in this image :
Bridge(a), Parking Lot(e), Car(g)
Streetlight(o), Tree(p)

Figure 6. An example image for Bit Set Structure

From the image in Figure 6, an attribute set {a, e, g, o, p} is obtained. Initially all of the bits of bit set structure are set to "0" as shown in Figure 7-a. With two known facts (attribute set and bit set structure), a bit set structure, which represents the image in Figure 6 such that we assign "1" to "a", "e", "g", "o", and "p" is obtained. The resultant bit set structure is shown in Figure 7-b.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a)

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
1	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1

(b)

Figure 7. Bit set structure representing image attributes of Figure 6

3.2. Building a Lattice Structure

Until recently, because of time complexity, only few systems dealing with relatively small amount of data have applied FCA. The worst case time complexity for building a lattice structure in general is $O(n^n)$. A fast algorithm for building lattice is introduced at Harvard University in which the time complexity is said to be $O(n^4)$ [27]. However, the time complexity is still not good enough for applications dealing with large amount of data. To build a lattice structure, all of the objects are collected, all concepts are extracted, and then the concepts are linked. However, for any addition of a new object, whole lattice structure needs to be rebuilt causing expensive mathematical and computing operation, thus limiting the use of FCA in different potential applications [27].

In this section, an addition method is introduced to build a lattice structure. This addition method is useful only when a lattice structure is already existed since it simply rebuilds only a part of the lattice structure rather than the entire lattice structure. The time complexity of this method is $O(2^n)$, where n is the number of attributes.

Lemma 1. : The total number of concepts in a lattice structure is at most 2^n-2.

Proof :

Let n be the number of attributes and S_n be the number of concepts in a lattice structure (excluding the top and the bottom nodes).

By inductive method

if n = 1 then $S_1 = 0$

if n = 2 then $S_2 = {}_2C_1 = 2$

if n = 3 then $S_3 = {}_3C_1 + {}_3C_2 = 3 + 3 = 6$

...

For an arbitrary n, it is necessary to find the number of concepts.

If n = k then $S_k = {}_kC_{k-1} + {}_kC_{k-2} + \ldots + {}_kC_1 = \sum_{r=1}^{k-1} {}_kC_r$

Now, by binomial expansion ($(a+b)^n = \sum_{r=0}^{n} {}_nC_r a^{n-r} b^r$) and by substitution a = 1 and b = 1, we obtain :

$$2^n = {}_nC_0 + {}_nC_1 + \ldots + {}_nC_{n-1} + {}_nC_n, \text{ where } {}_nC_1 + \ldots + {}_nC_{n-1} = \sum_{r=1}^{n-1} {}_nC_r$$

$$\Rightarrow \sum_{r=1}^{n-1} {}_nC_r = 2^n - 2$$

Therefore, the total number of concepts $S_n = 2^n - 2$.

Corollary 1.1. : Possible concepts[1] are obtained among 2^n–2 concepts in worst case.

Possible concepts are obtained by comparing an attribute set with attribute sets in a lattice structure and in worst case, 2^n–2 concepts are in the lattice structure as described in Lemma 1. Therefore, possible concepts are obtained by checking among 2^n–2 concepts in the lattice structure. Steps for how to get and how to use possible concepts are described in Section 3.2.1.

Lemma 2. : The worst case time complexity of addition method is O(2^n).

Proof :

As described in Lemma 1., 2^n-2 concepts exist in a lattice structure in worst case and possible concepts are obtained among 2^n–2 concepts as described in Corollary

[1] Possible Concept : a possible concept is a concept, which could be one of concepts of a lattice structure. If the lattice structure already has a concept, which has same attribute set as that of a possible concept, the possible concept does not need to be added into the lattice structure. If not, the possible concept is added. Therefore, we call it as a possible concept.

1.1. An addition method described in 3.2.1. extracts these possible concepts. In other words, the addition method checks 2^n-2 concepts except for the top and the bottom nodes in the lattice structure and extracts all possible concepts. Therefore, 2^n-2 comparisons are needed i.e., the worst time complexity for addition method is $O(2^n-2) = O(2^n)$.

3.2.1. Addition Method

In the proposed addition method to add new concept in an existing lattice structure, first, all possible concepts related to the new object are found then the superconcepts and subconcepts of all or some of possible concepts are tried to be find. Finally, all or some of possible concepts are linked to them (superconcept and subconcept) by the hierarchical order of FCA. As explained in the annotation of Possible Concept, if an attribute set of possible concept is the same as one of attribute sets in a lattice structure, the attribute set of possible concept is not added into a lattice structure. Therefore, in some cases, the addition method deals with only some of possible concepts rather than all of them. In this section, the addition method is introduced with six steps.

Step 1

In this step, the addition method finds possible new concepts by comparing the attribute set of new object with the attribute sets of concepts in a lattice structure. The size of possible new concepts obtained in this step determines the efficiency of addition method. The more possible new concepts created, the more time taken. However, in this addition method, all concepts in a lattice structure are examined on the first examination as shown in Lemma 2. As a result, redundant attribute sets[2] or empty attribute sets[3] of possible new concepts are created in some cases. Therefore, the second examination is needed to remove some of possible concepts whose attribute sets are redundant attribute sets or empty attribute sets. Formally, let a_i be an attribute set of a possible new concept, a_r be a redundant attribute set, a_e be an empty attribute set, where $1 \leq i \leq 2^n-2$.

- If $|a_i| = 0$ then $a_i \equiv a_e$.
- if $|a|_i \neq 0$ and $a_i \equiv a_j$, where $1 \leq j < i$ and $i < j \leq 2^n-2$ then $a_i \equiv a_r$.

Consequently, a possible new concept whose attribute set is considered as a_r or a_e is removed in this step.

Step 2

After possible new concepts are computed in Step 1, it is necessary to find superconcepts of those concepts. This step employs the depth first and top-down search methodologies. From the top node, subconcepts of current node are checked with conditions described below. The lattice structure is a hierarchical tree structure requiring checking of the same concept more than once. Therefore, by using the characteristic of the depth first search, this step avoids checking a concept, which is already examined.

In order to be a superconcept for possible concepts, each concept in a lattice structure must satisfy the following simple conditions, which are based on the order relation ($\leq$) as described in Section 2.6.1.

[2] Redundant Attribute Set is an attribute set whose elements are same as one of attribute sets in a lattice structure. In other words, the redundant attribute set is already existed in a lattice structure.
[3] Empty Attribute Set is an attribute set whose size is 0. The attribute set is obtained when the intersections of new object's attribute set and attribute sets in the lattice structure are empty set.

Condition 1 : Let C1 be a concept in a lattice structure, a_1 be an attribute set of C1, a_i be an attribute set of subconcept of C1 and a_n be an attribute set of possible concept. If ($a_1 \subseteq a_n$) and ($\forall i$, $a_i \not\subset a_n$) then C1 is a superconcept of the possible concept as shown in Figure 8.

Condition 2 : if two attribute sets are the same and the number of elements of a object set in a possible concept is greater than the number of elements of a object set in a concept of lattice structure, then an object set of a concept in lattice is replaced with the object set of a possible concept.

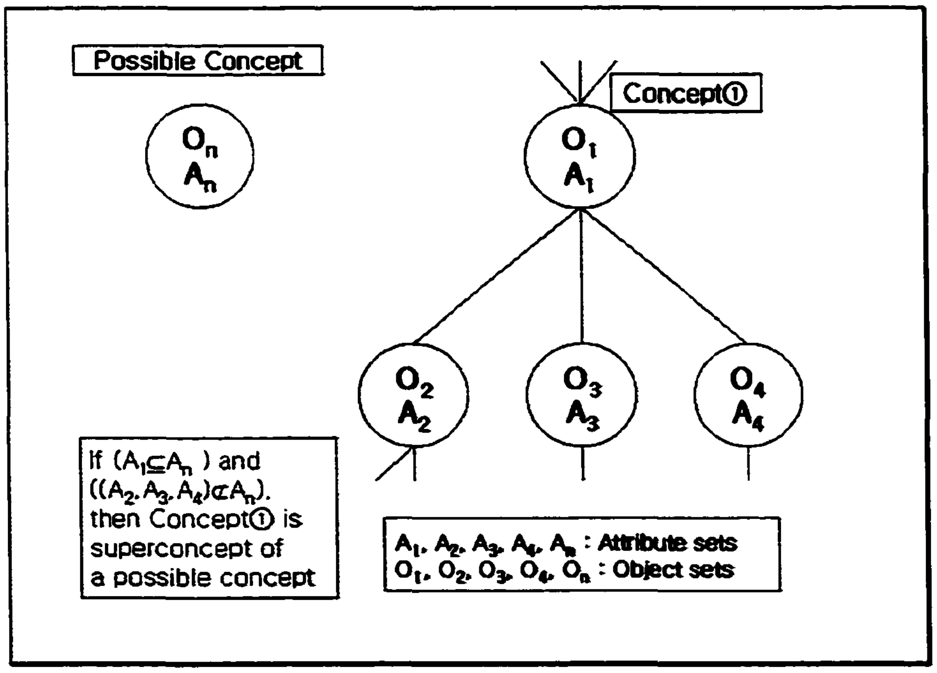

Possible Concept
Concept①
O_n A_n
O_1 A_1
O_2 A_2
O_3 A_3
O_4 A_4
If ($A_1 \subseteq A_n$) and ((A_2, A_3, A_4)$\not\subset A_n$), then Concept① is superconcept of a possible concept
A_1, A_2, A_3, A_4, A_n : Attribute sets
O_1, O_2, O_3, O_4, O_n : Object sets

Figure 8. Finding superconcepts (Step 2)

Step 3

In Step 2, superconcepts of possible concept are found. In this step, it is necessary to find subconcepts of possible concept. This step is based on the depth first and bottom-up search methodologies. To find subconcepts, it is not necessary to check the concepts since these are already checked in Step 2. Therefore, in this step, the bottom-up search is used instead of a top-down search. The condition of Step 3 is similar to the Condition 1 of Step 2 and is given as :

Condition 1 : Let **C1** be a concept in a lattice structure, $\mathbf{a_1}$ be an attribute set of C1, $\mathbf{a_i}$ be an attribute set of superconcept of C1 and $\mathbf{a_n}$ be an attribute set of possible concept. If ($\mathbf{a_n} \subseteq \mathbf{a_1}$) and ($\forall \mathbf{i}$, $\mathbf{a_n} \not\subset \mathbf{a_i}$) then C1 is a subconcept of the possible concept as shown in Figure 9.

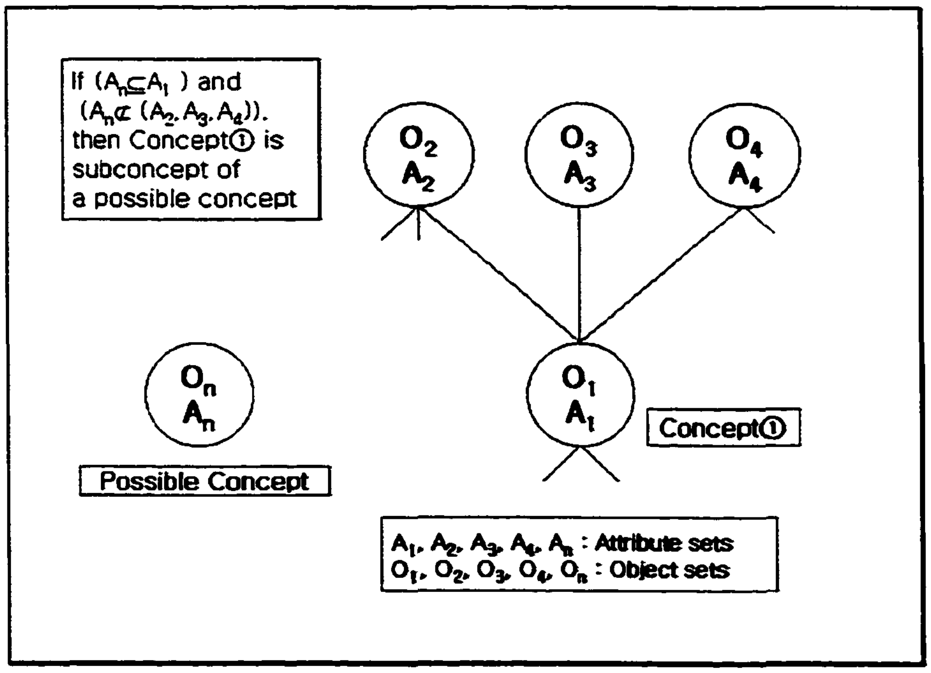

If ($A_n \subseteq A_1$) and ($A_n \not\subset (A_2, A_3, A_4)$), then Concept① is subconcept of a possible concept
O_2 A_2
O_3 A_3
O_4 A_4
O_n A_n
Possible Concept
O_1 A_1
Concept①
A_1, A_2, A_3, A_4, A_n : Attribute sets
O_1, O_2, O_3, O_4, O_n : Object sets

Figure 9. Finding subconcepts (Step 3)

Step 4.

In this step, it is necessary to link a possible concept to its superconcepts and subconcepts, as found in Step 2 and Step 3. However, in this process, it is also necessary to find if a superconcept and a subconcept of a possible concept are already linked. If it is the case, it is necessary to disconnect them and connect them again to a possible concept as shown in Figure 10.

