COLOR SEGMENTATION BASED ON ADJUSTABLE FUZZY NEURAL NETWORKS AND HIGH LEVEL UNDERSTANDING OF MAPS

NING ZHONG

Color Segmentation Based on Adjustable Fuzzy Neural Networks and High Level Understanding of Maps

by

©Ning Zhong

A thesis submitted to the School of Graduate Studies

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Memorial University of Newfoundland

August 2004

St. John's Newfoundland

Color Segmentation Based on Adjustable Fuzzy Neural Networks and High Level Understanding of Maps

Ning Zhong

Memorial University of Newfoundland, School of Graduate Studies, 2004

ABSTRACT

To digitize and record electronically the huge collection of topographical maps, development of a framework for computer-based color segmentation and high-level map understanding has shown an increasing importance.

Color map segmentation requires special treatment with respect to line features. However, few research efforts have been put into this area. A novel approach of color segmentation based on adjustable fuzzy neural networks is proposed. The approach is capable of capturing pixel variations along thin line features. Based on a physics model, two heuristics have been derived, which suggest certain pixel variation behaviors in overlapping and boundary areas respectively. Fuzzy neural networks combined with self-adjustment components have been developed. The self-adjustment components dynamically adjust sample pixels among different sample clusters in neural network training. A feature that distinguishes this method from previous supervised neural computing methods is that, by adoption of self-adjustment architectures, it does not strictly require that all samples should have their desired outputs given in advance. Experiments show that the developed method can produce satisfactory segmentation results.

This study also proposes a novel method for high-level map understanding. A Description Logics (DL) language, $\mathcal{GALC}(\ell, \mathcal{D})$, is introduced to represent spatial objects and their relationships. A set of derivation rules is then proposed to derive a special grammar, called map grammar, from the DL representation. Thus, the map understanding process can be treated as a grammar parsing process. The map grammar is different from traditional grammars in that: (1) it allows for ambiguity; and (2) the input is a collection of primitive map objects instead of an ordered sequence of tokens. A map grammar parsing algorithm based on the Multiple Path Stack (MPS) is proposed. One advantage of this approach is that the knowledge representation is verifiable at design time. In addition, the implementation based on this approach is more robust and highly reusable, since the knowledge representation is separated from the inference mechanism. This research is a first step towards applying DL theory to map understanding. A prototype map understanding system was developed and applied on several test maps. The results show that this method can obtain a satisfactory interpretation of map phenomena.

Keywords: Conceptual Modeling, Knowledge Based System, Map Understanding, Description Logics, Syntactic Analysis, Color Segmentation, Fuzzy Logic, Neural Networks, Image Analysis, Pattern Recognition, Feature Extraction, Geographic Information Systems, Meta-Data Modeling.

Acknowledgments

Writing a dissertation is very challenging work. Without the generous help of others, it would be virtually impossible to finish this work. A lot of people have offered invaluable support, advice, guidance, assistance, and inspiration either directly or indirectly. I would like to thank all those people who made this thesis possible and gave me continuous support during the program.

First of all, I wish to express my highest gratitude to Dr. John Shieh, who guided this work and helped whenever I was in need. Without his constant support, guidance, challenges and encouragement, this thesis would not have taken this final form. I would also like to express my deep appreciation to the members of my supervisory committee, Dr. Siwei Lu and Dr. Krishnamurthy Vidyasankar, who took great efforts in reviewing and providing valuable advice and comments regarding various aspects of the thesis.

Thanks are due to the graduate students and postdoctoral fellows in the department for vigorous and helpful discussions. Those long nights at the computer in the department lab will not be forgotten. Special thanks to Dr. Boting Yang, Dr. Guangda Hu, Yaguang Chen, Vasantha Adluri, Kaleem Momin, Xueming Li, Jianmin Su and many more.

I am greatly indebted to the faculty and staff of the Department of Computer Science, Dr. Paul Gillard, Mr. Nolan White, Ms. Elaine Boone, Mrs. Jane Foltz, Ms. Sharon Deir and many more, for their assistance and support throughout my

program.

I would also like to thank Dr. Robert Gravina for his advice and the time he spent on proofreading the thesis.

Last but not the least, I would like to thank my family: my father, Chunxiang Zhong; my stepmother, Dan Zuo; my wife, Yi Miao; my brother, William Zhong; and my adorable daughter, Denise. Without their endless encouragement, support and love for me, I would never have been able to finish this thesis.

This thesis is dedicated to the memory of my mother, Huiqian Wang.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Description and Motivation

With the rapid progress in Geographic Information Systems (GIS), it is more and more desirable to develop computer based map image processing systems. To build such a system, various computer vision and image processing techniques and methodologies have to be analyzed, applied and integrated. This study is concerned with the problem of color map interpretation. Specifically, it focuses on two commonly encountered topics when building a map understanding system: (1) methods to achieve robust color map segmentation, and (2) methods to achieve a meaningful high-level description of a map.

The term "map" is a concept frequently used in all aspects of our daily life. A map is a graphic depiction which is drawn usually on a plane surface. It represents all or a part of a geographic realm in which the territorial features have been emulated

using conventional signs in their correct spatial locations at a reduced scale, serving as a method of visually interpreting the data representing real-world features. The conventional signs (legends) used on a map are normally printed in the margins of the map.

In a general sense, understanding means that we acquire and conceptualize information in the way we are familiar with, so that the information is ready to be applied to solve problems. To understand a map means to grasp its meaning expressed in the drawing. Scanned color map images contain raw geographic data which are 2-dimensional arrays of pixels. The computer based map understanding system usually starts from raw data, and then goes through a complex conversion process to obtain a meaningful high-level description in terms much closer to the ones human beings use in their thinking. Such terms can be "river", "road", "bridge", "near", "on", "over", "cross", and so on, which human beings feel comfortable to use in their daily lives. As a specialized form of image understanding, map understanding is usually carried out at two different levels: low-level and high-level. Unfortunately, most of the research to date only addresses the issues of low-level processing, such as automatic segmentation and recognition of isolated and primitive map features (line segments and point symbols). Although the word "understanding" has been used a lot in the literature, it carries different meanings in most cases, since it is not clearly defined. By "high-level map understanding", we mean an understanding of rather complicated phenomena at a conceptual level on a scanned map. A complex phenomenon may involve many

basic map features. The entire map scene can be looked at as one single map phenomenon, which in turn can contain other map phenomena, such as river networks. A map phenomenon may also refer to a group of other phenomena that share common properties, such as "the cities that are flown through by the Mississippi river" and "all the road networks and river networks found on the map". To understand a map scene, that is, to acquire all kinds of geographical and spatial information in the scene, means to explicitly represent the *contents* of the scene in a comprehensive and concise way. For a human being, understanding a map is a process that converts the geographical and spatial information encoded in a physical map into an explicit and structural representation, conforming to his natural way of thinking. An important property of such a representation is that it can be directly and intuitively applied to a problem. Similarly, automatic map understanding is a process used by a computer system to convert the same map information into a description that can be directly used to carry out computer vision tasks involving map information. Consider, for example, the following query: Locate a lake that is within 3 miles of the Charles river, that is north of highways 95 and 14, and that is surrounded by a number of buildings. This type of query would be answered in a simple and straightforward manner if the relevant spatial objects and relationships are stored in a taxonomic format. By understanding, the intended meanings (usually implicitly) encoded in the physical map are described in an organized and well classified fashion, so that manipulation of such understanding results can be carried out naturally by computers. The understanding

results can further be applied to various applications. There has been no system that can perform such kind of high-level understanding tasks on maps.

In this dissertation, the issue of how to separate map images into different color layers is discussed first, then an approach to high-level map understanding based on the results of low-level image processing is presented. Since colors are widely used to distinguish map features, robust methods are needed to take advantage of the color information. The scribed sheets used by the modern printing industry to mass produce maps can provide a desirable means to distinguish features of different colors. However, such scribed sheets are not always available or easily accessible. For example, the large collection of historical color maps does not have available printing scribes. In order to store them in digital format, color segmentation method based on a printed map is indispensable to ensure the quality of the digitization. A physics model is used to describe light reflection on maps, so that factors that cause pixel variations are studied. A novel method, which is able to capture pixel variations in overlapping and boundary areas, is proposed.

Consider, for example, a sample map image in Figure 1.1. Shown on this map are river networks, road networks and some point symbols, such as city and bridge symbols. The following basic building components can be found on a typical map: circles, polygons, areas, and line segments. Through existing low-level image interpretation techniques, we can extract these features. However, we cannot understand this map only by putting together these basic features, even for such a simple map.

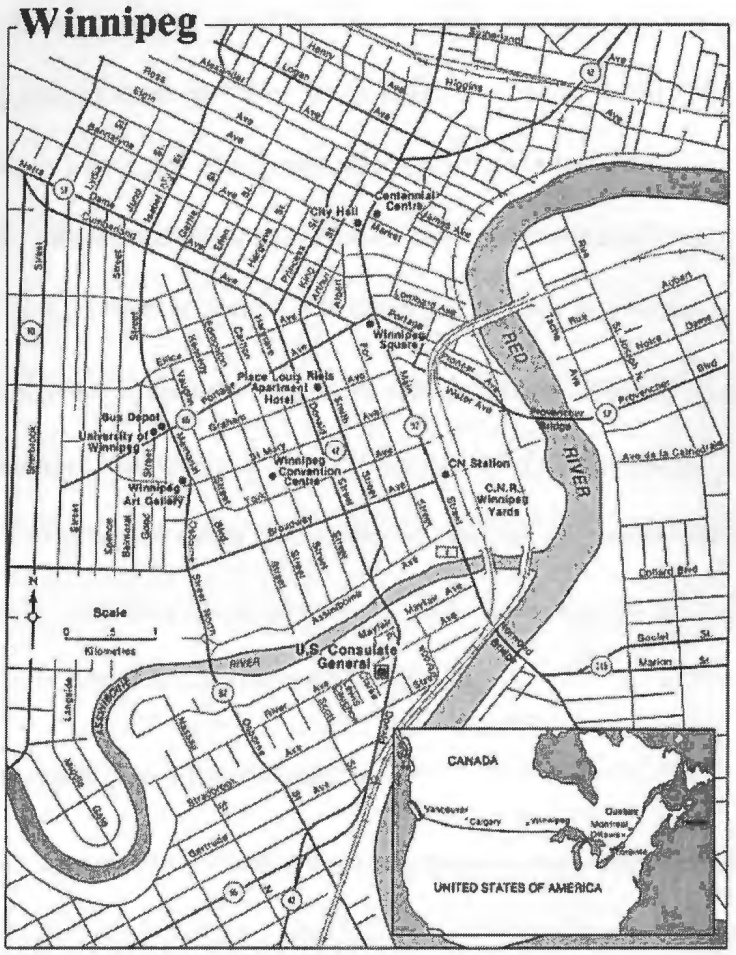


Figure 1.1: A city map of Winnipeg, Manitoba. Map image courtesy of UT Library Online.

Instead, by saying that we understand a map, we have to be able to answer questions such as:

- How are the basic map features grouped together to represent certain geographic phenomena, especially those that occupy a large area?

  This question requires us to specify how a map feature can be defined with other map features. Usually we look at a map at different abstract levels.

Sometimes we may just want to have a rough idea of what are in the map by asking questions such as "How many roads are there?" and "How many rivers are there?". There are also times when much detailed information needs to be obtained. "How many branches does a road system have?" and "What is the location of a particular road?" are examples of such questions.

Occasionally the map information is revealed from different viewpoints. The same map may be explained very differently by different users. A driver sees a map as a drawing of roads and cities, so that he can figure out how to travel from city to city using the road system, while a traveler by train may only want to know how areas are connected by railroads.

- How are map features interrelated?

  Answering this question means revealing various relationships among a number of map objects, especially the spatial relationships. Suppose, for example, that we want to visit those cities and attraction sites along highway 93 when we drive from New York to Boston. The map understanding system has to go beyond the extraction of isolated map objects and discover relationships among highway symbols and other point symbols.

- How does a map feature represent its counterpart in real world?

  This is a question regarding the semantics of a map feature. It is known that a map is an analogical representation of a real world scene. It cannot be under-

stood merely by what are explicitly drawn on a map. The intended meaning of a map object may not be told by its metrical and topological properties. Its context (relationships with other spatial objects) also plays an important role in determining its meaning. For example, a segment of a line drawn with blue color may be recognized as a river. But it can also be seen as part of a river bank, or even part of a lake boundary. A cartographer's intentions in drawing these blue lines are not necessarily reflected in the map itself. Although map features are symbolized according to government standards and rules, common-sense knowledge is still needed to accurately capture the semantics of a map feature.

The language we use to describe the map understanding results should be able to describe not only pieces of geometry objects (polylines, circles, and line segments), but also those meaningful entities in the real world (bridges, cities, and highways). Whether an understanding is sufficient enough depends upon the purpose. For example, if we have to carry out the task of storing the map information contained in Figure 1.1 in a GIS database, the facts about the map that are needed in further analyses and queries have to be extracted. We have to understand the fact that the map includes one river system and one road system. Further information about how the road network and the river network are made of road sections and river sections respectively has to be obtained. The understanding process will keep exploring the map until we have made explicit all the details that must be stored in GIS.

Generally, high-level map analysis and understanding cannot be directly performed on original map images. A series of preprocessing or low-level feature extraction steps is needed to convert maps into formats appropriate for high-level understanding. A source of inaccuracy results from the data acquisition stage. Paper maps are usually more or less distorted by tears and wrinkles. One important task of map understanding, therefore, is to correct some of the errors generated from previous stages.

Hence, automatic map understanding is an extremely challenging task. In an attempt to obtain a systematic and effective approach for automatic high-level map understanding, a novel approach is proposed in this dissertation to overcome the following limitations of current methods:

1. Current research in this field only advances on the experimental frontier. The problem is not stated in a formal fashion. This dissertation attempts to formalize the concepts and the reasoning mechanism involved in solving the map understanding problem. The formal approach enables us to combine and extend existing theories such as Descriptions Logics and formal language parsing theory into a more comprehensive, coherent and general theory for map understanding. One of the outstanding advantages of formal descriptions, which cannot be achieved by current methods, is that we can verify the consistency of our knowledge representation. Any conflicts in the knowledge base can be discovered at design time. The precise nature of formal representation makes it

easier and more flexible to be extended to other domains.

2. Almost all the current methods address the issue of knowledge representation and the underlying inference mechanism. However, the formation of those knowledge representations (rules, semantic networks) or inference algorithms is an ad-hoc process. From an overview of the current methods, we can see that only general strategies, such as top-down or bottom-up, are given, which are not enough to solve a complicated problem. A formal approach enables us to *explicitly* manipulate any piece of knowledge involved and any inference step taken. Thus, the analysis of the reasoning process becomes much easier.

3. Understanding has to reveal information about a phenomenon at various abstract levels and from different viewpoints. It is still a challenge for current methods to handle hierarchies of various objects, especially in the context of maps. Features on such maps have more hierarchies than certain drawings (cadastral and utility maps) have. Cadastral maps and utility maps are usually drawn with objects of regular geometric shapes, whereas maps are full of irregular shapes, since they are abstractions of the real world surface features. Based on Description Logics, this dissertation provides an effective and well-formalized way to capture the semantics of objects and relationships.

4. Most existing solutions are difficult to adapt to other application domains, because the knowledge representation and the reasoning mechanism are not clearly

separated. Through the formal representation of map objects and their relationships, the reasoning algorithms can be specified without reference to a specific domain.

Figure 1.2 shows an overview of activities involved in the development of a map understanding system using the proposed methodology. The ovals represent a series of development steps. The input and output specifications for the steps are represented by underlined phrases.

The tasks of each step are elaborated in more detail in the rest of this work. A brief description of the several stages undergone to obtain a solution is given below. An initial step of building the system is to identify objects and relationships among objects. The result of this step is an object-relationship graph. The identified objects and relationships are further formalized with a DL language, $\mathcal{GALC}(\ell, \mathcal{D})$. $\mathcal{GALC}(\ell, \mathcal{D})$ is an extension of the conceptual language $\mathcal{ALC}(\mathcal{D})$ [38]. With the help of a set of derivation rules, a map grammar different from traditional grammars is obtained. Then a parsing algorithm designed specially for the map grammar is proposed and implemented. The output of a map understanding system should give an accurate and meaningful representation of map objects and their relationships suitable to be stored in computers [75]. Figure 1.3 gives a graphical sketch of objects and relationships involved in the construction of a road network.

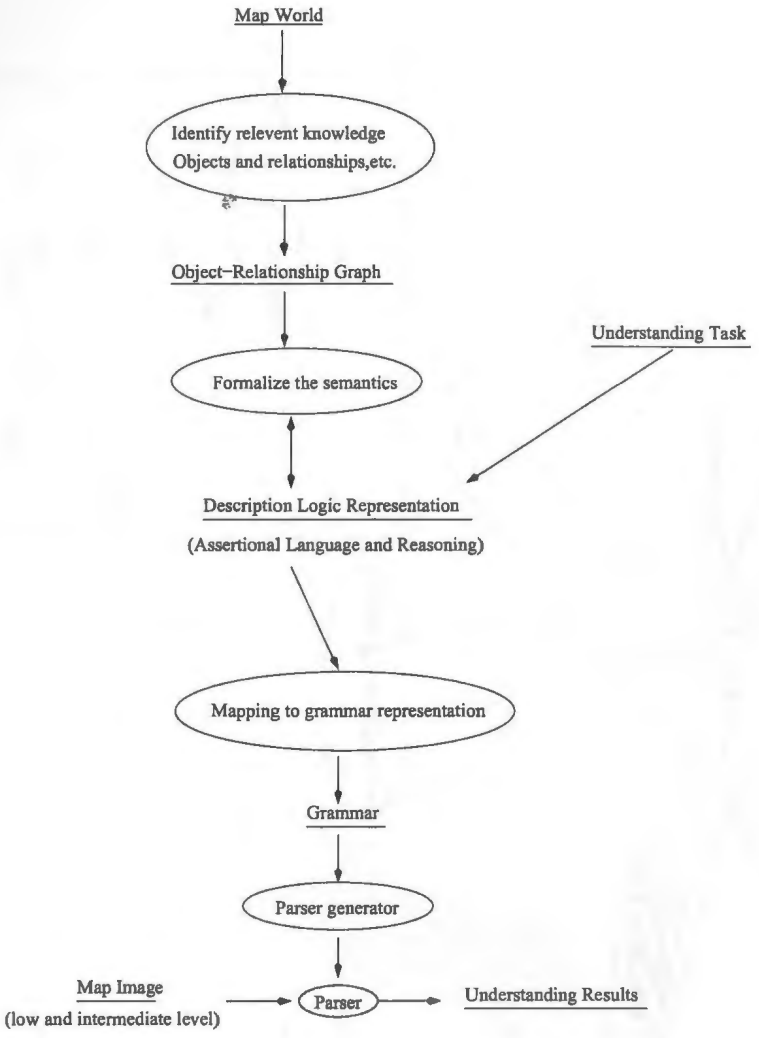


Figure 1.2: An overview of system development activities.

1.2 Summary of Contributions

The main contributions of this work are summarized as follows.

Firstly, a novel approach is proposed to attack the problem of color map segmentation. A physics model that describes light reflection from paper maps is presented. Two application specific heuristics that describe color pixel variations in overlapping

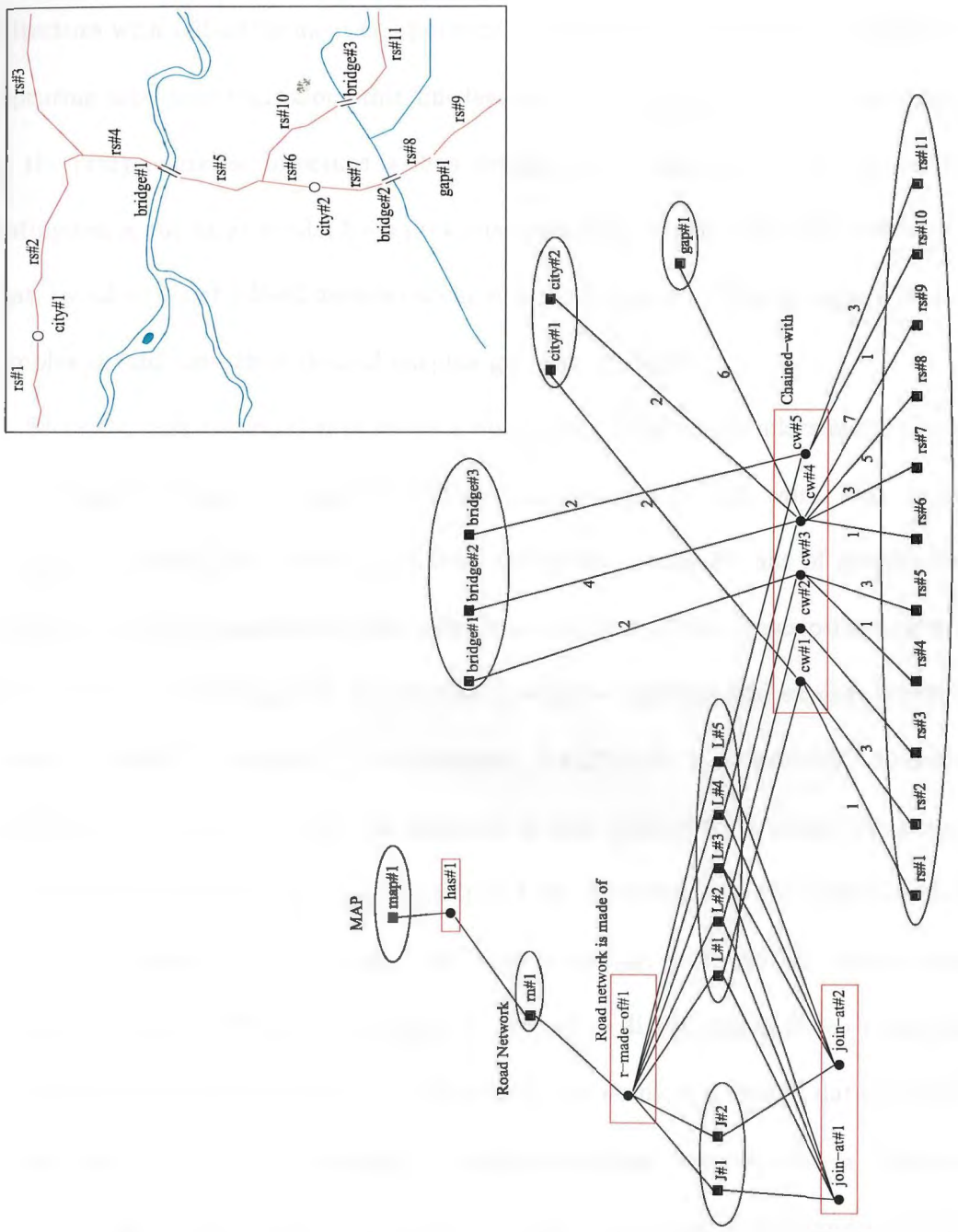


Figure 1.3: Graphical depiction of map understanding output.

and boundary areas are derived. Based on the heuristics, a unique fuzzy neural architecture with self-adjustment components is proposed. This approach is capable of capturing pixel variations along thin line features. A color segmentation system based on the fuzzy neural architecture is then designed and implemented. A feature that distinguishes our fuzzy model from previous supervised neural computing methods is that, by adoption of self-adjustment architectures, it does not strictly require that all samples should have their desired outputs given in advance.