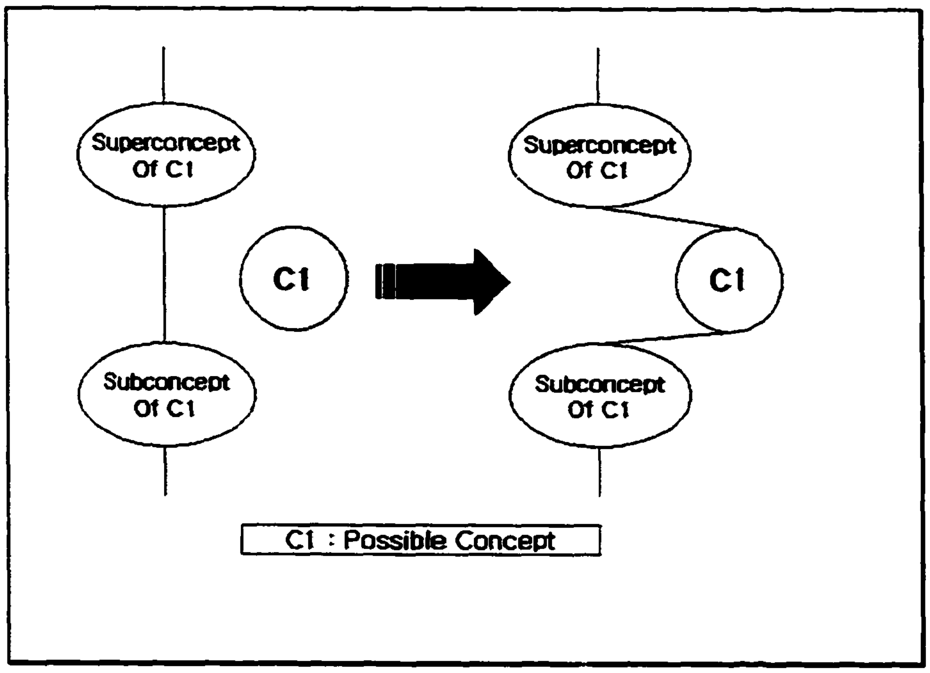


Figure 10. Linking superconcepts and subconcepts to a possible concept

Until now, only a single possible concept as created in Step 1 has been dealt. However if more than two possible concepts are created, it is necessary to repeat Step 2 - Step 4 for remaining possible concepts. After addition of all possible concepts, it is essential to add the new object as described below in Step 5 and Step 6.

Step 5

This step is similar to Step 2 and Step 3. The difference hence is that new object rather than a possible concept is added. However, before carrying out this step, it is essential to create a concept for the new object as shown in Figure 11 followed by Step 5, which is the same as Step 2 and Step 3.

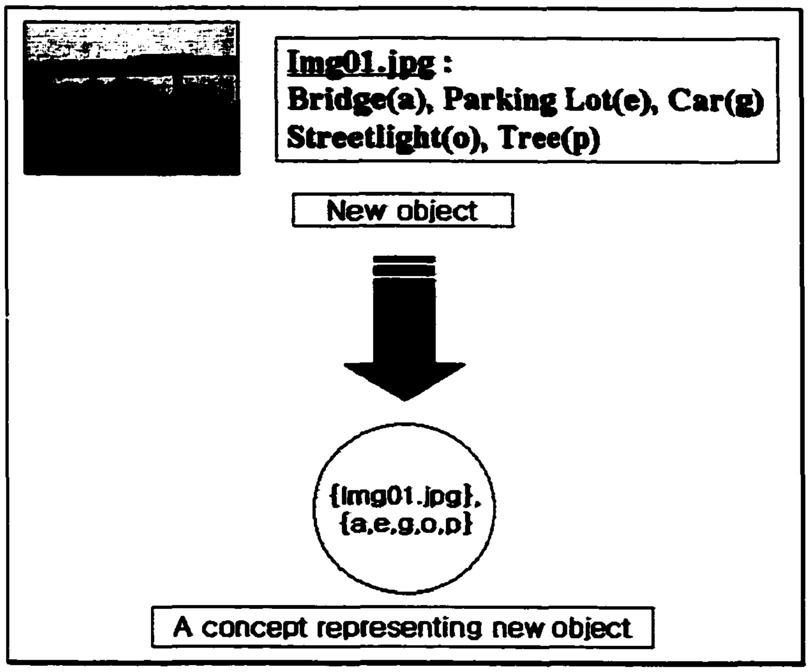


Figure 11. Create a concept for new object

Step 6

In this step it is essential to link the concept of new object to the superconcepts and subconcepts as found in Step 5. This step is the same as Step 4.

3.2.2. Example of Building a Lattice Structure

Initially, there is an empty lattice containing only [T] and [B] nodes, which represent the TOP and the BOTTOM respectively (Figure 12-a). The addition of the first object into a lattice structure does not require Step 1 to Step 4 because no concepts are presented in the initial lattice structure. During the processing, a concept for the new object is created as shown in Figure 11 and then the concept is added to the lattice structure connecting it to [T] and [B]. The nodes [T] and [B] become a superconcept and a subconcept respectively of the first object as shown in Figure 12-b.

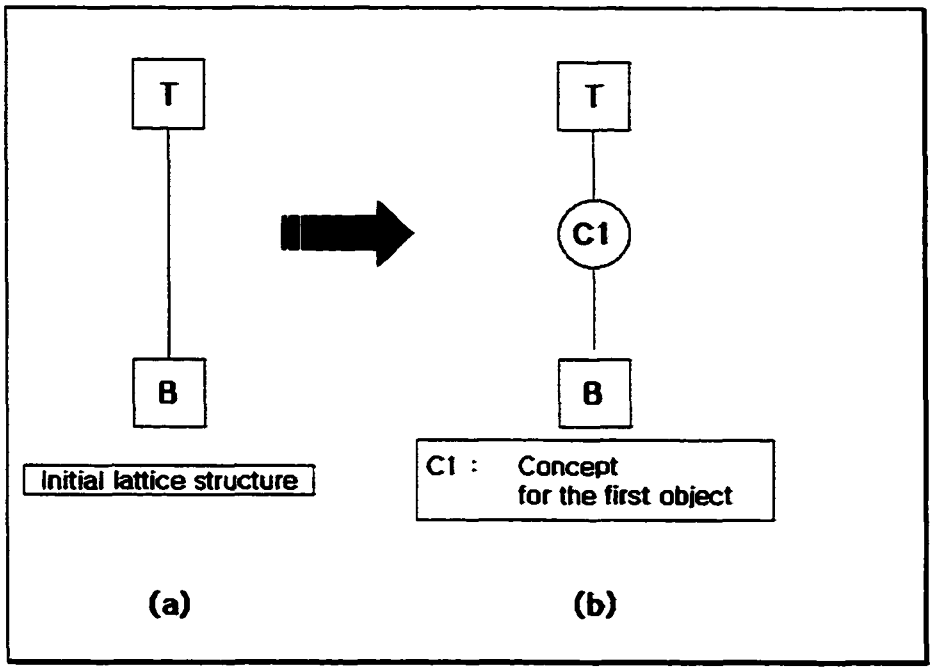


Figure 12. (a) Initial lattice structure, (b) after adding first object

Suppose two objects (Figure 13-a) are already added into an initial lattice structure resulting in a lattice structure with three concepts as shown in Figure 13-b. Now the third image as shown in Figure 14 is to be added. According to Step 1, the attribute set of the third object is intersected with the 3 attribute sets of the existing concepts (C1, C2, C3) in the lattice structure and three possible concepts (PC1, PC2, PC3) are obtained as shown in Figure 15.

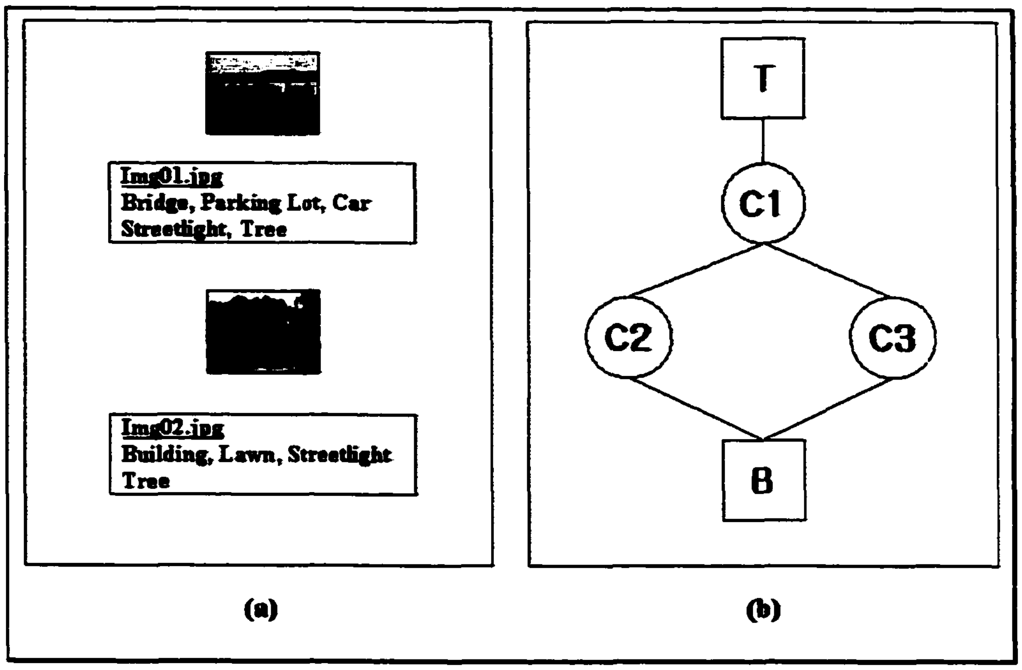


Figure 13. (a) Objects and (b) their addition into the lattice structure

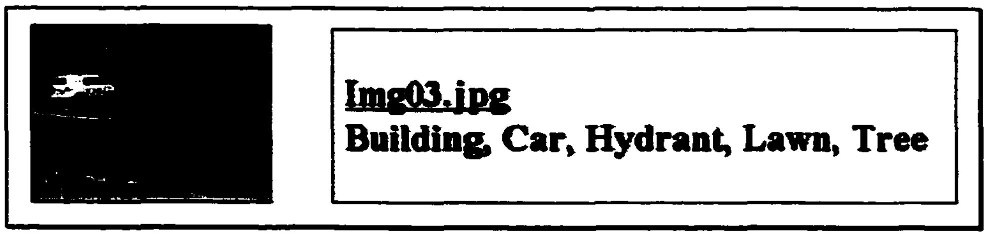


Figure 14. Addition of 3rd image and objects

After obtaining possible concepts, it is essential to find a superconcept and a subconcept for PC1 concept. In this cas, [T] and C1 are the superconcept and the subconcept repectively and need to be linked to PC1 (Figure 16-a). For PC2, PC1 is a superconcept and C2 is a subconcept. After linking these, the resulting lattice structure is shown in Figure 16-b. For PC3, PC1 is a superconcept and C3 is a subconcept. After linking these, the lattice structure is obtained as shown in Figure 16-c.

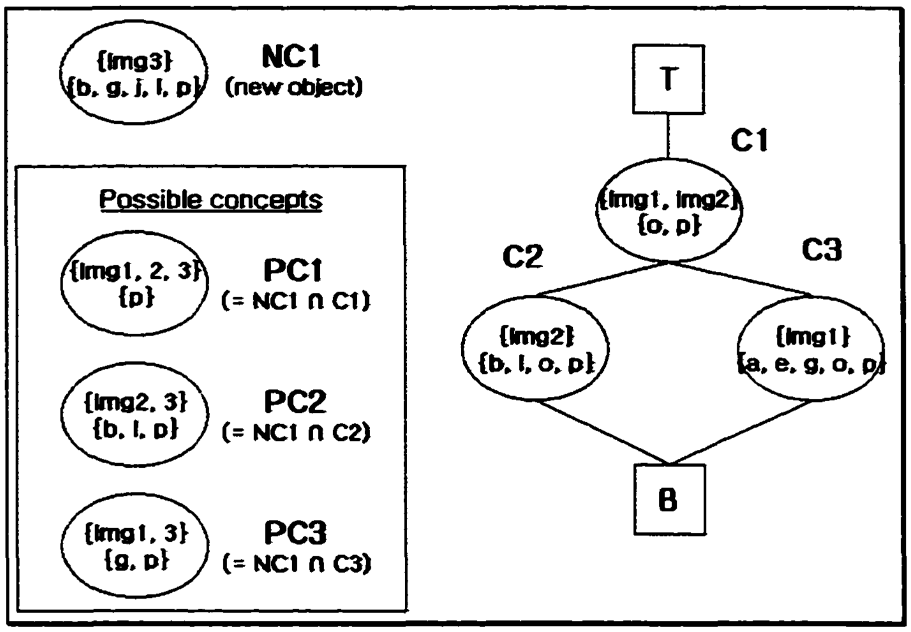


Figure 15. Possible concepts

So far, only three possible concepts, which are obtained by intersecting new object with existing concepts in a lattice structure have been added. Now, it is essential to add a new object. The steps are involved below :

- Find superconcepts : In this case, two superconcepts (PC2 and PC3) are obtained.
- Find subconcept : which is the Bottom node [B].
- Connect the nodes.

The resulting lattice structure with seven concepts after addition of three objects is shown in Figure 17. With this addition method, new objects have been added by rebuilding only a part of the lattice structure rather than rebuilding the entire lattice structure.