Secondly, this dissertation proposes a novel method for map understanding based on Description Logics, a formal representation system. In this work, map understanding is treated as a process of transforming the analog format of geographical information (color maps) into a high-level interpretation. With Description Logics as the conceptual modeling tool, we can clearly separate implicit and explicit representations of spatial knowledge. A DL language, $\mathcal{GALC}(\ell, \mathcal{D})$, is proposed by extending $\mathcal{ALC}(\mathcal{D})$ with n-ary roles. The DL language is then applied to formalize the semantics of map objects and their relationships. A set of derivation rules is presented to formulate a special grammar, called the map grammar, from the DL based knowledge representation. The map grammar is defined to distinguish between existential symbols and constraint symbols. An algorithm that employs a special data structure called multiple path stack is designed to carry out the map parsing process. To determine the feasibility of the proposed method, a prototype map understanding system has been developed and applied on several test maps.

1.3 Dissertation Organization

The organization of the rest of this dissertation is given below. Chapter 2 gives a review of color image segmentation and automatic map interpretation methods. The main techniques are presented and summarized. Chapter 3 presents the color segmentation method based on adjustable fuzzy neural networks. The discussion is centered around how to capture pixel variations along line features. Chapter 4 briefly reviews the motivations of conceptual modeling in map understanding. Then the general issues concerning using DL as the modeling tool are discussed. In the same chapter, a conceptual language $\mathcal{GALC}(\ell, \mathcal{D})$ is proposed by extending $\mathcal{ALC}(\mathcal{D})$. Chapter 5 is devoted to the formalization of map knowledge based on the Description Logics theory. Typical concepts and roles in the domain of map understanding are presented. Chapter 6 covers the map understanding process, which is viewed as a process to build a data structure that holds object instances and relation instances. Chapter 7 describes the method to establish a reasoning mechanism for the map understanding system. A method to form a set of grammar rules for map understanding tasks is presented. It is demonstrated that a map understanding process can be treated as a grammar parsing process. The experiments conducted and the results obtained are presented in Chapter 8. Finally, this dissertation is concluded in Chapter 9.

Chapter 2

Related Work

There are different types of images, such as medical images, remote sensing images, X-ray images, and map images. Among the topics in image processing and interpretation, map image processing is a relatively under-researched area. Different types of images are obtained from different sources, or produced with different devices, using different methods, and with different focuses of interest. Although they share some general methodologies in terms of processing techniques, each of them has exclusive characteristics that have to be treated individually.

Maps are an analog or an abstraction of real world phenomena, which do not directly depict naturally occurring objects. There exist a limited number of colors and symbols on maps. The number of constrained interactions among the map symbols is also limited. However, it is very difficult to capture the semantics of map features and the constraints among them because a lot of domain specific knowledges are implied in those analog formats and abstractions.

There is a large amount of literature aimed at drawing analysis and interpretation at the lexical and the syntactic level, such as character recognition, symbol recognition, and line feature extraction. The amount of publications dealing with the high-level image interpretation is very small, and most of them are confined within some particular application domains. In this chapter, we will review various research efforts related to the whole or part of a map understanding process, as well as low-level image processing methodologies, which are the basis for map understanding.

2.1 Color Image Segmentation Techniques

Automatic map feature extraction and recognition is an active research area and is especially demanded in GIS applications. An effective color map analysis system should be able to separate colors on maps where each type of feature is designed to be painted with a unique color. Color image segmentation is generally viewed as a process of dividing an image into a number of disjoint areas, each of which contains pixels representing one color. This kind of research can be roughly classified into two groups: (1) feature space analysis techniques, and (2) physics-based segmentation techniques.

The feature space analysis techniques [25, 84, 20] are widely used for solving low level image understanding tasks. In general, pixels belonging to the same significant feature appear to be similar. By mapping the feature vectors into the space spanned by their components, significant features can usually be classified because

they correspond to high density regions.

The clustering methods are further divided into supervised and unsupervised methods. When prior knowledge about objects is available, supervised color classification methods can be used. Some researchers [65, 85] have used an unsupervised method, k-nearest neighbors, in color image classification. Pixels are assigned to classes based on the distribution of their "nearest neighbors". Usually the classification algorithm selects the K pixels with known classes which are closest to the pixel being investigated and selects the class which has more pixels in the neighborhood. One of the drawbacks of this method is that the process of searching the k-nearest neighbors for each point is time consuming. Another Clustering method, the k-means algorithm, also has been used in color image segmentation tasks [1]. Traditional k-means methods aim at minimizing the sum of squared distances of all pixels to their preassigned cluster centers. It involves an iterative scheme that operates over cluster centers. The disadvantages of this technique are that it is time-consuming and that the number of cluster centers must be known in advance. Some variants of the k-means algorithm use the unsupervised method. These variants are more adaptive since estimation methods are used to compute the initial clusters. However, some of the resulting clusters may be high density spots within a limited local region. Such clusters cannot constitute significant features themselves. They, in turn, should belong to some other clusters. Moreover, the effectiveness of traditional clustering methods lies in the assumption that individual clusters obey multivariate normal

distributions.

To avoid the drawbacks mentioned above, Comaniciui and Meer [25] have used a non-parametric density estimation procedure based on the mean shift algorithm to estimate density gradients. First, an adequate number of search windows are defined at random locations in the space. The mean shift vector is then computed and the search window is translated by that amount. This is repeated until convergence. The computation model of this method is simple, but this method only focuses on extracting high density centers. For some images, finding the cluster centers means a key step for segmentation, while for some others, the issue is how to discern those pixels with close values but belong to different clusters.

When the standard pixel values corresponding to each specific color are known in advance, the categories of pixel values can be told through their distances from the standard ones. Valavanis [90] takes advantage of this observation in his method. A single-value total color difference (TCD) measurement for scene segmentation is proposed and evaluated experimentally. Both chrominance and luminance difference criteria are considered. The luminance component is defined by a unit in luminance change expressed in terms of MacAdam's Just Noticeable Difference (JND). The chromaticity component is derived directly from JND. Experiments using both pixel and region analysis show that the proposed TCD can effectively indicate object boundaries over a wide range of luminance changes. This method is based on the assumption that a color clusters around a standard value. It cannot take into account other fac-

tors, such as overlapping colors, poor scanner quality, or boundary colors. It is very easy for human vision to distinguish different colors, but it is difficult to assign the standard values.

The methods mentioned above assume that the same color in a map image has approximately the same pixel value. The pixel variance is only caused by noises and random errors. Therefore, the pixel values will be centered around cluster centers. The closer any two pixel values are, the more possibly they belong to the same cluster. The methods are suitable when prior knowledge about pixel behaviors is known and the pixel changes are predictable. The difficulty is that without enough knowledge on the color pixel distribution, the difference criteria are difficult to obtain. Therefore, more accurate and flexible methods are needed to avoid the above problems.

Neural networks are another class of color clustering techniques. The strength of neural networks is their learning and adaptation abilities via parameter training. It has been argued that a three layer neural network can form arbitrary complex decision regions and can therefore separate populations of patterns [27, 95]. However, the quality of training depends on the samples given. Some pixel classification methods have used neural networks to perform supervised classification tasks. There are different kinds of neural networks, each of which, in its own way, has advantages and disadvantages. Yan and Wu [91] proposed a multi-layer neural network based technique for extracting characters and lines from geographic color map images. This neural network is first trained with feature values at known characters, lines, and

background pixels, and then used for image classification. The image segmentation problem is treated as a pattern classification process and a neural network classifier is used to generate non-linear decision regions to separate the foreground and background of an image that may contain a number of non-uniform regions with different colors. This method produces better results than the adaptive thresholding method does. However, it aims to separate lines and characters from the background. The characters, river lines and road lines are treated as one feature, and the issue of further separating them is not addressed.

Unsupervised learning methods, such as Hopfield networks, competitive networks and self-organizing maps (SOM), are also used in color clustering [62, 63, 5, 6]. Most of them combine clustering and histogram threshold techniques. Color image segmentation is frequently based on pixel classification, either supervised or unsupervised, without considering spatial information. By applying histogram analysis, the prominent color clusters are obtained. A segmented image can be obtained by assigning the mean color of each cluster to the corresponding pixels of the image. The methods assume that homogeneous regions in an image correspond to color clusters which generate at least one visible histogram peak. Tseng *et al.* [64] proposed a circular histogram thresholding which takes the periodic property into account. The hue component plays a very important role in recognition, because the human eye can only detect a few dozen intensity levels in a complex image, but can perceive thousands of chromatic variations. This method uses a 3-D cylindrical colorspace (IHS),

where hue is a periodic function of angles with period 360^o. Unlike the traditional histogram method, this method does not treat pixels at hues H(0^o) and H(359^o) as two far-apart clusters, so that it yields better segmentation results than a traditional histogram method does. The drawback of histogram-based methods is that their effectiveness relies on the proper threshold selections. The segmentation results are less satisfactory when the histogram does not provide distinguishable cluster information.

There have been growing interests in using fuzzy logic theory in color image segmentation [11, 12]. Unlike traditional color clustering methods, which make crisp decisions about whether a pixel belongs to a cluster, fuzzy logic based methods allow ambiguous boundaries between clusters. Huntsberger *et al.* proposed an iterative optimization method, which decides the memberships of a pixel in each cluster according to the distances between the pixel and the cluster centers. A criterion function is used in each iteration in order to minimize the distance between a pixel and its cluster center, and to maximize the distances among cluster centers.

Fuzzy c-means is a clustering method that has been applied to various color image segmentation tasks [13, 14]. Lim and Lee [2] presented a two-stage (coarse and fine) segmentation algorithm. Histogram thresholding is used for coarse segmentation with the purpose to automatically find the number of clusters and the center of each cluster. At the fine stage, fuzzy c-means is used to assign the unclassified pixels to the closest cluster using the detected cluster centers. Like most thresholding techniques, this method is based on the assumption that the histograms show valleys that suggest

coarse cluster information.

Physics based techniques [3, 4] are based on physical models which describe the effects of light reflecting, shading, and highlights in a scene, especially the reflection and illumination interaction that is responsible for the color differences in an image. Through the modeling of the luminance and reflection behaviors of objects in the scene, these approaches analyze the influences of models on pixel values and establish criteria to classify the pixels. The traditional methods of color image segmentation suffer from too many erroneous regions because they have not accounted for the influence of optical effects on object colors [8, 9]. Physics based approaches are used to deal with various color images. Unfortunately they are seldom used on map images. Drew *et al.* [26] have proposed a linear-transform approach to describe illumination change among RGB channels within a certain illumination environment. An underlying principle different from that usually suggested was proposed. The resulting new method, the Linear Color algorithm, is more accurately illuminant-invariant than the previous methods. An implementation of the method uses a combination of wavelet compression and DCT transform to fully exploit the technique of low-pass filtering for efficiency. The method can produce very encouraging results for extracting objects that cover continuous regions.

Klinker [89] proposed an approach based on an intrinsic model to describe the effects of shading and highlighting in a scene. This method shows that the description of physically feasible color changes on one object can guide an algorithm in interpret-

ing color images. The algorithm finds characteristic color clusters in an image and relates their shape to hypotheses about the object color and degree of shading, as well as the strength, position, and color of the highlights on the object. Although the photos of the images used in this study were taken under a controlled condition in order to eliminate certain noise effects, this method demonstrates that physical models can lead to reliable and effective color image segmentation methods. One limitation of the physics based methods is that they are only used to model light interaction phenomena in a scene. Light interacts in many complex ways with the environment, and there is no single model that can take all the factors into consideration and give an exact description of pixel variations in the scene. Therefore, most methods based on physics models have to concentrate on relatively few significant factors that affect pixel values.

In the methods discussed above, color information is primarily used for segmentation; spatial information is not taken into consideration. In many cases, spatial information also plays an important part in the determination of color clusters. Hopfield based methods are such an example of considering relations of neighboring pixels. The Hopfield network provides a general algorithm for finding approximate solutions for "hard" combinatorial optimization problems [7, 62]. The segmentation problem is treated as one of minimizing a suitable energy function for Hopfield networks. In the design of the energy function, the assignment of adjacent pixels to the same class is favored. Huang [7] presented a method to locate the significant peaks by applying

histogram analysis and designed three different networks (one for each color feature). The segmentation results of the three color components are combined to get the final image. RGB color features, the I1, I2 and I3 color features, and the Karhunen-Loeve transformation of the RGB color features (KL-RGB) have been used in the experiments. Campadelli [62] proposed an algorithm to build a single Hopfield network with $M \times N \times S$ neurons to segment color images. M, N are the image size, and S is the number of selected clusters obtained by histogram analysis. For both algorithms, histogram analysis is very important since it produces the coarse segmentations and determines both network structures and their initializations. In the algorithms, the spatial information is also considered in order to produce consistent color pixel labeling.

Region growing and splitting algorithms are usually used in combination with color, contour, and shape information for color image segmentation [44, 61, 74, 79]. Okamoto [92] used a method for color segmentation by integrating color and range data. The segmentation consists of two steps: initial segmentation and region splitting. The initial segmentation is based on the histogram of the entire image. The brightness, hue and saturation information are used. In low saturation regions, hue information is not important, so it is not used. The image is first divided into two categories according to the saturation. Low saturation regions are segmented by brightness, and high saturation regions by brightness and hue. Since similar color regions may still be grouped in one region, they should further be split into the corre-

sponding regions using edge information obtained from both the brightness and hue. Initial regions are then merged by using color and range information. However, this method is only effective in segmenting 3-D scenes with some 3-D objects of regular shapes and planes, such as indoor scenes with a few boxes, tables and walls.

So far there is no universal methodology for color image segmentation. In other words, all the currently existing methods are application dependent, and no colorspace representation is applicable to all color images. Color map images are one class of color images that have attracted much less attention than others. In addition, while many of the assumptions and pixel level treatments can be applied, with or without adaption, to a vast range of color images, such assumptions and treatments are not suitable for color map images, not even to a small degree. Most of the current work on color image segmentation is performed with the goal to segment objects or regions from one another. Most approaches define objects (regions) as continuous areas on images that have similar color, or show certain patterns of behaviors. Color map images, on the other hand, display a largely different nature from those color images that have been heavily researched. The most significant features on maps are line features. They extend across a large portion of a map and usually a whole map, and they do not bear any resemblance to usual regions. They express fairly complex spatial phenomena, yet the pixels they take only constitute a small percentage of the total pixel number of an image. Moreover, overlapping colors make the segmentation tasks much more challenging. Some segmentation techniques, such as histogram

thresholding, clustering, and region growing, cannot be applied to color map images in the usual way, since the thin line features can easily become ignored. It is essential to supplement any solutions with a priori knowledge of the specific color images. As a result, it is necessary to develop a color segmentation method specifically designed for color map images.

2.2 Automatic Map Interpretation

A number of works in automatic map interpretation have been reported. They all work towards revealing spatial properties of geographical entities and their relationships with minimal human intervention. For a human the process of understanding a map is associated with forming a mental image through his vision system [43]. Automating such a process using computers is a very difficult task. Human beings treat the map pictorially as a whole, but computers have to start the process from primitives such as lines and points and go through a lot of computations before reaching a conclusion. To verify a spatial fact (such as parallelism of two roads) using human eyes is much easier than using computing algorithms. Most of the high-level image understanding methods use a knowledge based approach [46, 48, 66, 71, 72, 69], which tries to incorporate human intuition and experience.

2.2.1 Low Level Image Processing Techniques

After the color segmentation stage, a number of map layers containing various features are produced. Feature extraction is always the key step in the whole recognition process. The purpose of feature extraction is to discern interesting features from a group of features. It can be found in the literature that most of the research concentrates on the design of algorithms for features with topologic attributes, such as curve lines, characters, and symbols. To extract features from map images, different situations have to be dealt with [42, 45, 53, 60]. Most of the time we need to apply different techniques to solve different problems. Different techniques in combination may undertake the overall task of feature extraction. Therefore, the techniques in feature extraction can be divided according to the sub-problems to solve. They are usually divided into the following categories: (1) techniques to separate different objects [46, 55, 56, 58, 73, 81], (2) techniques to detect line objects [41, 47, 50, 70], (3) techniques to track and reconstruct defective line objects [54, 56, 67, 76, 80, 86, 87, 88], and (4) techniques to vectorize line objects with accuracy [78, 93, 94]. Since the quality of low-level feature extraction methods has extensive impact on high-level map understanding, their capabilities and limitations can be viewed as part of the domain specific knowledge that high-level understanding should take into consideration.

2.2.2 Knowledge-based interpretation

Knowledge-based interpretation is a necessary tool toward application-independent systems. The knowledge can take many forms, depending on the type of application and the knowledge engineer's design.

Starting from the examination of various aspects of human visual information processing, Tjahjadi and Henson [68] proposed a knowledge-based model for image understanding. The model was proposed for depicting general principles in the knowledge acquisition and application. Only a basic structure has been given and a limited detailed knowledge has been provided. Mayer [29] presented an approach for knowledge based automatic extraction of map objects, such as roads, pavements, and buildings. This approach was based on a multi-level model. Each level describes objects and certain relations. However, only two relations (part/part-of and specialization/generalization) are considered. Only model-driven and data-driven search strategies are given as the knowledge utilization mechanism. Domain specific relation semantics and knowledge inference mechanism are not provided.

In order to make knowledge-based systems practical, many researchers confine their work within certain types of drawings instead of general images. Janssen [32] reported a model based method for cadastral map interpretation. This kind of interpretation is directed by the models, which specify map objects to be recognized, the properties these objects should possess, and how to detect these objects. The limitation of this method is that it lacks flexibility. This method works only with

"well-defined" objects, such as straight lines, curved lines, and characters. The model can clearly define the criteria and thresholds the objects should satisfy, so that the interpretation algorithms can check against these criteria and thresholds to determine which category an object belongs to. In topographical maps, objects intervene with one another. It is hard to specify the criteria and thresholds without considering the effects of neighboring objects, let alone handle the high-level understanding task involving hierarchies of various objects.

Abdelmoty [30] explored a rule-based method for ordnance survey map reading, especially rules for road network identification and road naming. This approach depends on identifying rules which exploit spatial properties of map objects. It assumes that a map is represented by a set of lines and points with associated feature codes. Each map feature (such as an island) corresponds to several rules that are formed to express how to search the map and find the feature by calculating geometric and topologic attributes (distance, connectivity, and angle between vectors). Except at the rule expression level, the map knowledge is still represented in a procedural way. Most of the spatial attributes and relations are not symbolized explicitly. Furthermore, despite the efficiency and simplicity of reasoning, vectorization introduces unwanted inaccuracies. Without morphology information, mere calculations of geometric and topologic attributes may not be sufficient for extracting map features.

Hartog [31] described a knowledge-based interpretation method for utility maps. Through primitive extraction, a representation of an image by its connected compo-

nents is obtained. The prior knowledge in the method is represented in a semantic network. The primitives and the semantic network are used as the input of the interpretation process. The search strategy of this method is based on the context reasoning mechanism. The interpretation process proceeds by matching the geometry description of the object type being searched with that of primitives, and by adding a series of new search actions based on the relationships specified by the semantic network. One issue with this method is that generating new search actions is not at all trivial; that is, it is not as simple as finding related object types and updating the search list with newly expected types. For example, relationship names "contains", "chained with", and "bridge over river" require well coordinated search actions and properly stored history information. The drawback of the semantic network based methods is their lack of precise semantic characterization [10]. The ultimate result of this was that every system behaved differently from the others, in many cases despite virtually identical-looking components and even identical relationship names. Different users can have very different interpretations of the relationships. Therefore, in many occasions, the success depends on whether a semantic network suitable for the understanding task is identified, and proper reasoning strategies are employed.

The following are the main problems most researchers attempt to attack: (1) Identifying the knowledge representation for map objects and relationships to be recognized, and (2) Establishing an efficient search strategy for the application of the knowledge. However, relatively little research has been described regarding the is-

sue of capturing accurate semantics of map objects and relations. In addition, most existing map interpretation methods do not use declarative knowledge to represent domain knowledge other than geometric properties of map objects. In order to make improvements in drawing understanding, the knowledge representation should be better organized and structured [59, 77]. Description Logics [10, 38, 21, 22, 23], which evolved from semantic networks, formalize the semantics of objects and relationships, and in the mean time retain the taxonomic structure of semantic networks. Description Logics provide us with a tool to express and reason with complex definitions of objects, classes, and relations. The author proposes to use DL languages to formally describe geographical entities and their relationships in map understanding systems. Not only geometric and topologic attributes of map objects, but also the relations among map objects, task specific knowledge, states of interpretation process, and other control knowledge can be represented explicitly with accurate semantics. This lends us the opportunity to achieve maximum separation of declarative and procedural knowledge, and therefore to adopt more flexible search strategies. Especially for topographic maps, DL based approaches make it easier to incorporate human intuition and expertise into domain knowledge representation, such as the expertise to remedy distorted features resulting from the low-level primitive extraction methods. Furthermore, the declarative knowledge can be easily adapted and reused for new understanding tasks, because it provides improved readability and better understanding of map features.