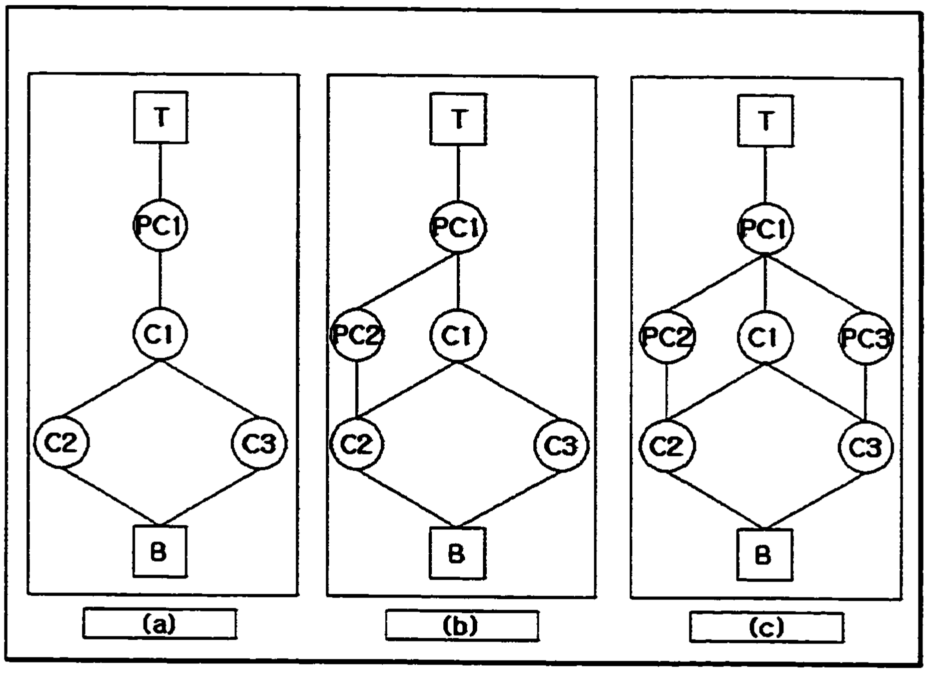


Figure 16. Addition of three possible concepts

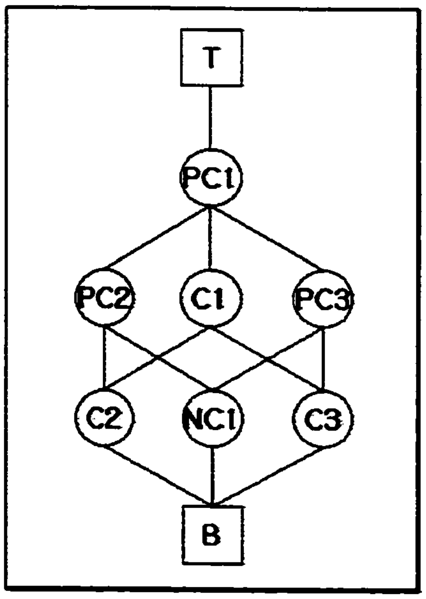


Figure 17. Add a new object after adding possible concepts

3.3. Image Retrieval

To retrieve images corresponding to a query image, it is necessary to find a concept. This is because a concept in a lattice structure consists of an object set and an attribute set. The object set consists of images and the attribute set consists of real objects. Therefore, if a concept is found by comparing its attribute set with query attribute set, images, which satisfy with the given criteria, are retrieved from its object set.

In order to search for images corresponding to a query image, we start with any superconcept of the Bottom node [B] then examine attribute sets of concepts in a lattice structure to find the concept. If the attribute set obtained after intersecting a query attribute set with a superconcept of [B] is the same as the query attribute set, then we keep searching by comparing a superconcept of the superconcept of [B] with the query set until the condition below is satisfied. For example, suppose $\mathbf{a}_l$ is an attribute set of a superconcept of [B], $\mathbf{a}_q$ is an attribute set of query image, and $\mathbf{a}_i$ is an attribute set after intersecting $\mathbf{a}_q$ with $\mathbf{a}_l$. In other words, $\mathbf{a}_q \cap \mathbf{a}_l = \mathbf{a}_i$. If $\mathbf{a}_i$ is the same as $\mathbf{a}_q$ ($\mathbf{a}_i \equiv \mathbf{a}_q$) then we keep checking superconcepts of the concept, whose attribute set is $\mathbf{a}_l$ until the following condition is satisfied :

Condition : If the intersected attribute set obtained by interesting a query attribute set with a concept① attribute set in a lattice structure is a subset of the concept①, and the intersected attribute set is not a subset of superconcepts of the concept①, then the concept① is the concept satisfying the user query. For example, suppose a concept① has 3 superconcepts and $\mathbf{a}_l$ is the attribute set of the concept① and $\mathbf{a}_{s1}$, $\mathbf{a}_{s2}$, $\mathbf{a}_{s3}$ are the attribute sets of the three superconcepts of concept① and $\mathbf{a}_q$ is the attribute set of query image. To satisfy the above condition, the following conditions must hold :

- $a_1 \cap a_q = a_q$
- $a_{s1} \cap a_q \neq a_q$
- $a_{s2} \cap a_q \neq a_q$
- $a_{s3} \cap a_q \neq a_q$

Formally, the condition is described as follow :

Let	C_i	be a concept from C_p in a lattice structure
	C_{si}	be one of superconcept of C_i
	C_p	be a subconcept of C_i
	a_i	be an attribute set of C_i
	a_{si}	be an attribute set of C_{si}
	a_q	be an attribute set of query

- if $a_i \cap a_q \neq a_q$ then move back to C_p
- if $a_i \cap a_q \equiv a_q$ and $\exists i, a_{si} \cap a_q \equiv a_q$ then move to C_{si} and keep searching
- if $a_i \cap a_q \equiv a_q$ and $\forall i, a_{si} \cap a_q \neq a_q$ then C_i is the concept, which is satisfied with given user query.

3.3.1. Image Retrieval Examples

Suppose a query set consists of attributes {b, l, o, p} and for convenience, only a part of the lattice structure is taken as shown in Figure 18. Figure 18 shows part of the complete lattice structure given in Figure 3.

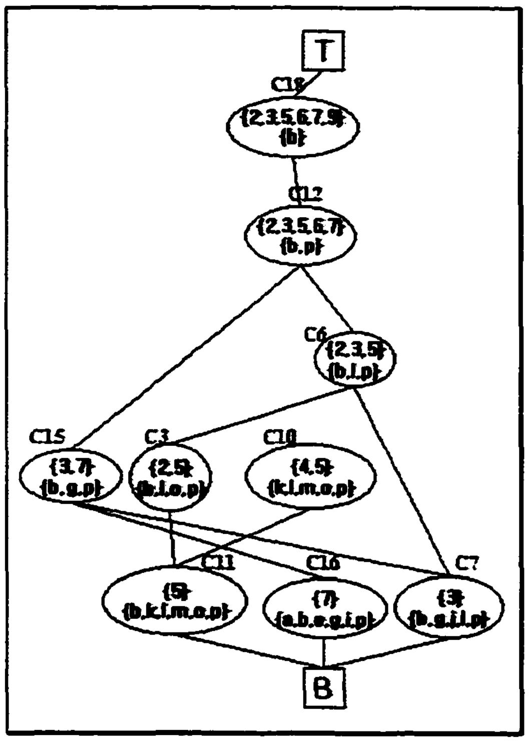


Figure 18. Section of complete lattice of Figure 3.

- At [B], check the superconcepts of [B].

<u>Concept C7</u> :

$\{b,l,o,p\}_{\text{query}} \cap \{b,g,j,l,p\}_{\text{Concept C7}} = \{b,l,p\}_{\text{intersected set}}$

$\{b,l,o,p\}_{\text{query}} \neq \{b,l,p\}_{\text{intersected set}}$

Go back to [B], and check the concept C11.

<u>Concept C11</u> :

$\{b,l,o,p\}_{\text{query}} \cap \{b,k,l,m,o,p\}_{\text{Concept C7}} = \{b,l,o,p\}_{\text{intersected set}}$

$\{b,l,o,p\}_{\text{query}} = \{b,l,o,p\}_{\text{intersected set}}$

Then go to the concept C11.

- At Concept C11, check the superconcepts of C11.

<u>Concept C3</u> :

$\{b,l,o,p\}_{\text{query}} \cap \{b,l,o,p\}_{\text{Concept C7}} = \{b,l,o,p\}_{\text{intersected set}}$

$\{b,l,o,p\}_{\text{query}} = \{b,l,o,p\}_{\text{intersected set}}$

Then go to the concept C3.

- At Concept C3, check the superconcepts of C3.

  <u>Concept C6</u> :

  $\{b,l,o,p\}_{\text{query}} \cap \{b,l,p\}_{\text{Concept C7}} = \{b,l,p\}_{\text{intersected set}}$

  $\{b,l,o,p\}_{\text{query}} \neq \{b,l,p\}_{\text{intersected set}}$

  Go back to C3, and no more superconcepts of C3.

- At C3, there are no more superconcepts to be checked, and a query set is a subset of Concept C3, then the Concept C3 is the concept, which satisfies with the given query. After finding C3, the images (Img2, and Img5) can be retrieved by checking the object set of the Concept C3.

3.4. Search and Retrieval Scheme

Retrieval of images in this case is a combination of the depth first and bottom-up search. This section describes the significance of the use of the bottom-up search in this scheme. Three cases are introduced to demonstrate its use and importance.

Why bottom-up search

Because of hierarchical structure, generally a top-down search is applied to find a concept in the lattice structure of FCA. From the top, the attributes are extended while reducing the objects. For example, suppose there are "CD" and "Floppy Disk" as elements of the object set, and "item can store electronic data", "item is bendable", and "item has round shape" as elements of attribute set. At the beginning, an attribute "item can store electronic data" is only suggested then objects "CD" and "Floppy Disk" together can be retrieved. However, if another attribute "item has round shape" as the second attribute is also suggested then only "CD" is retrieved. Because of this advantage that more specific object is retrieved by extending more attributes, the top-down search is commonly used. However, the top-down search is not efficient in this case. The reason will be explained in Case 1 and Case 2 below. Another reason that a bottom-up search is used is to reduce the number of comparisons. If a top-down search is used, more comparisons are made. Therefore, the use of a bottom-up search is suggested. To demonstrate the reasons for use of bottom-up search, three cases are introduced. The first two cases are about the top-down search and the last case (Case 3) is about how to solve the previous two problems (Case 1 and Case 2) with the bottom-up search

Case 1 (Problem 1 of top-down search)

Let {l, o} is the query attribute set. From the node [T] in Figure 3, any concepts containing either {l} or {o} is not found and searching is stopped, even though concepts (C8, C3, C11) containing {l, o} exist in the lattice. Therefore, the top-down search fails to provide required information. This situation occurs because of the characteristics and properties of the lattice structure. A lattice structure is hierarchical structure and implies that superconcepts have less attributes then subconcepts. In the absence of full number of concepts in a lattice structure in this system, this problem will keep on occurring, thus, resulting in less than satisfactory results.

Case 2 (Problem 2 of top-down search)

Consider an attribute set {b, g, j, p}. In order to take an advantage of the bit set structure, intersection of two attribute sets is used resulting in one of the following two cases :

- Case ① : compare the intersected attribute set with a query attribute set. If the intersected attribute set is a subset of a query attribute set, then continue search.
- Case ② : compare the intersected attribute set with a concept attribute set. If the intersected attribute set is the same as the concept attribute set, then continue search.

In case ①, the extra comparisons are needed. For example, $\{b, g, j, p\} \cap \{p\}_{\text{Concept C5}} = \{p\}_{\text{intersected set}}$. To decide whether the intersected set is a subset of query set, the {p} has to be compared with every element of {b, g, j, p}. That means four extra comparisons are needed. In case ②, sometimes it is impossible to find a concept. For example, $\{b, g, j, p\} \cap \{b, g, j, l, p\}_{\text{Concept C7}} = \{b, g, j, p\}_{\text{intersected set}}$. The intersected set and the concept(C7) attribute set are different. Concept C7 is rejected, even though the Concept C7 is the concept, which satisfied the given query set.

Case 3 (Solving the problems with Bottom-up)

As mentioned earlier in Section 3.3, the two attribute sets are intersected and the intersected set is compared with the query attribute set. With the query attribute set given as {l, o}, consider the concept C11 of Figure 3 with attribute set {b, k, l, m, o, p}. $\{l, o\}_{\text{query}} \cap \{b, k, l, m, o, p\}_{\text{Concept C11}} = \{l, o\}_{\text{intersected set}}$. Now the intersected set {l, o} is compared with the query set {l, o}, which are same. Therefore, it is essential to check the superconcepts (C3, and C10) of concept(C11). After comparison with C3, the same result is generated and is essential to check the superconcepts (C6, C8) of C3. With C6, the intersected set is not the same as that of query set. With C8, we have a matching attribute set, thus we need to check the superconcept (C2, C9) of C8. Neither C2 nor C9 gives us a matching attribute set. Since C8 contains the attributes we are looking for but not its superconcepts (C2 and C9), C8 is the concept we are looking for. The problem in Case 1 above is solved by using bottom-up search.

In order to demonstrate the proposed solution for the above problem of Case 2, we use the attribute set {b, g, j, p} and follow the same procedure as outlined above. With a concept (C7), we have satisfying results, so we need to check the superconcepts (C6, C15). Neither C6 nor C15 gives us the satisfied result. Therefore, the C7 is the concept we are looking for, and its attribute set contains the required information {b, g, j, p}.

4. Image Retrieval Time

It has been discussed that the number of attributes for retrieval image rather than the number of images or concepts for retrieval image because the number of attributes is the main factor in this scheme. The time complexity of retrieving images is also discussed based on the number of attributes. The time complexity of retrieving images is considered to be the time complexity of finding a concept whose attribute set is equal to the attribute set of the user query.