2.2.3 Description Logics Based Methods

There are two categories of approaches for knowledge representation: formal and non-formal methods. Usually in the development of domain applications the non-formal methods are more popular, because they can evolve from intuition. Frequently we can start building applications without capturing all facts and relations about the world. A lot of problems can be left to be explored, refined, and solved during the later stage of the process. Some formalism may be used, but only to be expected as general purpose thinking or idea exchanging tools. The formal methods require unambiguous identification of properties and relations. All facts of interest that may participate in the future inference process have to be explicitly symbolized and their semantics to be clearly defined. No conflicts may occur in the formal specification, which is a requirement difficult to meet. Formal methods may seem unnatural for human beings because most of the time human begins make decisions based on belief, experience, and natural reasoning, instead of formalized logical reasoning. Nonetheless, through intensive analysis and well understanding, formal methods can be used in many domain applications, and the same solution can be applicable to different types of problems. Description Logics are such a modeling tool that provides the power and general machinery of formalism. It also allows the use of intuitive terms that are similar to the ones we currently use.

Current DL languages are derived from KL-ONE [39], which is a direct result of formal analysis of the shortcomings of semantic networks. It introduced most of the

key notions on DL languages. Most of the focus of the original work on KL-ONE was on how to present the concepts and how to reason with these concepts. The key characteristics of Description Logics reside in the formal definitions of the following: (1) Elements in the domain; (2) Concepts that are groups of elements sharing certain common properties; (3) Relationships of concepts; (4) Attributes of elements; and (5) Complex concept terms constructed by atomic concepts and roles. The basic relationships among concepts are value restrictions ($\forall$R.C) and exists-in restrictions ($\exists$R.C). A DL knowledge base includes two parts: T-Box and A-Box. Concepts are given set-theoretic interpretation: a concept is interpreted as a set of individual elements and a role is interpreted as a set of individual relations. The syntax and the semantics make it suitable for modeling all kinds of applications at an abstract level. Description Logics have been used in the following domains: natural language processing, software engineering, database management, configuration, medicine, digital libraries, web-based information system, and other domains [15, 16, 17, 18, 19].

Recently, there has been a growing interest towards content based image retrieval. It shares some common features with image understanding in terms of knowledge representation. Both try to recognize some objects or features of images by some kinds of thermal or structural descriptions of visual properties (color, shape and texture). The difference is that content based retrieval searches a collection of images and finds a matching image, while image understanding needs to retrieve those objects of interest and their relations on a single image. Meghini *et al.* [40] presented an image

retrieval model based on DL. A syntactic representation in DL to represent image form and image contents is given, as well as a formal query language. However, no example is given to demonstrate the application of such a model. The effectiveness of the model is yet to be verified.

Chapter 3

Color Segmentation of Map Images

3.1 Overview

Segmentation of color map images is very challenging because of the difficulty to capture pixel variations in overlapping and boundary areas along thin line features. This dissertation provides an effective method to solve this problem. Based on a physics model that describes the reflection property on maps, two heuristics have been derived which suggest certain pixel variation behaviors in overlapping areas and along boundary areas respectively. Fuzzy neural networks combined with self adjustment components have been developed that dynamically adjust sample pixels among different sample clusters in fuzzy network training. A feature that distinguishes the proposed fuzzy model from previous supervised neural computing methods is that, by adoption of self adjustment architectures, it does not strictly require that all samples should have their desired outputs given in advance. For some sample data, the

desired outputs are decided automatically in learning. This can make preparation of sample data much easier when obtaining representative samples from overlapping and boundary areas. The self adjustment ability provides a flexible way to design a fuzzy model with a priori knowledge. Experiments show that the developed method can produce satisfactory segmentation results and is capable of capturing pixel variations which are crucial to guarantee high segmentation quality.

3.2 Introduction

Most existing color segmentation algorithms work with the assumption that each of the different colors on a color image has similar pixel values. This assumption is suitable and applicable for segmenting large color regions in images, combined with other supplementary image processing techniques, such as region growing and color region grouping. However, the features of interest on maps include not only regional features, but also line features which may extend throughout the whole map. Before subsequent recognitions, the information of line features must be extracted with certain accuracy, especially in complex regions, such as areas where different features intersect and overlap.

So far the color segmentation methods that aim at line feature recognition are not extensively researched. It is true that some image segmentation methods based on gray-scale value information can also be used to deal with color images if color images are treated as having only the intensity component. But they will suffer the

information loss when pixels of different colors happen to have the same gray-scale value.

Most color segmentation methods fall short to deal with the color segmentation of maps because of the following reasons: (1) Some line features are very thin; (2) Color variations along a line feature that spreads all over the map are difficult to grasp; and (3) Colors overlap with one another. This research is special in that color information of thin line features is captured in a much more accurate manner.

A novel method that is capable of capturing color pixel variations and adaptively learning from samples is presented in this chapter. In Section 3.3, a reflection model is given to describe the light interaction on a map surface. The model indicates that pixel variations normally happen along boundaries and in overlapping areas, and obey certain heuristics that have been exploited to develop the self adjustable structures described in Section 3.4.2. In Section 3.4, a fuzzy neural structure is presented, which is able to classify base colors through training. In Section 3.4.2, after the limitations of the fuzzy neural network are analyzed, it is shown how to incorporate a self adjustment structure into the neural network in order to capture pixel variations without supervision. Finally, experimental results are given in Section 3.5.

3.3 Reflection Model

To study the physics that cause pixel variations on topographic maps, the map producing process is first briefly reviewed in this section. The printing of a topographic

map involves several steps. Once the map manuscript is compiled, a map-size film negative is obtained from the compiled manuscript by photographic reduction. The image on this negative film is then photochemically reproduced on several thin plastic sheets to which a soft translucent coating (called scribecoat) has been applied. The next step is scribing. Engraving instruments are used to etch the map's lines and symbols. This is done by removing the soft coating from the hard plastic guide sheet. A map is edited several times before final scribed sheets are completed. A lithographic (photo-offset) plate is made for each color. A series of scribed sheets, each containing the map details of a single color, is prepared from the multiple plastic sheets. The actual printing is done by using a set of special inks. For each color, the map paper is put through the press once. Each time a new plate is put in the press, and the printing ink is replaced.

When digital files are available, computer-assisted technologies are increasingly used in the production of new maps and revision of existing ones. Some maps are printed in CMYK (four-color offset printing) method. At first, four color printing is not appropriate for map printing because of the difficulty to print very thin lines. On some occasions, publishers can print color maps on demand using CYMK method for customers with the help of special raster technologies. However, the majority of maps produced by USGS and NTS are still printed in conventional ways. In this dissertation, the color pixel variations on the traditionally printed maps are studied.

The light source and illumination conditions of a scanner are strictly controlled.

This lends us an opportunity to study the physics of light reflection from paper maps. When a ray of light hits a paper map, it is reflected on the paper or the color pigments painted. Depending on where a ray hits, three cases are considered: (1) hits the paper directly; (2) hits areas painted with only one color; and (3) hits areas painted with more than one color. It is assumed that a piece of paper is made of opaque material, whereas pigments used on maps are considered semi-transparent materials. The optical properties determine the power distribution spectrum of the reflected lights.

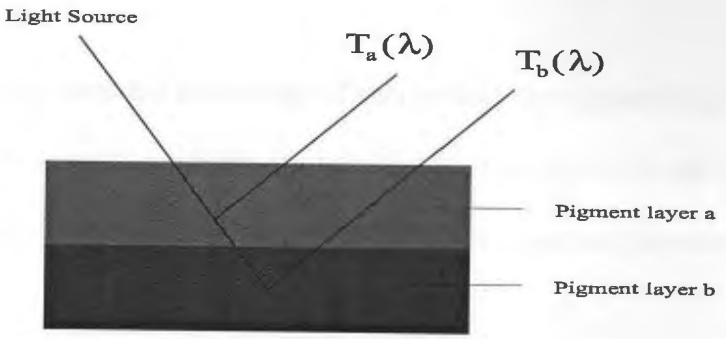


Figure 3.1: Light reflection on an area painted with one color.

Let us consider the light reflected from a point on the map where only one color **A** is painted. A certain percentage of the light that penetrates through the pigment layer is reflected on the paper material, as shown in Figure 3.1. The total reflected light $C_a(\lambda)$ can be described as a mixture of the light $T_a(\lambda)$ reflected by color **A**

pigment layer and the light $T_p(\lambda)$ reflected by the paper beneath the pigment layer:

$$C_a(\lambda) = T_a(\lambda) + T_p(\lambda) \tag{3.1}$$

Moreover, it is assumed that the reflected lights $T_p(\lambda)$ and $T_a(\lambda)$ are proportional to the amount of light energy that each layer receives. Therefore, the reflection equation 3.1 can be rewritten as

$$C_a(\lambda) = k_a \cdot R_a(\lambda) + (1 - k_a) \cdot P(\lambda) \tag{3.2}$$

where $0 \leqslant k_a \leqslant 1$ gives the percentage of energy that the pigment layer receives, and $R_a(\lambda)$ and $P(\lambda)$ are reflected light spectral distributions when the full amount of light energy is received. For the same reason the reflection equation for color **B** is given by

$$C_b(\lambda) = k_b \cdot R_b(\lambda) + (1 - k_b) \cdot P(\lambda) \tag{3.3}$$

When two colors are painted on the map, for example, when **A** is painted on top of **B**, part of the amount of energy which was solely received by **B** is taken away by **A**. Thus, the strength of the reflected energy distributions of both layers will be reduced to a certain level because of the existence of each other. The energy which was solely applied on layer **A** has to distribute among **A** and **B**. Suppose that k_2 percentage is applied on layer **B**, while the other $1 - k_2$ percentage is on layer **A**. As a result, the

reflection energy distribution on layer **A** should be $(1-k_2)\cdot k_a \cdot R_a(\lambda)$, and that on layer **B** be $k_2 \cdot k_b \cdot R_b(\lambda)$. The energy distribution $C_{ab}(\lambda)$ from the overlapped area is

$$\begin{aligned} C_{ab}(\lambda) &= k_1 \cdot (1-k_2) \cdot k_a \cdot R_a(\lambda) + k_1 \cdot k_2 \cdot k_b \cdot R_b(\lambda) \\ &+(k_1 - k_1 \cdot k_2 \cdot k_b - k_1 \cdot (1-k_2) \cdot k_a) \cdot P(\lambda) \end{aligned} \tag{3.4}$$

where $0 < k_1 < 1$. When two layers of pigments are painted and more energy is absorbed, the reflection strength on each layer is correspondingly weakened. The coefficient k_1 reflects this phenomenon. From the equations above a linear expression can be derived, that is,

$$C_{ab}(\lambda) = k_1 \cdot (1-k_2) \cdot C_a(\lambda) + k_1 \cdot k_2 \cdot C_b(\lambda) \tag{3.5}$$

Note that k_1 depends on k_2. It is found that $k_1 \to 1$ when $k_2 \to 0$, and also $k_1 \to 0$ when $k_2 \to 1$. This means that Equation 3.4 degrades to Equation 3.2 or 3.3 when $k_2 = 0$ or 1 respectively.

The scanner sensors consist of a set of three receptors with sensitivities $s_1(\lambda)$, $s_2(\lambda)$, $s_3(\lambda)$, and the light source spectral power distribution is given by $E(\lambda)$. Thus the RGB value p at a certain location on the map is,

$$p = k \cdot \int E(\lambda) \cdot C(\lambda) \cdot s_i(\lambda) d\lambda \qquad \text{for i} = 1, 2, 3. \tag{3.6}$$

where $C(\lambda)$ is the reflection and k is the scaling constant. Therefore, the RGB values of pixels corresponding to color **A**, **B** and the overlap color of **A** and **B** also obey the linear relationship. That is,

$$P_{ab} = k_1 \cdot (1 - k_2) \cdot P_a + k_1 \cdot k_2 \cdot P_b \tag{3.7}$$

with P_{ab}, P_a, and P_b being vector representations for color **A**, **B**, and their overlapped color respectively.

Based on this reflection model, two application specific heuristics are discovered and given below:

- **Heuristic H.1:** Plotting the vectors representing colors **A**, **B** and their overlapped color into the RGB space, the overlapped color of **A** and **B** will distribute within a narrow wedge shaped space which is defined by color **A** and **B**, as illustrated in Figure 3.2. Since $0 < k_1, k_2 < 1$, $(1 - k_2) \cdot P_a + k_2 \cdot P_b$ falls on the line between P_a and P_b, and P_{ab} is confined in the narrow wedge shaped space defined by P_a, P_b, and the origin of the feature space.

- **Heuristic H.2:** The RGB values of pixels along the boundary areas of two neighboring colors approximately obey a linear distribution. That is

$$P = k_1 \cdot P_a + (1 - k_1) \cdot P_b \tag{3.8}$$

where P_a, P_b are RGB values of two neighboring colors **A** and **B**. Two cases are

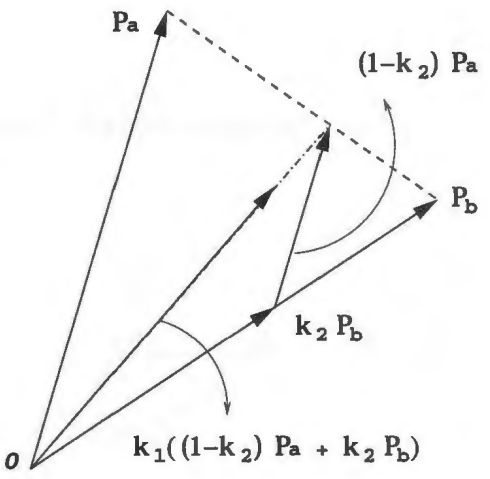


Figure 3.2: Distribution of overlapping colors.

considered: (1) the paper color and a pigment color, (2) two pigment colors. For case (1), Equation 3.2 holds. For case (2), the neighboring areas are regarded as overlapping areas. Since the degree of overlapping is low, $k_1 \approx 1$. With $k_1 = 1$, Equation 3.7 describes exactly a linear relationship.

Visualization of the pixel color distribution with the aid of a 3D tool *Geomview* conforms to these two heuristics.

3.4 Fuzzy Neural Approach

In the color map situation, it is important to embed a priori knowledge about the pixel variation behavior into the proposed color segmentation system as much as possible. The fuzzy neural model has been proven to be a useful tool to construct intelligent systems. The proposed fuzzy neural method for segmentation of color map images is

described below.

3.4.1 Fuzzy Neural Architecture

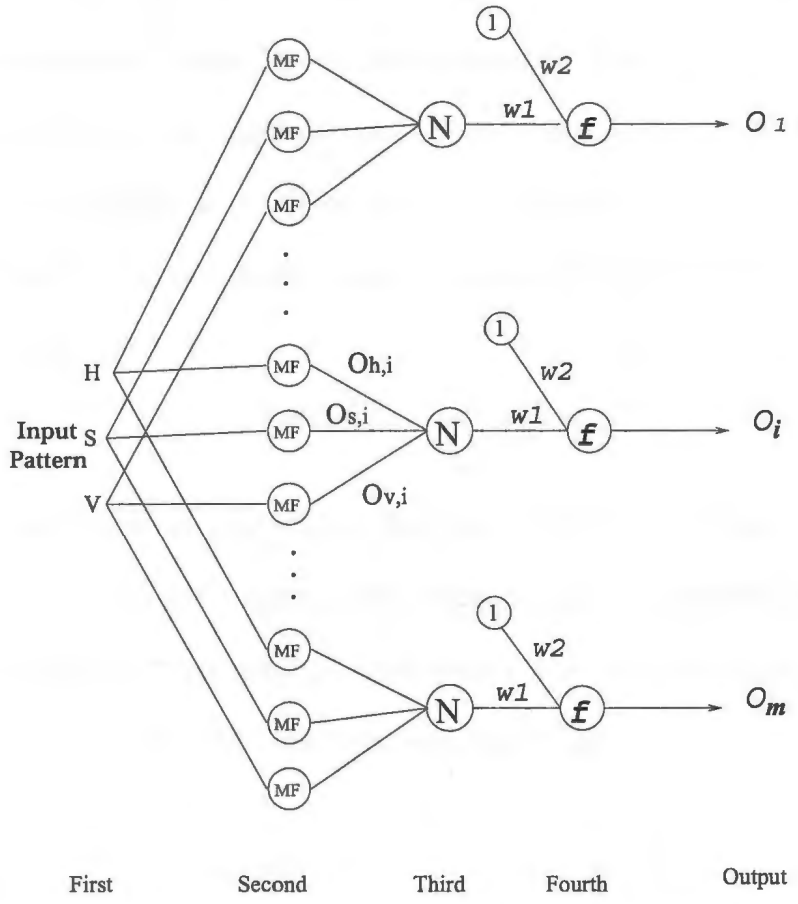


Figure 3.3: A fuzzy neural network to classify m colors.

The architecture of the proposed fuzzy learning model, shown in Figure 3.3, has three inputs and m outputs O_1, O_2, ..., O_m, which correspond to the m colors to be classified. There are four layers in the fuzzy neural network. Nodes at the same

layer have similar functions. The first layer is the input layer, which takes three input values that correspond to the HSV components of a sample pixel. The second layer is designed as a fuzzy member function filter, which measures the membership grade of a fuzzy linguistic value, such as "Hue is around 0(red)". A color cluster i (i=1, 2, ..., m) corresponds to three **MF** nodes in the second layer. The three **MF** nodes connect to an **N** node, the weighted sum of whose output and a constant 1 is taken as the input of an **f** node at the fourth layer. The output O_i of this **f** node suggests the desired target of color i. In this model the generalized bell function is chosen as the member function:

$$MF(x) = \frac{1}{1 + |\frac{x-c}{a}|^{2b}} \tag{3.9}$$

where a, b, c are the parameters which define the bell shape of the function (see Figure 3.4). By iteratively updating these parameters, the learning algorithm can reshape member functions until they suggest cluster centers in the most approximate fashion.

The output of an **N** node in the third layer is given by

$$O_{3,i} = min(O_{h,i}, O_{s,i}, O_{v,i}), \qquad i=1, 2, ..., m \tag{3.10}$$

where $O_{h,i}$, $O_{s,i}$, and $O_{v,i}$ are the outputs of the three **MF** nodes of the second layer. An **N** node ensures that the output is high (close to 1) only when all its three input values (hue, saturation, and intensity) are high. Each output gives the fire strength of a rule, such as

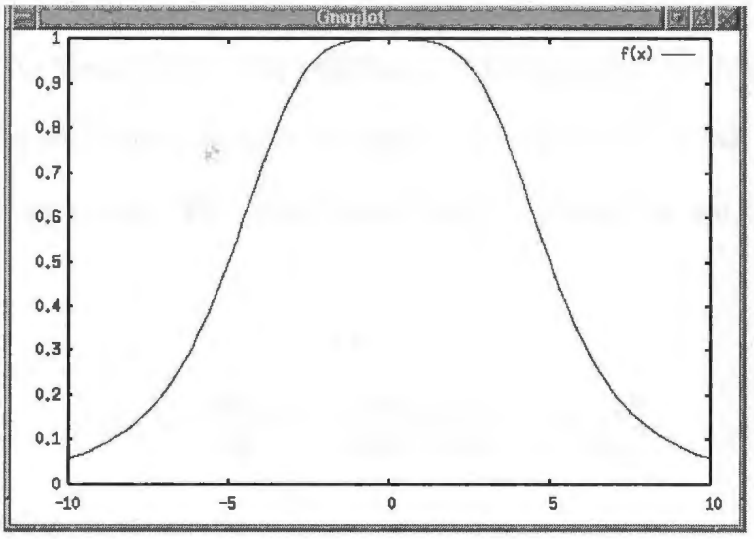


Figure 3.4: A generalized bell function.

IF *Hue is around* H_i, *and*

Saturation is around S_i, *and*

Intensity is around I_i

THEN *Classify the pixel as Color Cluster* i.

Related to a node in the last layer is an adaptive function

$$f(x) = \frac{1}{1 + e^{-(w1 \cdot O_2 + w2)}} \tag{3.11}$$

where *w1* and *w2* are weights associated with the connections between the third and fourth layers. This function is used to enhance the output from the previous layer.