Lemma 3. : The worst case time complexity of retrieving images is $O((n - r)(r + 1))$

where	n	:	the number of attributes
	r	:	the number of attributes in query ($1 \leq r < n$)

proof :

First, we assume that concepts whose attribute set size is equal to each other should be in the same level. According to the above assumption, a lattice structure has n levels if the size of the attribute set is n. We find that the number of concepts at level (n-r) is equal to ${}_nC_r$ ($= {}_nC_{n-r}$), where $r = 1, 2, \ldots, n-1$. Also, at level r, the size of the attribute set of each concept is (n-r). Therefore, in order to find a concept when the size of the query attribute set is r, we need to consider concepts from level 1 to level (n-r).

Case 1 : At level 1

(1) The size of the attribute set of each concept at **level 1** equals (n-1) and the number of concepts including the r attribute set necessarily equals ${}_{n-r}C_{n-1-r}$ ($= {}_{n-r}C_1$

= n – r) and we call these concepts Selectable concept[4]s. In other words, (2) the number of concepts, which do not include the r attribute set, equals ${}_{n}C_{1} - {}_{n-r}C_{1} = r$. Therefore, (3) the worst case of finding a concept whose attribute set includes the r attributes is ${}_{n}C_{1} - {}_{n-r}C_{1} + 1 = r + 1$. For example, n = 4 in Figure 19, and suppose a query set is {a} implying r = 1 :

(1) The size of the attribute sets at level 1 in Figure 19 is 3, which is the same as (n – 1) = (4 – 1) = 3.

(2) C14 is the only concept whose attribute set does not have {a} implying that the number of concepts which do not include {a} is one, which is the same as r = 1.

(3) There are three concepts (C11, C12, C13) containing {a} as an element of the attribute set and one concept (C14) not containing {a}. It implies that in the worst case scenario, the number of comparisons required to find a concept whose attribute set contains {a} is two, which is the same as r + 1 = 2.

Case 2 : At level 2

The size of the attribute set of each concept at **level 2** equals to (n-2) and the number of concepts including r attribute set equals ${}_{n-r}C_{n-2-r} = {}_{n-r}C_{2}$. The selectable concepts found in level 1 have ${}_{n-1-r}C_{1}$ (= n – 1 – r) superconcepts whose attribute set includes the r attributes. In other words, the number of concepts, which do not include the r attribute set, equal (n – 1) – (n – 1 – r) = r. Therefore, the worst case

[4] Selectable Concept : A selectable concept is a concept in a lattice structure. An attribute set of selectable concepts includes all elements of the query attribute set. It implies that selectable concepts could be selected to check other concepts, which are superconcepts of a selectable concept at the next level.

scenario for finding a concept whose attribute set includes the r attributes is r + 1.

Case 3 : At level $\geq$ 3

Similarly, the concepts including r attributes at **level n-r-1** have ${}_{n-(n-r-1)-r}C_{n-r-1} = {}_{1}C_{n-r-1} = 1$ superconcepts whose attribute set includes the r attributes at **level n-r**. In other words, the number of superconcepts whose attribute set does not include the r attributes equals (n – (n – r – 1)) – r = r. Therefore, the worst case for finding a concept whose attribute set includes the r attributes is r + 1. Consequently, by applying the inductive method, the number of the worst case comparisons equals (n – r)(r + 1) comparisons except for two comparisons for Intersection and Equality operation. Therefore, the worst case complexity is O((n – r)(r + 1).

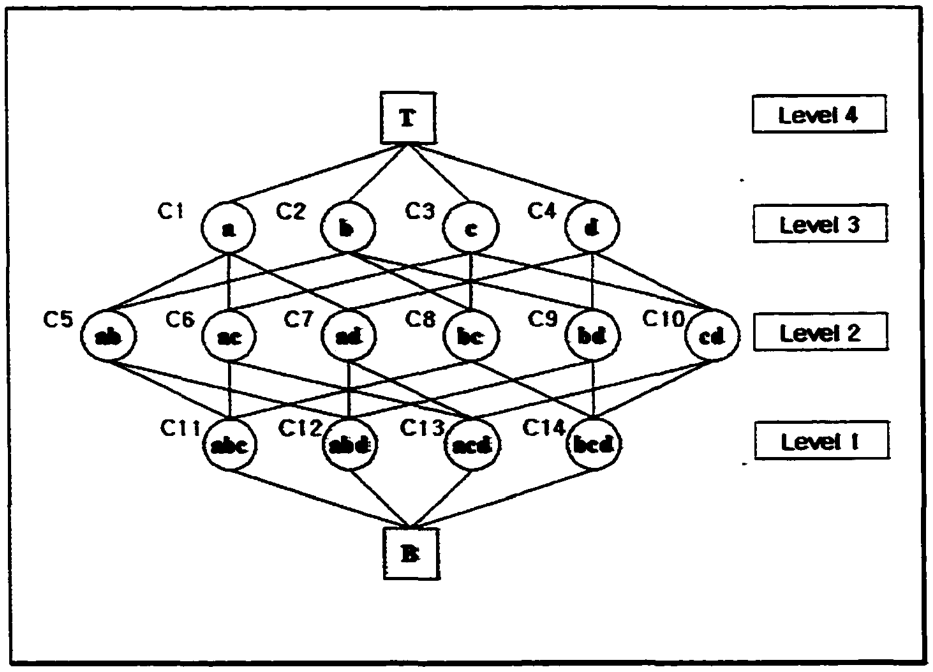


Figure 19. Worst case of a lattice structure with {a,b,c,d}

To demonstrate the Lemma 3, Suppose we have an attribute set {a, b, c, d}. The resultant lattice structure (worst case) is shown in Figure 19. This lattice structure shows all possible combinations of {a, b, c, d}. With the lattice structure and two query attributes ({d}, {c, d}), we demonstrate the time complexity of retrieving images. Because we use a bottom-up search, finding {d} would be the most difficult case in terms of finding a concept with one element of a query attribute set, and {c, d} with two elements of a query attribute set.

Before demonstration of the time complexity, it should be mentioned again that the bit set structure, which requires only 2 comparisons for the intersection and equality operation is used.

Example 1 : Suppose the query attribute set is {d}

From the [B], there are four superconcepts, and for each of these concepts, two comparisons (∩, =) are needed to find the next concept. Furthermore, there is {d} in the concepts C12, C13 and C14. Therefore, only two comparisons are needed to find the next concept in worst case : first, C11 must be checked, and then one of the following concepts (C12, C13, C14), which contains {d}. As a result, we have the following condition :

2(∩, =) × 2(C11, one of {C12, C13, C14}) = 4 comparison

The next concept to be chosen is C12. From the concept C12, there are three superconcepts (C5, C7, C9), but if one looks at the concepts (C7, C9), there is {d} in each concept. Therefore, the following equation applies :

2(∩, =) × 2(C5, one of {C7, C9}) = 4 comparison

From C7 or C9, each concept has two superconcepts, and one of them has {d}, which is the same as the query attribute set. Therefore, the following equation is generated (choosing C7) :

2(∩, =) × 2(C1, C4) = 4 comparison

Consequently, we have the following number of comparisons :

$2 \times 3 \times 2 = 12$ Comparisons

$2 \times (4 - 1) \times (1 + 1) = O((n - r)(r + 1))$,

where **$n = 4$** : the number of attributes.

$r = 1$: the number of attributes in query ($1 \le r < n$).

Example 2 : Let query attribute set {c, d}

From [B], there are four superconcepts, and for each of these concepts, two comparisons (∩, =) are needed to find the next concept. Furthermore, there are {c, d} in the concepts C13 and C14. That means three comparisons are needed to find the next concept in the worst case : first, one would need to check C11, and C12, and then one of the following concepts (C13, C14), which contains {c, d}. Consequently, the following equation applies :

2(∩, =) × 3(C11, C12, one of {C13, C14}) = 6 comparison

The next concept to be chosen is C13. From the concept C13, there are three superconcepts (C6, C7, C10), and if one looks at the concept C10, it contains attributes {c, d}, which is the same as the query attribute set. Therefore, the following equation applies :

$2(\cap, =) \times 3(C6, C7, C9) = 6$ **comparison**

Consequently, the following number of comparisons are obtained :

$2 \times 2 \times 3 = 12$ **Comparisons**

$2 \times (4 - 2) \times (2 + 1) = O((n - r)(r + 1))$,

where $n = 4$: the number of attributes.

$r = 2$: the number of attributes in query ($1 \leq r < n$).

From the above two examples, we can derive the following equations :

$2 \times (4 - 1) \times (1 + 1)$, when size of query attribute is one

$2 \times (4 - 2) \times (2 + 1)$, when size of query attribute is two

$2 \times (4 - 3) \times (3 + 1)$, when size of query attribute is three.

...

Then by substitution $n = 4$ and $r = 1, 2, 3, \ldots$

$2 \times (n - r) \times (r + 1)$, when number of query attributes is one

$2 \times (n - r) \times (r + 1)$, when number of query attributes is two

$2 \times (n - r) \times (r + 1)$, when number of query attributes is three

...

where n is the number of attributes

r is the number of attributes in a query.

Therefore, the time complexity of retrieving images is :

$O(n) = (n - r)(r + 1)$, where

n	:	the number of attributes
r	:	the number of attributes in a query

5. Implementation

In order to demonstrate concepts proposed in this thesis, we have built an application using various buildings in the world. Around 460 images are collected from "The GREAT BUILDINGS COLLECTION" CD-ROM from greatbuildings.com with conditional permission. The next few sections provide details of implementation including the process required to collect images, classification of the attribute set, explanations of images and lattice structure, results of experiments and discussion about the User-Interface. The program was developed on a Celeron 450MHz computer system running MS Window98 using Java programming language.

5.1. Collecting Images

For demonstration purposes, images with text annotations are needed. We chose "The Great Buildings Collection" CD-ROM because it has an accurate and detailed text annotation of images as shown in Figure 20. To use images in the CD-ROM, the permission from www.greatebuildings.com was needed. The letter from www.greatebuildings.com is attached in Appendix A. Because of conditional permission, none of these images are used in this manuscript – only in the application.

The attribute set used in this thesis includes five categories with each category consisting of nine sub-categories. Consequently, 45 attributes as elements of attribute set are introduced. The five categories and 45 attributes are :

- **Place :**
  - Europe : Austria, Belgium, France, Germany, Greece, Italy, etc.
  - Scan : Denmark, Finland, Norway, Sweden, Russia, Spain, Switzerland, etc.
  - UK : England, Scotland
  - A : Canada, Mexico, etc.
  - USA : United States of America
  - Oce : Australia, New Zealand, etc.
  - ME : Egypt, Israel, Syria, Turkey, etc.
  - Asia : China, India, Japan, Korea, etc.
  - Afr : Ethiopia, North Africa, etc.

- **Building Types :**
  - Home : Small House, Large House, Multi-family House, Villas, etc.
  - Pray : Cathedral, Church, Mosques, Temple, Monastery, etc.
  - Park : Park, Garden, etc.
  - Castle : Castle, Palace.
  - Exhibition : Art Galleries, Exhibition, Exposition, Museum, Theater, etc.
  - Commercial : Bank, Commercial building, Factory, Office, Hotel, etc.
  - Airport : Airport teminal.
  - Public : Government building, City Hall, etc.
  - School : School, Academic, Library, etc.

- **Climates :**
  - Desert : Desert

- Alpine : Alpine
- Temp : Temperate
- MT : Mild Temperate
- CT : Cold Temperate
- WT : Warm Temperate
- HT : Hot Temperate
- ST : Subtropical
- Tro : Tropical

• **Construction Types :**

- BM : Bearing Masonry
- Brick : Brick
- Con : Concrete
- FT : Fabric & Tensile
- Geo : Geodesic
- Glass : Glass
- WF : Wood Frame
- Steel : Steel
- Tim : Timber

• **Contexts :**

- Camp : Campus Context
- Hill : Hill or Cliffside
- Moun : Mountain Context

- River : Riverside, Waterfront, etc.
- Rural : Rural
- TC : Small Town or City
- Sub : Suburban
- Urban : Urban
- Vill : Village Context

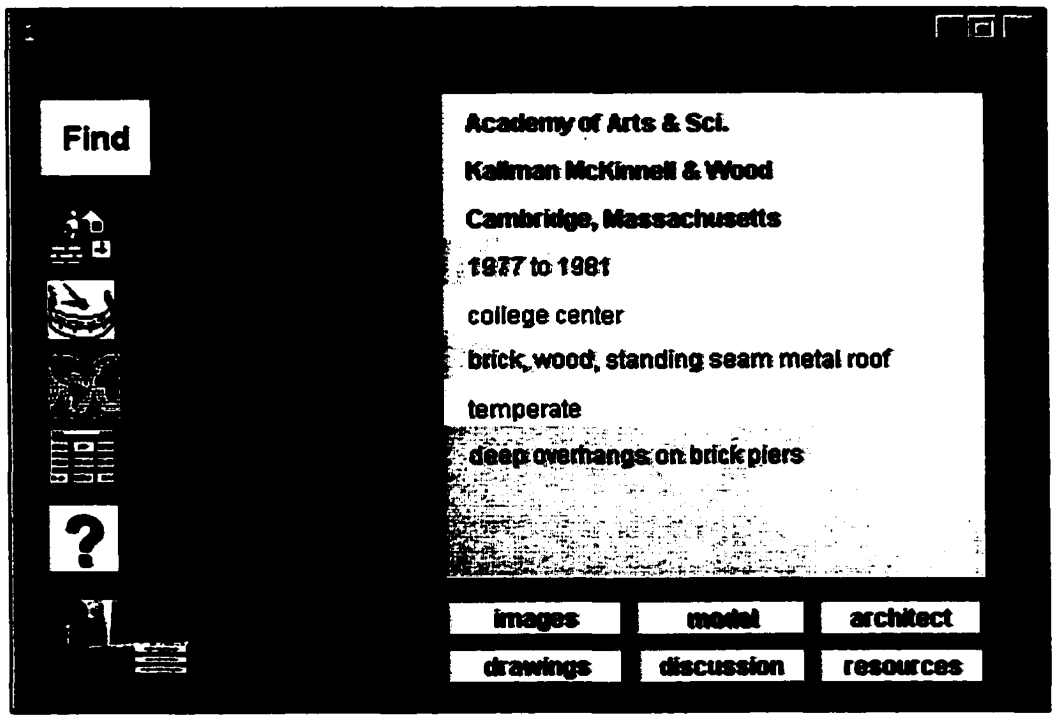

Academy of Arts & Sci.
Kallman McKinnell & Wood
Cambridge, Massachusetts
1977 to 1981
college center
brick, wood, standing seam metal roof
temperate
deep overhangs on brick piers

Figure 20. Text annotation of a building in CD-ROM.

5.2. Data Structures

In this thesis, we use two different databases : one for storing a lattice structure and the other for storing information about images. A lattice data structure is a collection of concepts and each concept consists of five parts i.e., name of concept, attribute set, object set, superconcepts and subconcept. Image data structure consists of three parts i.e., building name, building style and attribute set. Both of these data structures are needed during execution of the program.

5.2.1. Data Structure for Lattice.

The lattice structure consisting of concepts contains five parts :

- name of concept (represented as number)
- attribute set (represented as 0 or 1)
- object set (represented as image file names)
- superconcepts (represented as concept names)
- subconcepts (represented as concept names)

Each part is delimited by "/", and each element of the object set, superconcepts, and subconcepts is delimited by "|". For example, the concept structure as shown in Table 1 is obtained after addition of 50 images and represents **Concept 2**, where $\boxed{2}$ represents the name of the concept, $\boxed{\text{0000000000000000000 010000000000000000000000010}}$ represents the attribute set of this concept. The unboxed area represents the object set consisting of the name of the actual image file. Each name of real image file is delimited by "|". $\boxed{8|14}$ represents the name of the superconcepts, and the last part ($\boxed{24|25|38|48|49|71|85|109|126|151|182}$) represents the name of the subconcepts and also use "|" as delimiter marker.

With this concept structure, we can construct a lattice structure. The lattice structure is a collection of the concept structures. For example, if we have 100 concepts in a lattice structure, 100 concept data structures are saved in the lattice structure.

```
2/0000000000000000000000100000000000000000000000010/[0001]88WoodStreet.jpg|
[0002]AbteiburgMuseum.jpg| [0003]AlexandraRoadHousing.jpg| [0004]AltesMuseum
.jpg| [0010]Baltimore-OhioRailroadDepots.jpg| [0013]BanquetingHouse.jpg| [0016
]Bauhaus.jpg|  [0020]BostonCityHall.jpg| [0022]BrooklynBridge.jpg| [0027]Castel
Beranger.jpg| [0030]CentrePompidou.jpg| [0031]ChartresCathedral.jpg| [0032]Chase
ManhattanBank.jpg| [0033]ChateauDeVersailles.jpg| [0037]ChristianScienceCenter.jp
g| [0038]ChryslerBuilding.jpg| [0040]CircusAtBath.jpg| [0041]CiticorpCenter.jpg| [0
046]CrystalPalace.jpg/8|14/24|25|38|48|49|71|85|109|126|151|182
```

Table 1. Representation of Concept 2 in a Lattice Data Structure

5.2.2. Data Structure for Image.

The image data structure consists of three parts.

- building name
- building style
- attribute set (image file name)

The symbol "/" is used as a delimiter to separate three parts. For example, Table 2 is a part of [image.stoney] file, which is used as the image database.

```
88 Wood Street/High-Tech Modern/0010000000000001000001000000000100101000000000010
Abteiburg Museum/ Post-Modern/1000000000000010000000100000000100000000000000000010
......
```

Table 2. Part of the Image Data Structure

In this data structure, 88 Wood Street represents the building name in an image. High-Tech

Modern represents building style and 001000000000001000001000000001001010000000010 represents the attributes of the image given as a bit set.

5.3. Addition of Images

We have collected CPU time for addition of images in the database. Since a different order of insertion could lead to a different number of concepts, different insertion time and different number of levels as shown in Table 3, Table 4 and Table 5 respectively, four different orders and collected information about the average number of concepts as well as the number of levels and average insertion time are inserted as shown in Table 6. The four different orders are :

- Random 1 (R1) : this input consists of two groups ordered by building name. After the first group is inserted, the second group is inserted.
- Random 2 (R2) : all images are ordered by the first nine bits of an attribute set.
- Random 3 (R3) : this input consists of two groups ordered by the first nine bits of attribute set.
- Random 4 (R4) : all images are placed order in reverse to R2.

As one would expect, if there are more concepts in the lattice structure, more time will be needed to add images because more possible concepts are found by checking the existing concepts in the lattice structure. However, in some case, less time is required to add images, even though more concepts are in a lattice because of the number of generated possible concepts. Consequently, the main factor in CPU time to add images are (a) the number of concepts in the lattice structure and (b) the number of generated possible new concepts. Average insertion time as a function of images for different insertion orders is shown in Figure 21.

Number of images	Number of concepts (R1)	Number of concepts (R2)	Number of concepts (R3)	Number of concepts (R4)
0	0	0	0	0
50	208	98	98	199
100	442	200	354	396
150	635	507	590	594
200	810	712	823	717
250	977	949	997	889
300	1170	1169	1201	1094
350	1345	1316	1366	1355
400	1502	1437	1514	1498
450	1638	1638	1638	1638

Table 3. Number of concepts generated with four different orders

Number of images	Insertion time of R1 (ms)	Insertion time of R2 (ms)	Insertion time of R3 (ms)	Insertion time of R4 (ms)
0	0	0	0	0
50	1970	720	700	1370
100	10320	2970	8170	9660
150	30420	15330	25580	31470
200	59430	46520	59800	62610
250	100460	82000	102200	93920
300	157310	143580	161130	141760
350	221960	236080	235270	201360
400	311920	344010	345170	270840
450	399360	412330	434530	357460

Table 4. Insertion time with four different orders

Number of levels	3	4	5	6	7	8	9	10
Number of Images added (R1)	1		2	3,4	5	6	7-48	49-
Number of Images added (R2)	1	2		3-5	6,7	8-26	27-165	16-
Number of Images added (R3)	1	2		3-5	6,7	8-26	27-80	81-
Number of Images added (R4)	1	2		3,4	5	6-8	9-64	65-

Table 5. Number of levels with four different orders

Number of Inserted Images	Average Number of Concepts	Average Insertion Time (millisecond)	Number of Levels in Lattice
0	0.00	0.00	2
50	150.75	1190.00	9
100	348.00	7780.00	10
150	581.50	25700.00	10
200	765.50	57090.00	10
250	953.00	94645.00	10
300	1158.50	150945.00	10
350	1345.50	223667.50	10
400	1487.75	317985.00	10
450	1638.00	400920.00	10

Table 6. Average result from Table 3, Table 4 and Table 5

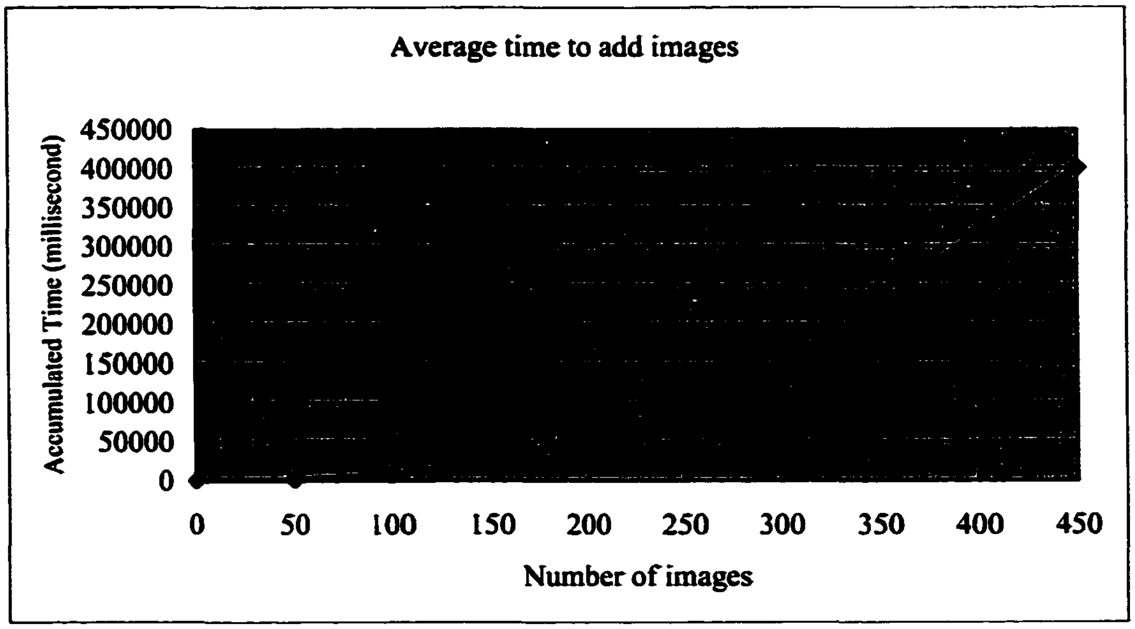


Figure 21. Average time to add images.

Information about the number of concepts as a function of the number of images has also been collected. As one would expect, the number of concepts increases with the number of images. However, with the sample image set we have in our database and the total number of concepts (2^{45}), the increase in the number of concepts in this case is nearly linear. A different ordering or domain could give rise to totally different. The number of concepts generated as a function of images is shown in Figure 22.

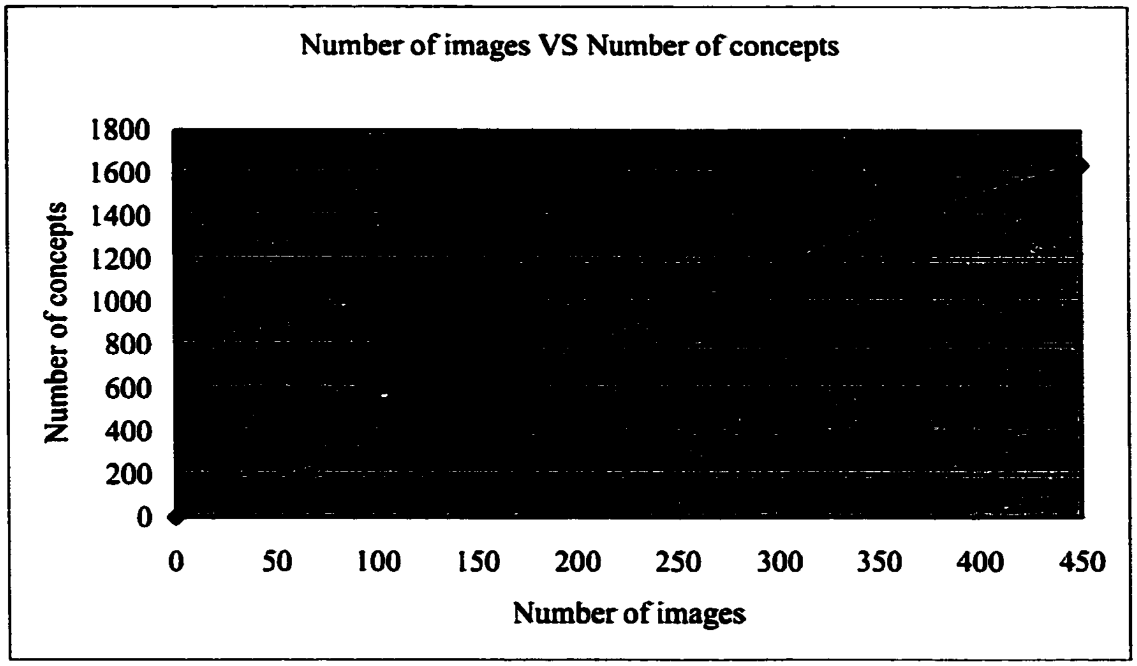


Figure 22. Number of Image VS Number of Concepts.

The last experiment result in this section concerns insertion time as a function of the number of attributes. For this experiment, 100 images are added into a lattice structure with a different number of attributes as shown in Table 7. As mentioned earlier in Section 3.2, the addition method depends on the number of attributes and its time complexity is $O(2^n)$. Therefore, the increment of insertion time is exponential as shown in Figure 23.