The learning algorithm uses a steepest descent method that is often referred to as backpropagation learning. The error at the output nodes is propagated backwards to

the first layer. The set of parameters to be trained are: $w1$, $w2$ in Equation 3.11, as well as a, b, c in Equation 3.9. The calculation of the gradients for all other weights except a, b, c is well known, so only the error computation for the **MF** nodes at the second layer is given here. The weight corrections Δa, Δb and Δc are defined by the delta rule:

$$\Delta a = -\eta \frac{\partial E}{\partial a} = -\eta \frac{\partial E}{\partial MF} \frac{\partial MF}{\partial a} = \eta \delta \frac{\partial MF}{\partial a} \tag{3.12}$$

$$\Delta b = -\eta \frac{\partial E}{\partial b} = -\eta \frac{\partial E}{\partial MF} \frac{\partial MF}{\partial b} = \eta \delta \frac{\partial MF}{\partial b} \tag{3.13}$$

$$\Delta c = -\eta \frac{\partial E}{\partial c} = -\eta \frac{\partial E}{\partial MF} \frac{\partial MF}{\partial c} = \eta \delta \frac{\partial MF}{\partial c} \tag{3.14}$$

where η is the learning rate parameter, and δ is the error at this layer, which is defined by

$$\delta = -\frac{\partial E}{\partial MF} \tag{3.15}$$

The partial derivatives of the member function **MF** can be calculated as follows:

$$\frac{\partial MF}{\partial a} = \frac{2b}{a} \frac{|\frac{x-c}{a}|^{2b}}{(1+|\frac{x-c}{a}|^{2b})^2} = \frac{2b}{a} MF(1-MF) \tag{3.16}$$

$$\frac{\partial MF}{\partial b} = -2\ln|\frac{x-c}{a}|\frac{|\frac{x-c}{a}|^{2b}}{(1+|\frac{x-c}{a}|^{2b})^2} = -2\ln|\frac{x-c}{a}|MF(1-MF) \qquad (3.17)$$

$$\frac{\partial MF}{\partial c} = \frac{2b(x-c)}{a^2}\frac{|\frac{x-c}{a}|^{(2b-2)}}{(1+|\frac{x-c}{a}|^{2b})^2} = \begin{cases} 0, & \text{if x = c,} \\ \frac{2b}{x-c}MF(1-MF), & \text{if x} \neq \text{c.} \end{cases} \qquad (3.18)$$

3.4.2 Self-Adjustment Architecture

In a supervised learning system, the performance of a network depends on sample patterns provided. If samples are not sufficiently representative, the trained network would not be powerful enough to classify all the color pixels. Thus, it is expected that sample points should catch pixel variations of the color they represent. It is necessary to obtain samples from boundary and overlapping areas where most of the pixel variations take place. When preparing samples, there is a dilemma. Boundary and overlapping areas are places where different pixels are densely mixed. They provide the essential information for capturing pixel variations, but it is hard to discern different samples. Even when extreme care is exercised by a user to select samples from these areas, some "noises" may still creep in. These noises will greatly degrade the network's performance. If the noise level rises to a certain degree, the network training will not converge. Therefore, the very nature of map color segmentation

necessitates a classification method that should not strictly require all the representative samples be collected and grouped into distinctive clusters by a human operator in advance. The method should be able to incorporate a priori knowledge into the network and have the ability for self distinguishment and adjustment. The self adjustment structure developed below meets these needs. It exploits the two heuristics described in Section 3.3.

The problem and the self-adjustment architecture are described as follows. Suppose that N colors, C_1, C_2, ..., C_N, need to be classified, and that $N + M$ sample pixel sets, S_1, S_2, ..., S_{N+M} can be obtained. These sample sets are categorized into two groups. The first one is the pure color group $G_p = \{S_1, S_2, ..., S_N\}$, which consists of sample sets obtained from areas painted with solely one color. The second is the composite color group $G_c = \{S_{N+1}, ..., S_{N+M}\}$, in which each of the M sample sets is obtained from "vague" areas where most of the color pixel variations are difficult to capture. An element in G_c is called a "vague" cluster in which the "vague" pixels come from overlapping or boundary areas. It is difficult to determine exactly which cluster a "vague" pixel belongs to when a human operator is preparing samples. The samples of a composite color group thus cannot be readily used to train the neural network. This causes a design problem. If G_p samples are used in the training, the pixel variations cannot be captured. Additionally, the G_c samples have much pixel variation information hidden behind but cannot be exploited by a traditional supervised learning method. Our solution is to incorporate a self adjustment architecture

into the fuzzy model introduced in Section 3.4.1. Hence G_c samples can now be included in the training process. In this way it enhances the network's ability to capture pixel variations. The function of this fuzzy model is to gradually find the cluster centers, while the purpose of the self adjustment architecture is to dynamically adjust the sample sets with the environment change.

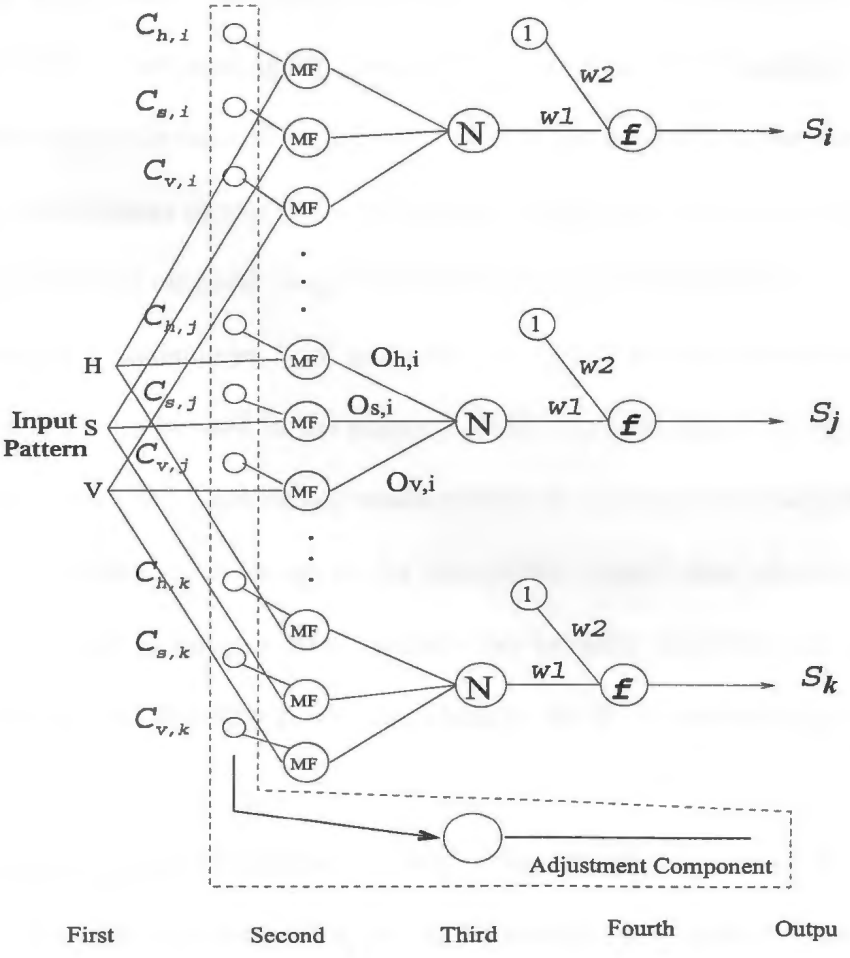


Figure 3.5: H.1 structure shown inside the dashed line area.

Two types of self adjustment structures have been developed: H.1 structure and

H.2 structure, which aim to exploit the two heuristics described in Section 3.3 respectively. Figure 3.5 schematizes a typical H.1 structure, which is designed to capture pixel variations in an overlapping area. The H.1 structure is built on the three subnets corresponding to the three clusters S_i, S_j, and S_k, where S_k represents the overlapping cluster of S_i and S_j. An adjustment component is incorporated into the network in Figure 3.3. Each small circle inside the dashed line box represents the cluster center component of its corresponding MF function (i.e., parameter **c** in Equation 3.9). The adjustment component takes these member function parameters from the second layer as inputs and produces signals (refer to Equations 3.19 - 3.21 on how the adjustment component produces outputs) to guide samples to one of the three clusters S_i, S_j, or S_k. Suppose that a sample set S_k of group G_c that is from an overlapping area of two colors **A** and **B** is to be used in the training. Since pixel samples in S_k are obtained from "vague" areas, it cannot be told exactly which color cluster each sample belongs to in advance. However, according to the type of the "vague" area, those color clusters over which the S_k samples may distribute can be easily told. Samples in S_k may belong to one of the following three color clusters: **A**, **B**, or overlapping color of **A** and **B** (let us denote it as $O_{\{ab\}}$).

In a supervised training process, a target value should be assigned to a sample before it is fed into the network. While for S_k samples, no predefined target value is available. Therefore, an adjustment component is employed to guide the samples to their proper target color clusters. On each iteration of the learning process, an S_k

sample will be temporally guided to one of three color clusters **A**, **B**, or $O_{\{ab\}}$, depending on which is the most favored by the adjustment component. The adjustment component uses the following criteria to set the direction where the S_k sample should go. From Heuristic H.1, it is known that $O_{\{ab\}}$ sample values should be restricted in a narrow wedge shaped space defined by **A** and **B**. The cluster centers corresponding to **A** and **B** have been suggested by the parameters of the member functions of the second layer. In Figure 3.5, parameters $C_{h,i}$, $C_{s,i}$ and $C_{v,i}$ suggest the cluster center of color **A**, $C_{h,j}$, $C_{s,j}$ and $C_{v,j}$ suggest that of **B**, and $C_{h,k}$, $C_{s,k}$ and $C_{v,k}$ suggest that of $O_{\{ab\}}$. To decide if $O_{\{ab\}}$ is favored, the adjustment component checks if all of the following three inequalities can be satisfied:

$$dist(H_s, C_{h,i}) < dist(C_{h,i}, C_{h,j}) \tag{3.19}$$

$$dist(H_s, C_{h,j}) < dist(C_{h,i}, C_{h,j}) \tag{3.20}$$

$$V_s < min(C_{v,i}, C_{v,j}) \tag{3.21}$$

where H_s and V_s are the hue and intensity values of the sample. dist(U, V) is defined as follows

$$dist(U,V) = \begin{cases} \mid U - V \mid & \text{if } \mid U - V \mid \leqslant 180 \\ 360 - \mid U - V \mid & \text{if } \mid U - V \mid > 180 \end{cases}$$

The above inequalities select those samples that fall into the narrow wedge shaped

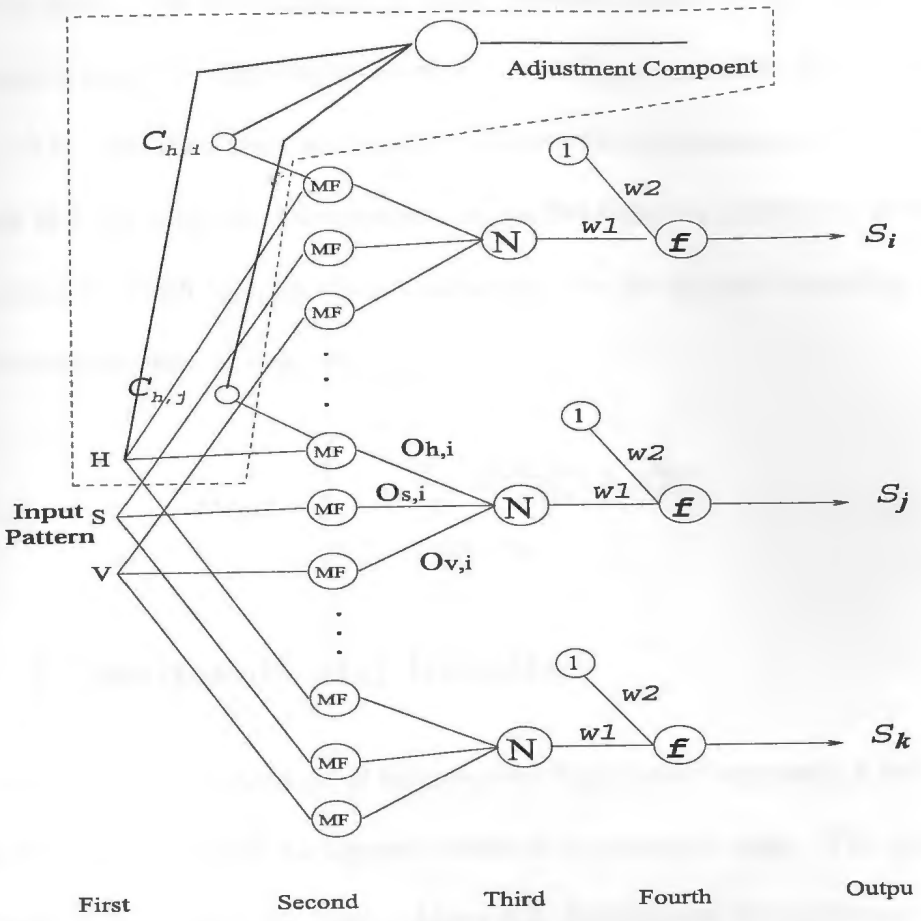


Figure 3.6: H.2 structure shown inside the dashed line area.

space and have intensity values smaller than the samples of clusters S_i and S_j. If these conditions are satisfied, the sample is treated as a color $O_{\{ab\}}$ sample; otherwise, the adjustment component checks the outputs of nodes N_i and N_j. The node with a sufficiently large output will be favored and the sample will be treated as belonging to the corresponding cluster.