Number of Attribute	5	10	15	20	25	30	35	40	45
Insertion Time (ms)	122	196	410	588	1330	3470	4780	5534	12794

Table 7. Insertion time with different number of attributes

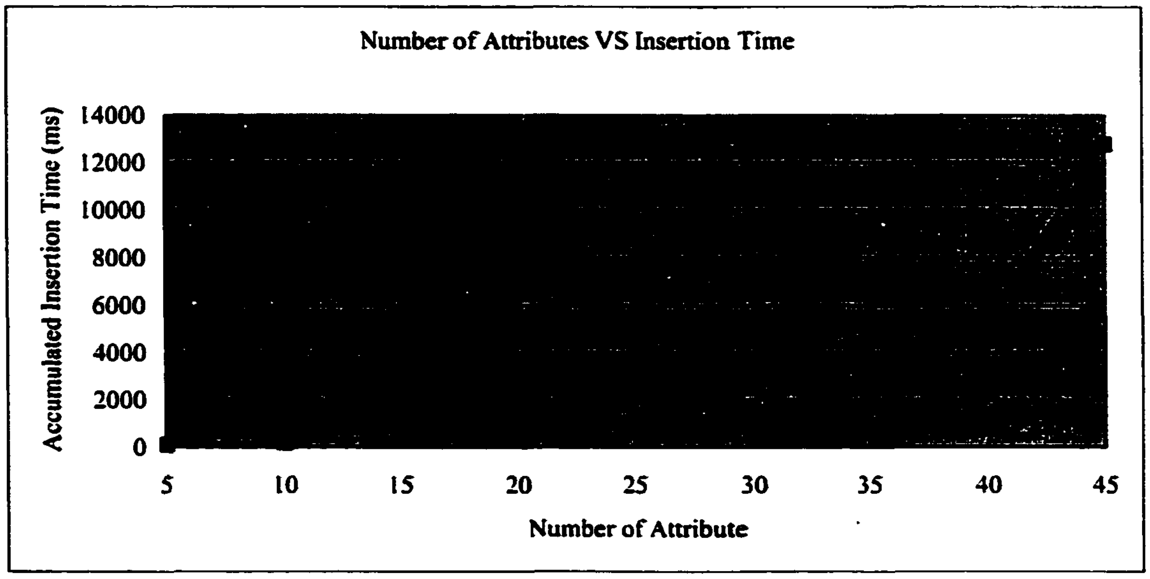


Figure 23. Number of Attributes VS Insertion Time

5.4. Experimental Result for Image Retrieval

In this section, the retrieval time of this system is compared to that of the exhaustive search method. It is assumed that finding concepts whose size of attribute set is one take more time than finding other concepts with a higher number of attributes because in this case such concepts are at higher levels (from bottom) of the lattice structure. Therefore, in this system we need to traverse more levels in the lattice structure to find concepts whose size of attribute set is one. The program for image retrieval is tested on a SUNW, Ultra-Enterprise running on SunOS Release 5.7.

Results of sample image retrievals are given in Table 8. The "attribute name" in Table 8 is a query attribute. In this case, we do not present all 45 attributes because some of the attributes may give us similar results as with others. The "number of images in this concept" refers to a concept whose attribute set is the same as "attribute name" and which contains an object set whose size is the "number of images in this concept". The "path from bottom" shows a path from bottom node to a concept whose attribute set is the same as "attribute name". The numbers represent concept numbers and the last one is the concept number whose attribute set is the same as "attribute name". The "Exhaustive search" shows the CPU time to retrieve images using a method, which uses an exhaustive search. The "Lattice search" shows the CPU time using our system.

Attribute name	# of images in this concept	Path from bottom	Exhaustive Search (ms)	Lattice Search (ms)
Temperate	261	0 – 1 – 4 – 38 – 2 – **8**	101.333	1.333
USA	176	0 – 13 – 661 – 55 – 171 – 54 – **29**	96.000	2.666
House	142	0 – 13 – 356 – 138 – 40 – 11 – **20**	58.666	1.666
Rural	95	0 – 13 – 356 – 663 – 212 – 160 – **167**	45.333	1.333
Suburban	93	0 – 13 – 356 – 138 – 137 – 61 – **64**	28.000	1.666
Pray place	75	0 – 147 – 143 – 144 – 150 – **164**	22.333	1.000
Commercial	67	0 – 1 – 362 – 369 – 367 – 365 – **366**	22.666	1.666
Exhibition	55	0 – 3 – 6 – 182 – 117 – **257**	17.000	1.000
School	52	0 – 52 – 297 – 50 – **290**	16.666	1.333
Scandinavia	39	0 – 94 – 407 – 309 – 101 – **188**	12.666	1.333
Timber	31	0 – 67 – 504 – 186 – 187 – **392**	10.333	1.666
Park	6	0 – 645 – 646 – 650 – **877**	6.000	1.666
Airport	3	0 – 219 – 1556 – **837**	3.000	1.000

Table 8. Retrieval time comparison between exhaustive and lattice search

For example, suppose we look for building images, which are in "Rural". A concept whose attribute set is only "Rural" is concept 167 and this concept contains 95 images as an object set. This concept is obtained by starting from concept 0 (bottom node), concept 13, concept 356, concept 663, concept 212, concept 160 and finally concept 167. All these appeared as concept numbers (13, 356, 663, 212, 160) except for 167 contains not only "Rural", but also other attributes as elements of an attribute set. Finally, it takes 45.333 milliseconds to retrieve images if an exhaustive search is used. However, it takes 1.333 milliseconds if we use our lattice structure system.

Another result in this section concerns retrieval time as a function of the number of images added into a lattice structure. As mentioned earlier, the retrieval time depends on the number of objects in images rather than the number of images. As an experiment, "Europe" is chosen as a query attribute and the retrieval time is checked every time 100 images are added into the lattice structure as shown in Table 9. Consequently, the number of images does not affect image retrieval time because the hierarchical path in a lattice structure does not change, even though the number of concepts is increased as shown in Table 6.

Number of Images	**100**	**200**	**300**	**400**
Retrieval Time(ms)	1.333	1.666	1.333	1.333
Concept Path	0 – 3 – 6 – 49 – 15 – 63			

Table 9. Retrieval time as a function of number of images

5.5. User Interface

User interface consists of two main parts: one is for addition of images [Add Image]; while the other one is for retrieval of images [Search Images]. To add and retrieve images, the user interface provides icons to users. In [Add Image], when a user loads an image, the interface asks the user "building name" and "building style" then the user can set required attribute set by using icons. In [Search Image], the user can retrieve images after clicking the icons. The user interface also provides details of images. Each part that shows the user how to operate the interface will be explained with captured images.

5.5.1. Introductory Screen

In order to start, the user interface provides an initial screen with two main options :

- [Add Image] or
- [Search Images].

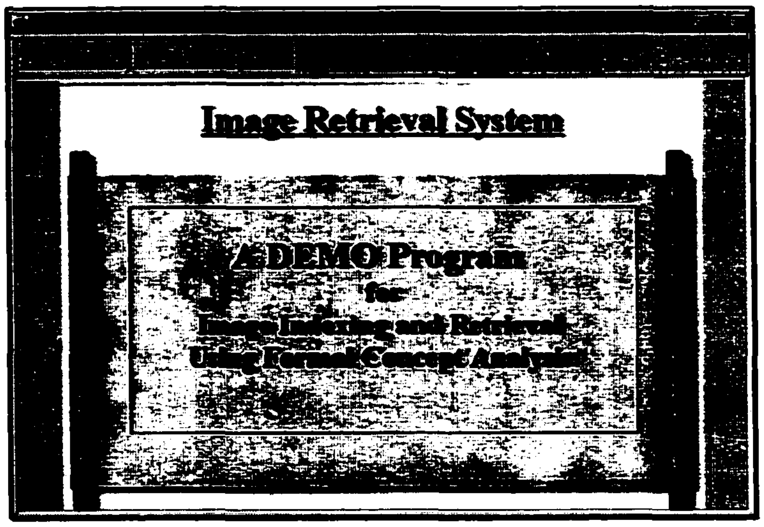


Figure 24. Initial screen of User Interface.

5.5.2. Image Addition.

In the [Add Image] option, the following steps, which will be explained with captured images, are provided :

- Select an image to be added
- Enter a name of building
- Enter building style
- Set attributes using icons
- Add the image

Initial screen of [Add Image]

When a user clicks the [Add Image] button in the initial screen (Figure 24), the user interface provides a new screen as shown in Figure 25, providing a brief instruction of how to add an image on the center of the user interface.

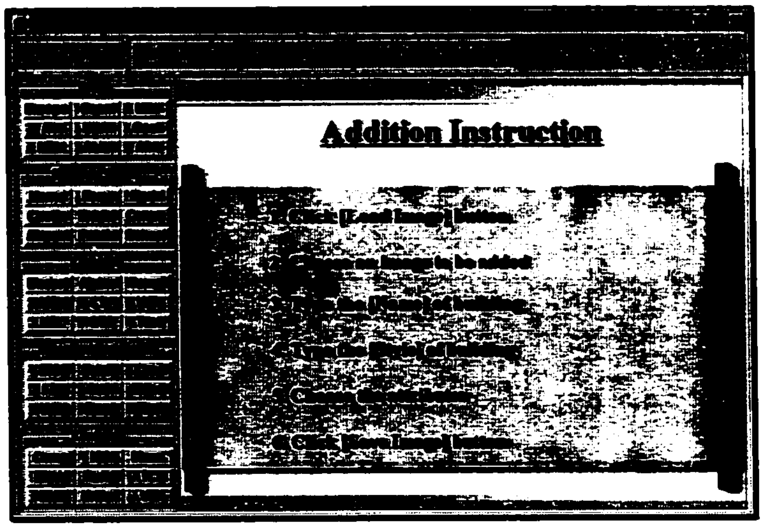


Figure 25. Initial screen of [Add Image].

Select an image file

In this step, we select an image file to add into the lattice structure as shown in Figure 26. Initially, the user interface provides a file filter for GIF, BMP, and JPG files. After the user chooses an image, the user interface will show the screen as shown in Figure 27.

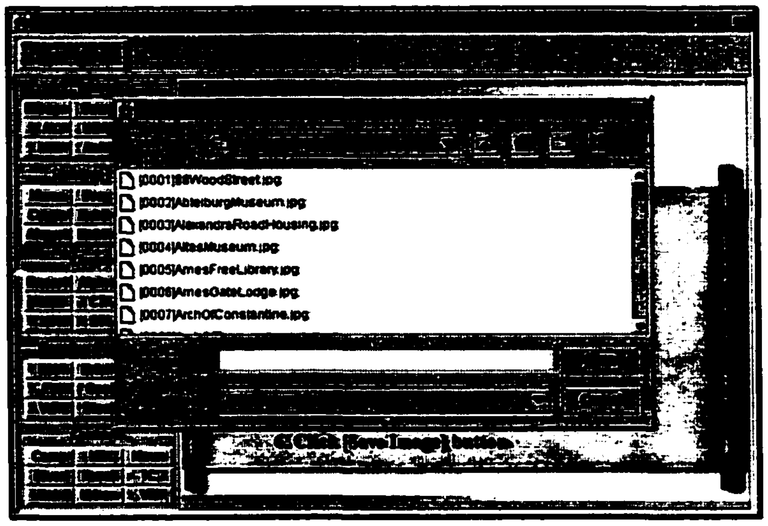

|0001|88WoodStreet.jpg
|0002|AbteiburgMuseum.jpg
|0003|AlexandraRoadHousing.jpg
|0004|AltesMuseum.jpg
|0005|AmesFreeLibrary.jpg
|0006|AmesGateLodge.jpg
|0007|ArchOfConstantine.jpg
Click [New Image] button

Figure 26. Select an image file.

Enter a building name

After users choose an image, the user interface shows the selected image and asks the user the building name as shown in Figure 27. This information is used in the image retrieval part.

Enter a style of building

The user interface also asks users to enter the style of building. Examples of styles are "Modern", "Post-Modern", and "Renaissance" to name a few. This information is also used in retrieval images part. The captured image for this step is shown in Figure 28.

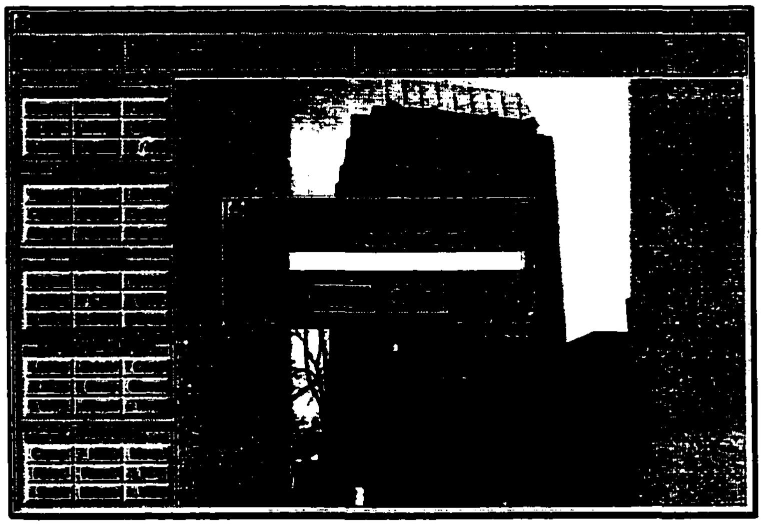


Figure 27. Enter a building name.

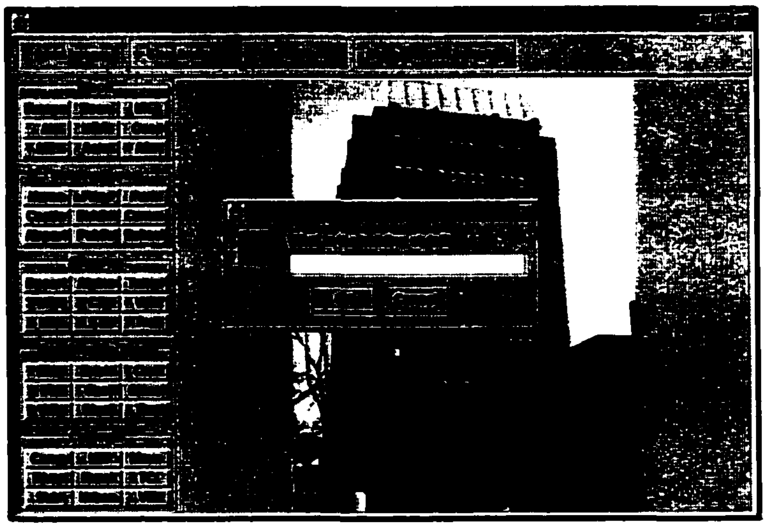


Figure 28. Enter a building style.

Choose attributes of image

After entering information (name of building, style of building), the user needs to set the attributes of the image by using text icons on the left side of the user interface. As explained earlier, there are five categories (place, building type, climate, construction type, and context). For each category, the user can choose more than two attributes. An example of a captured image for this step is shown in Figure 29.

Figure 29. Choose attributes.

Adding the image

To add an image with this information and the attributes set into the lattice structure, the user simply clicks the **[Save Image]** button. This allows the system to add an image into the lattice structure and the control returns to the initial screen ([Add Image]).

5.5.3. Image Search

To retrieve an image, the following steps are provided and will be explained with the help of captured images :

- Set the attributes as a query attribute set
- Retrieve images

- Get more details of an image
- No images in DB

Initial Screen of [Search Images]

This option allows a user to search and retrieve images from the database. On the initial screen, users are shown a searching instruction. When a user clicks the [Search Image] button, a new screen as shown in Figure 30, is provided to the user with additional search options and criteria.

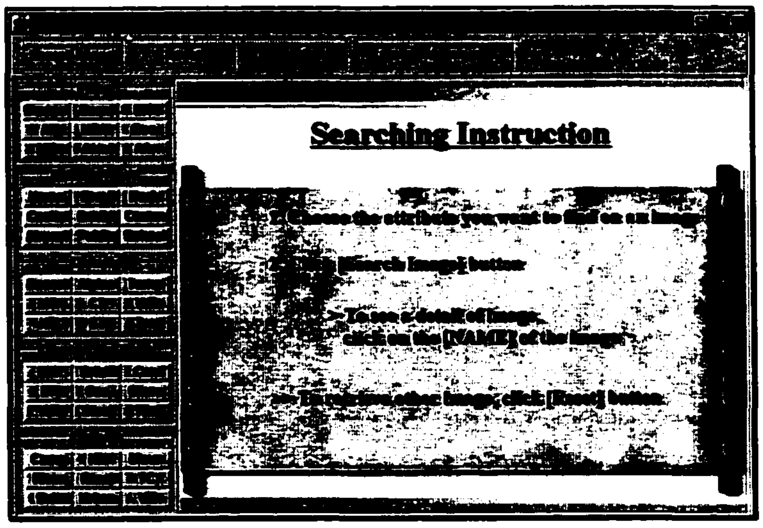
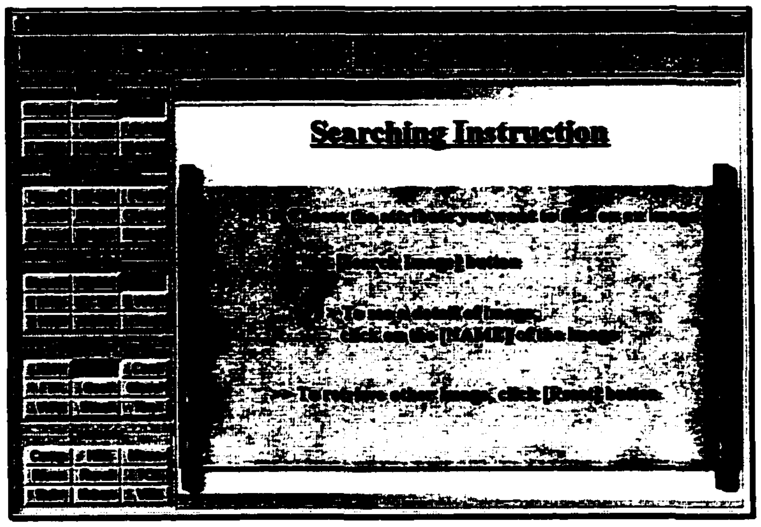


Figure 30. Initial screen of [Search Images].

Set the attributes

In this step, users set the attributes as shown in Figure 31, and the attribute set is used as the query attribute set. After setting the attributes, if the user clicks on the **[Search Image]** button, the user interface shows all of the retrieved images satisfying the criteria given in the user query.


Searching Instruction

[illegible]

[Search Image] button

To see detail of image, click on the [NAME] of the image

To retrieve other images, click [Reset] button.

Figure 31. Set Attributes.

Retrieved images

After submitting a query, the user interface shows thumbnails of the retrieved images along with names of buildings in those thumbnails satisfying criteria given in query. In order to get more specific information about a building, the user can click the name label of the building. Sample retrieved images are shown in Figure 32.

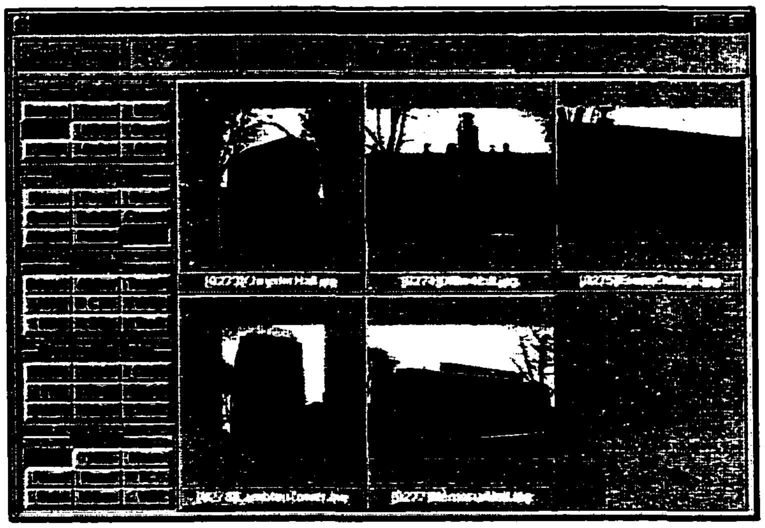


Figure 32. Retrieved images.

Specific information

By clicking on the name of a building in any of the retrieved images, the user can get more specific information about a building in another window, as shown in Figure 33.

Figure 33. Specific information.

No images message

The system fails to find any match in the database satisfying criteria given in the user query; then the system informs the user by printing a NO Images message because in our system, an exact matching scheme is used. However, there is another possible scheme called a ranking scheme. It will show users images by rank and the rank depends on the attribute. Instead of using the ranking scheme, our system allows the user to provide a new attribute set.

6. Conclusion and Future Work

On the one hand, the time complexity for image retrieval generally depends on the number of images. However, in the scheme presented in this thesis, time complexity depends on the number of attributes. On the other hand, we need to fix the size of an attribute set making it applicable to a specific predefined application domain, but offer a significant reduction in retrieval time. Furthermore, the annotation is done manually in this system, but if a method in which segmentation and semantic information can be done automatically is applied to our system, our system will be more efficient and powerful.

FCA is generally applied to only these systems, which demands a reasonable amount of data because of the cost involved to rebuild the whole lattice structure. This thesis introduces 'addition method' to rebuild a lattice structure in FCA. With the addition method, we could reduce the cost to rebuild a lattice structure. This method is not yet perfect, but would be improved by reducing the redundancy of possible concepts. This addition method only rebuilds a part of a lattice structure instead of rebuilding a whole lattice structure (Figure 34 and Figure 35). Use of FCA helps us in reducing the image retrieval time. This scheme could be applied to a number of different application domains. Examples of such domains include but are not limited to family photo albums, searching real objects in a specific part (Buildings, Flights, etc.) to name a few.

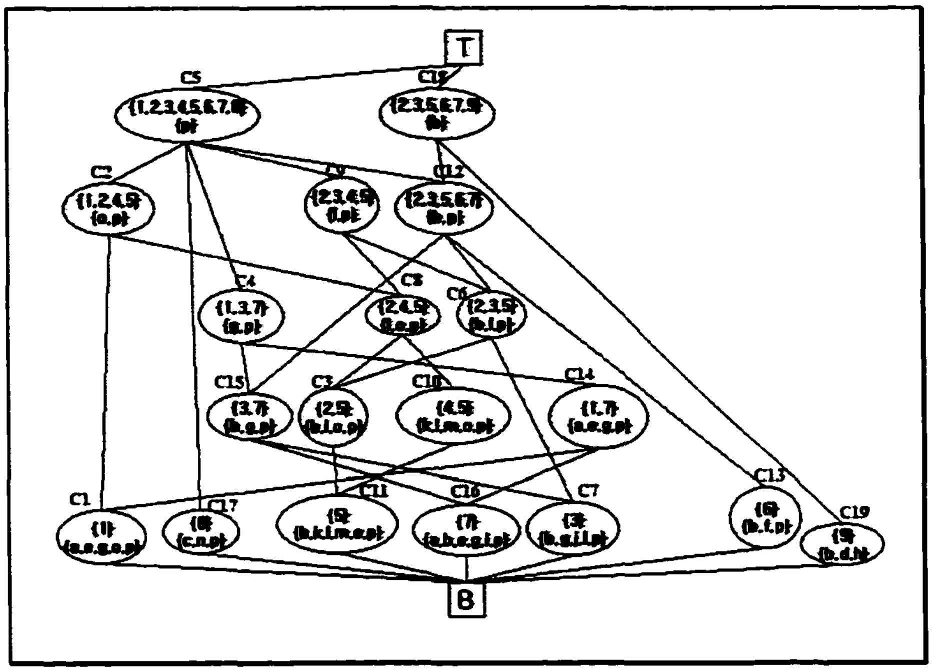


Figure 34. After adding 9 images

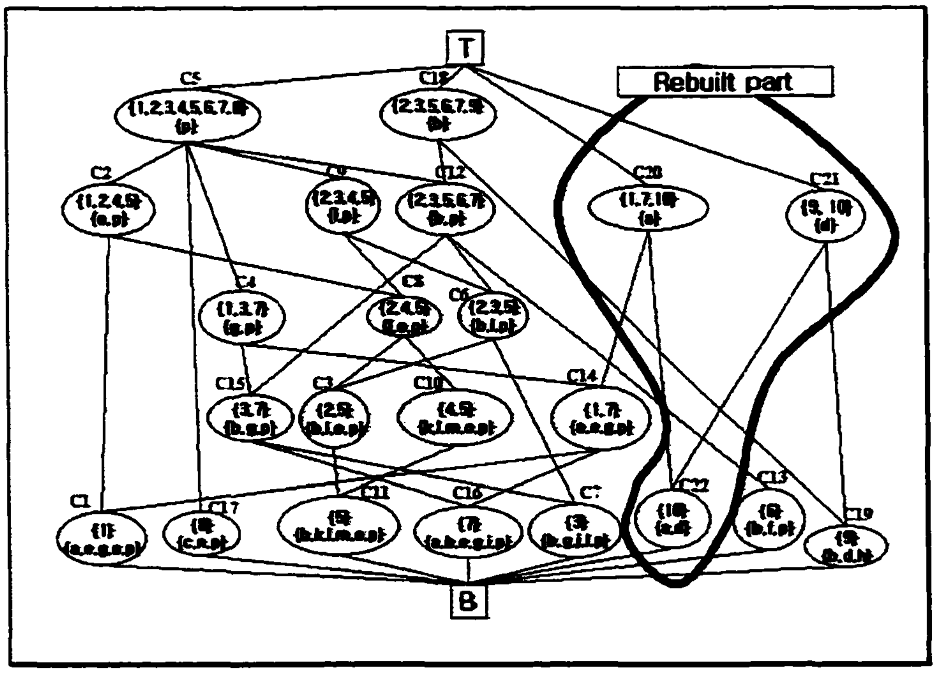


Figure 35. After adding 10th images.

The scheme presented in this thesis can be enhanced further in the following ways :

- **Reduction in time for addition of objects** : As mentioned earlier, there are redundant attribute sets and empty sets when we compare the attribute set of new object with the attributes sets in an existing lattice structure. A mechanism to deal with such sets could reduce the number of comparisons and effectively increase the efficiency of the 'addition method'. For example, suppose one finds possible new concepts from bottom to top. If a subconcept gives rise to an empty possible concept then superconcepts of the subconcept will yield empty possible concepts, so the superconcepts do not need to be checked.
- **A method for deleting objects** : In this thesis, only an addition method is introduced for addition of new objects in a lattice structure. However in some cases, it may also need to remove an object from the lattice structure. The introduction of a deletion method could make this system applicable in environments, which require frequent dynamic changes.
- **Methods for adding and deleting of attributes** : One of the limitations of this scheme involves the use of a fixed size of attribute set. If we change the size of a attribute set, we need to rebuild the whole lattice structure, which is an expensive operation. Therefore, methods for addition and deletion attributes would give us chances to expand this scheme more effectively.

REFERENCES

[1] I. Ahmad and W. I. Grosky. *Image Indexing and Spatial Similarity-Based Retrievals.* In Proceedings of the Workshop on Handling Image Data in Multimedia Database Systems, Bulgaria, pp. 96-129, Sofia,June 1997.

[2] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C.F. Shu. *The Virage image search engine : An open framework for image management.* In Proceedings of IS&T/SPIE Symposium on Electronic Imaging : Science and Technology – Storage & Retrieval for Image and Video Databases IV, San Jose, CA, Vol. 2670, pp. 76-87, February 1996.

[3] L. Bielski. *20 million photos will be digitized at Time Warner :The image database of the future begins.* Advanced Imaging, Vol. 10, No. 10, pp. 26-28.91, October 1995.

[4] R. Brunelli, and O. Mich. *Image Retrieval by examples.* IEEE Transactions on Multimedia, Vol3, No2, pp. 164-171, 2000.

[5] C.C. Chang and S.Y. Lee. *Retrieval of Similar Pictures on Pictorial Databases.* Pattern Recognition, Vol. 7, No. 24, pp. 675-680, 1991.

[6] E. Chang, B. Li, et al. *Towards Perception-Based Image Retrieval.* In Proceedings of IEEE Workshop on Content-based Access of Image and Video Libraries, South Carolina, pp. 101-105, June 2000.

[7] S.K. Chang and A. Hsu. *Image Information Systems : Where Do We Go From Where?* IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 4, pp. 431-442, 1992.

[8] S.K. Chang, Q.Y. Shi, and C.W. Yan. *Iconic Indexing by 2-D Strings.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 3, No. 9, pp. 413-428, 1987.

[9] G. Costagliola, M. Tucci, and S.K. Chang. *Representing and retrieving symbolic pictures by spatial relations.* In E. Knuth and L.M. Wegner, editors, Visual Database Systems, II, The IFIP TC2/WG2.6, Second Working Conference on Visual Database Systems, pp. 49-59, 1992.

[10] D. Daneels, D. Campenhout, W. Niblack, W. Equitz, R. Barber, E.Bellon, and F. Fierens. *Interactive outlining : An improved approach using active contours.* SPIE Storage and Retrieval for Image and Video Databases, Vol. 1908, pp. 226-233, 1993.

[11] J. Drakopoulos and P. Constantopoulos. *An Exact Algorithm for 2-D String Matching.* Technical Report 021, Institute of Computer Science, Foundation for Research and Technology, 1989.

[12] W. Equitz and W. Niblack. *Retrieving images from a database using texture – algorithms*

from the QBIC system. Technical Report, RJ 9805, Computer Science, IBM Research Report, May 1994.

[13] C. Faloutsos, H.V. Jagadish, Alberto O. Mendelzon, and T. Milo. *Signature technique for similarity-based queries.* Technical Report 112530-951110-16TM, At&T Bell Labs, 1996.

[14] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. *Query by Image and Video Content : The QBIC System.* IEEE Computer, Vol9. No. 28, pp. 3-32, 1995.

[15] C. Frankel, M.J. Swain, and V. Athitsos. *WebSeer : An Image Search Engin for the World Wide Web.* Technical Report 96-14, University of Chicago, August 1996.

[16] B. Ganter and R. Wille, *Formal Concept Analysis : Mathematical Foundations.* Springer-Verlag, Berlin-Heidelberg-New York, 1999.

[17] J. Gehring and A. Reinefeld. *MARS – A Framework for minimizing the job execution time in a metacomputing environment.* In Proceedings of Future Generation Computing Systems, 1996.

[18] Abby A. Goodrum, *Image Information Retrieval : An Overview of Current Research.* Special Issue on Information Science Research, College of Information Science & Technology, Drexel University, Vol. 3, No. 2, 2000.

[19] W.I. Grosky and Z. Jiang. *A Hierarchical Approach to Feature Indexing.* In Albert A. Jamberdino and Wayne Niblack, editors, Image storage and retrieval systems, SPIE. University of Virginia Technical Report CS-94-40 12, Bellingham, Washington, pp. 9-20, February 1992.

[20] V.N. Gudivada and V.V. Raghavan. *Content-Based Image Retrieval System.* IEEE Computer, Vol. 28, No. 9, pp. 18-22, 1995.

[21] A. Gupta and R. Jain. *Visual information retrieval.* Communications of the ACM, Vol. 5, No. 40, pp. 69-79, 1997.

[22] N. Herodotou, K.N. Plataniotis, and A.N. Venetsanopoulos. *A Content-based Storage and Retrieval Scheme for Image and Video Databases.* In Proceedings of Visual Communications and Image Processing '98, SPIE Vol. 3309, Part II, pp. 697-708, 1998.

[23] J.N. David Hibler, C.H. Leung. and K.L. Mannock and N.K. Mwara. *A System for Content-Based Storage and Retrieval in an Image Database.* In Image Storage and Retrieval Systems, SPIE'92, San Jose, California, pp. 80-92, 1992.

[24] F.J. Hsu, S.Y. Lee, and B.S. Lin *2D C-Tree Spatial Representation for Iconic Image.* Journal of Visual Languages and Computing, Vol. 12, No. 10, pp. 147-164, 1999.

[25] P.W. Huang and Y.R. Jean. *Using 2D C^{+}-Strings as Spatial Knowledge Representation*

for Image Database Systems. Pattern Recognition, Vol. 27, No. 9, pp. 1249-1257, 1994.

[26] T. Kato. *Database Architecture for Content-Based Image Retrieval.* In Proceedings of SPIE : Image Storage and Retrieval Systems, pp. 112-123, 1992.

[27] B.S. Kim. Advanced Web Search Based On Formal Concept Analysis. MSc Thesis Defense, School of Computer Science, University of Windsor, 2001

[28] P. Kofakis and S.C. Orphanoudakis. *Image Indexing by Content.* In M. Osteaux et. al., editor, A Second Generation PACS Concept, Springer-Verlag, chapter 7, pp. 250-293, 1992.

[29] J. Larish. *Kodak's still picture exchange for print and film use.* Advanced Imaging, Vol. 4, No. 10, pp. 38-39, 1995.

[30] D Lee, R. Barber, W. Niblack, M. Flickner, J. Hafner, and D. Petkovic. *Indexing for complex queries on a query-by-content image database.* 12th IAPR International Conference on Pattern Recognition, pp. 142-146, 1994.

[31] S.Y. Lee, M.K. Shan, and W.P. Yang. *Similarity Retrieval of Iconic Image Database.* Pattern Recognition, Vol. 22, No. 6, pp. 675-682, 1989.

[32] J. Lu, R. Algazi, and Robert R. Estes. *Comparison of Wavelet Image Coders Using the Picture Quality Scale (PQS).* In Proceedings of the SPIE, Vol. 2491, pp. 51-60, April 1995.

[33] Y. Lu et al. *A Unified Framework for Semantics and Feature Based Relevance Feedback in Image Retrieval Systems.* In Proceedings of ACM MM2000, pp. 31-38, 2000.

[34] W. Ma and B.S. Manjunath. *NETRA : A toolbox for navigating large image database.* Multimedia Systems, Vol7, No. 3, pp.184-198, 1999.

[35] M. Martucci. *Digital still marketing at PressLink.* Advanced Imaging, Vol. 4, No. 10, pp. 34-36, 1995.

[36] J. McCarthy. *Metadata management for large statistical database.* In the international Conference on Very Large Data Bases, pp. 470-502, Mexico City, 1982.

[37] F. Mokhtarian, S. Abbasi and J. Kittler. *Efficient and Robust Retrieval by Shape Content through Curvature Scale Space.* In Proceedings of International Workshop on Image Databases and MultiMedia Search, Amsterdam, The Netherlands, pp. 35-42, 1996.

[38] W. Niblack, R. Barber, and et al. *The QBIC project: : Querying images by content using color, texture and shape.* In Proceedings of SPIE Storage and Retrieval for Image and Video Databases, Vol. 1908, pp. 173-187, 1994.

[39] A. Ono, M. Amano, and M. Hakaridani. *A Flexible Content-Based Image Retrieval System with Combined Scene Description Keyword.* In Proceedings of IEEE Conference on

Multimedia Computing and Systems, pp. 201-208, 1996.

[40] D.S. Park, J.S. Park, J.H. Han, and T.Y. Kim. *Image Indexing using Weighted Color Histogram.* In Proceeding of the 10th International Conference. On Image Analysis and Processing, 1999.

[41] Y. Park. *Polymorphic Function Retrieval via Formal Concept Analysis.* In Proceedings of The AoM/IaoM International Conference on Computer Science (ICCS), pp. 165-170, 1999.

[42] Y. Park. *Software retrieval by samples using concept analysis.* Journal of Systems and Software, Vol. 3, No. 54, pp. 179-183, 2000.

[43] G. Pass, and R. Zabih. *Histogram Refinement for Content-Based Image Retrieval.* IEEE Workshop on Applications of Computer Vision, pp. 96-102, 1996.

[44] G. Pass, R. Zabih, and J. Miller. *Comparing Images Using Color Coherence Vectors.* In Proceedings of ACM Conference on Multimedia 96, Boston MA USA, pp. 65-73, 1996.

[45] A. Pentland, R. Picard, and S. Sclaroff. *Photobook: Content-based manipulation of image databases.* In Proceedings of SPIE, Vol.2185, pp. 34-47, 1996.

[46] Y. Rui, T.S. Huang, and S.F. Chang. *Image retrieval : Current techniques, promising directions and open issues.* Journal of Visual Communication and Image Representation, Vol. 10, No. 4, pp. 39-62, April 1999.

[47] Y. Rui, T.S. Huang, S. Mehrotra, and M. Ortega. *Automatic matching tool selection using relevance feedback in MARS.* Proceedings of 2nd International Conference on Visual Information Systems, 1997.

[48] B. Scassellati, S. Alexopoulos, and M. Flickner. *Retrieving images by 2D shape : A comparison of computation methods with human perceptual judgments.* SPIE storage and Retrieval for Image and Video Databases, San Jose, Vol. 2185, pp. 2-14, 1994.

[49] E. Sciore, M. Siegel, and A. Rosenthal. *Using semantic values to facilitate interoperability among heterogeneous information systems.* ACM Transactions on Database Systems, Vol. 2, No. 19, pp. 254-290, June 1994.

[50] M. Siff and T. Reps, *Identifying modules via concept analysis.* The International Conference on Software Maintenance, Bari, Italy, pp. 170-179, October 1997.

[51] J.R. Smith and S.F. Chang. *Intelligent multimedia information retrieval,* edited by M.T. Maybury. In Querying by Color Regions Using the VisualSEEk Content-Based Visual Query System, 1996.

[52] J.R. Smith and S.F. Chang. *Visually searching the web for content.* IEEE Multimedia, Vol. 3, No. 4, pp. 12-20, July-September1997.

[53] J.R. Smith and S.F. Chang. *VisualSEEk : A fully automated content-based image query system.* ACM Multimedia 96, Boston MA USA, pp. 87-98, 1996.

[54] M. Stricker and M. Orengo. *Similarity of Color Images.* In Proceedings of SPIE : Storage Retrieval Image and Video Database III, Vol. 2420, pp. 381-392, 1995.

[55] M. Stricker and M. Swain. *The Capacity and the Sensitivity of Color Histogram Indexing.* Technical Report 94-05, Communications Technology Lab, 1994.

[56] H. Tamura and N. Yokoya. *Image Database Systems : A Survey.* Pattern Recognition, Vol. 1, No. 17, pp. 29-49, 1984.

[57] T. Tilley, Formal Concept analysis and Formal Methods. PhD Research Proposal, Faculty of Engineering and Information and Technology, Griffith University, December 2000.

[58] R. Wille, *Restructuring Lattice Theory : An Approach Based on Hierarchies of Concepts,* In Ivan Rival, editor, Ordered Sets, D. Reidtel, Dordrecht, Holland, 1982.

APPENDIX A : Correspondence with greatbuildings.com

Following is a print out of electronic communication by Mr. David Owen of 'greatbuildings.com'. This email explains the copy right issues for pictures by greatbuildings.com. It is important to note that these pictures are used to build a demonstration of this technique and for collection of results only and are not part of this manuscript at all.

Dear Stoney,

Thank you for writing to Artifice Images.

As part of the end user license for the CD-ROM, you are permitted to use
images from the GreatBuildings Collection for private, academic viewing with
no special permission required. Similarly, you may use images from
GreatBuildings.com for private, academic viewing.

If you'd like to include some images in your master thesis, then additional
permission may be required for non-private distribution of your thesis.

If there are specific images that will be included in your thesis, then we'll need
some additional information in order to determine if additional permission
is required. For example, how will the thesis be published and distributed (if at all)?

Please let us know whenever you may have further questions or other
suggestions for how we may be of service.

Best wishes,

David Owen
sales@artifice.com

On 2001.10.30 at 20:54, stoneyclub@home.com (stoney) wrote:

> Dear greatbuildings.com
>
> I have a question about the licensing.
> I'm a master student in Canada,
> and looking for some images(over 500 images)

```
> to be used in my thesis.
> My question is if I buy the CD-Rom, can I use the images
> in my thesis? I will use the images only in my thesis.....
> My thesis is about retrieving images like a search engine.
> I really need some images with well-formed information.
> I think your web-site is great and has everything I need....
>
> Please, help me and answer my question......
>
> Thank you for your help
>
> Sincerely
>
> Stoney

+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
  Artifice, Inc.                  ...the way of architecture
  http://www.artifice.com
  http://www.cadoutpost.com
  http://www.greatbuildings.com
  http://www.designcommunity.com
  800.203.8324 toll free . 541.345.7421 voice . 541.345.7438 fax
  creative tools and media for spatial design . Eugene, Oregon, USA
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
```

VITA AUTORIS

NAME	: JANG, TAEK-SUENG
PLACE OF BIRTH	: CHUNJU, SOUTH KOREA
YEAR OF BIRTH	: 1970
EDUCATION	: CHUNG-ANG UNIVERSITY, SEOUL, KOREA
	1990-1991, 1995-1997 B.Sc
	UNIVERSITY OF WINDSOR, ONTARIO, CANADA
	1999-2002 M.Sc