An H.2 structure is shown in Figure 3.6, which captures pixel variations along

boundary areas. It is used to adjust samples between clusters S_i and S_j, which are the sample sets of two neighboring colors. $C_{h,i}$ and $C_{h,j}$ are member function parameters used by the adjustment component to decide the adjustment criteria. Due to Heuristic H.2, the adjustment component can use the following criterion to determine the direction in which samples along a boundary area are adjusted according to the target value computed as follows:

$$target = \begin{cases} S_i, & \text{if } \frac{dist(H_s, C_{h,i})}{dist(C_{h,i}, C_{h,j})} < \frac{O_i}{O_i + O_j} \\ S_j, & \text{otherwise} \end{cases}$$

3.5 Experiments and Results

To demonstrate the effectiveness of the proposed fuzzy-neural approach, a prototype system has been developed to segment colors on topographic maps. The system is running on SUNOS 4.1.4 and Redhat Linux 6.2. Experiments are performed on six sample color images taken from some topographical and city maps.

The first map to be classified is a portion of a map with a scale of 1:250,000. The image has a size of 154×136 in pixels, which is scanned into the computer by an HP laserjet II scanner with a resolution of 300dpi. Pixel samples for each color cluster are manually prepared using an image ROI extraction utility.

Image processing tools GIMP and KHOROS[1] are used to extract color pixel sam-

[1]GIMP is a freely distributed image manipulation tool (refer to www.gimp.org for more information). KHOROS is an integrated system of tools and programming environment for image processing and visualization.

ples from raw images. An image is magnified first using GIMP, so that it is easier for human operators to locate and select small areas from maps. For each color, five or six small areas from different regions of a map are collected. In this way a sufficient number of pixel samples are selected. Each sample cluster contains 180 to 250 pixels. Then KHOROS is used to convert the pixel samples into the input format used by the color segmentation system.

Six colors often used on maps: red, green, blue, brown, black, and white, are classified in our experiment. Six G_p sets and two G_c sets have been identified. One of the two G_c sets is collected from overlapping areas of red and blue, the other one from boundary areas of red and white colors. With these G_c sets, the performance of the self adjustment structures described in Section 3.4.1 has been tested. It took 267 seconds to finish the training on a Pentium IV (1.4 MHz) machine running RedHat Linux 8.0. The results show that most of pixels around "vague" areas can be correctly classified due to the presence of the self adjustment structures. The original image (Figure 3.7(a)) shows a scene with red, blue and brown line features. Figure 3.7(b)–(g) shows some of the segmentation results. Experiments were also performed on the sample portion of the map, but with higher resolutions (600dpi and 1200dpi). The result shows little difference from those scanned with 300dpi, because there is no significant change in the patterns of pixel variation.

More experiments are carried out on several city maps of scale 1:500,000. These maps show relatively complex map features, since several thin line features, such as

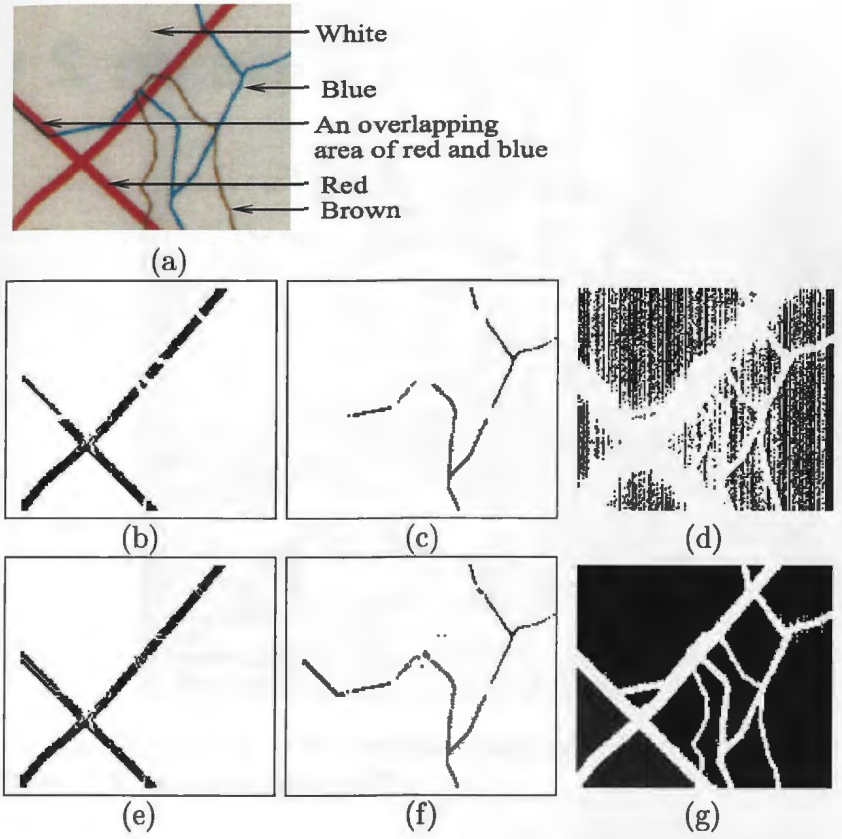


Figure 3.7: (a) Original image (Portion of National Atlas of Canada Series, 1981. Natural Resources Canada). (b) Red color, without using H.1 structure. (c) Blue color, without using H.1 structure. (d) White color, without using H.2 structure. (e) Red color, by using H.1 structure. (f) Blue color, by using H.1 structure. (g) White color, by using H.2 structure.

roads, rivers, as well as isolines, are present. These experiments aim to separate the maps into a set of color layers, especially layers with road and river networks. An ideally segmented layer would be one that has all the major areas painted with the target color correctly shown and does not include significant noise.

In one of the experiments, the proposed approach is applied to separate the map

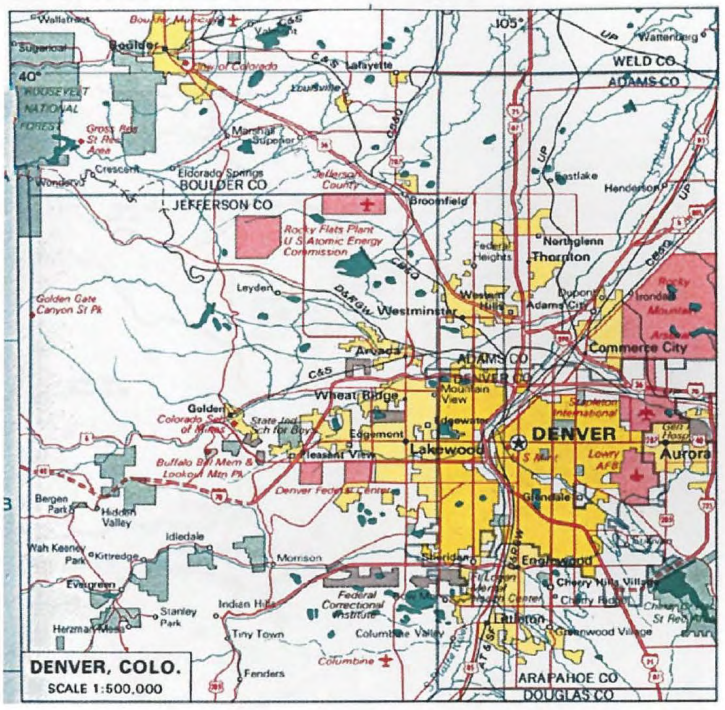


Figure 3.8: A portion of the regional map of Denver, Colorado. Original scale 1:500,000 U.S. National Atlas 1970.

image (945 × 1024 pixels) shown in Figure 3.8 into the following six color layers: red, blue, black, white, light yellow, and yellow. Since the effects of overlapping and boundary colors are disregarded, a fuzzy neural network similar to the one in Figure 3.3 is constructed, yet without any self adjustment components. As the most frequently seen feature on topographical maps, the road network (the red layer) reaches almost all regions and interacts with other features. Therefore, it is the most difficult layer to extract. The effectiveness of the proposed approach can be demonstrated using the segmentation result of the red layer. The extracted red layer is shown in Figure 3.9. It can be seen that many parts of the red color areas are faintly recognized;

and in many places, road lines become fragmented.

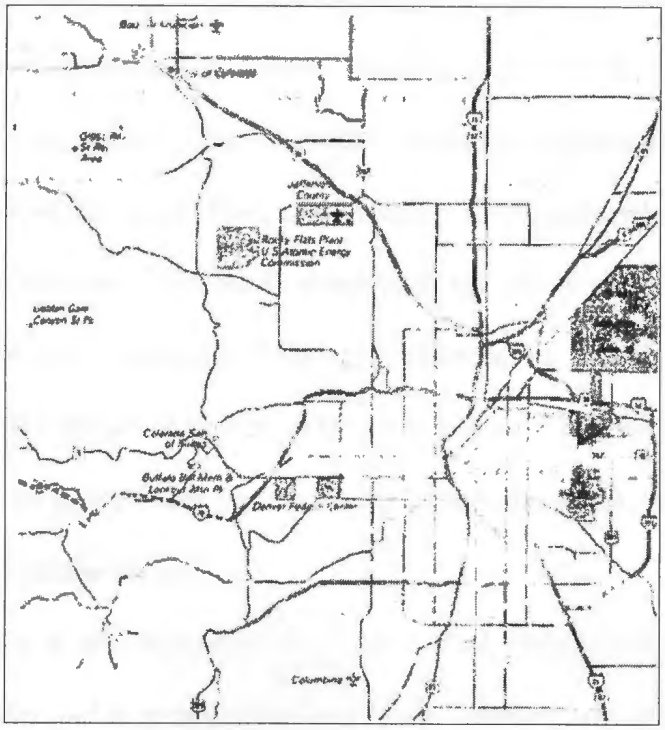


Figure 3.9: The red color layer obtained using a fuzzy neural network without self adjustment components.

In a further experiment, the overlap of red and light yellow, and that of red and yellow are taken into consideration. Pixel variations of two neighboring colors, red and white, are considered as well. As a result, three H.1 and one H.2 self adjustment components are added to the fuzzy neural network. The obtained red layer is shown in Figure 3.10. A remarkably improved result is achieved. The roads are drawn with very thin lines with a width of one or two pixels. The pixel values vary towards their neighboring colors. It is natural for an H.2 structure to capture such variations.

Although some road lines are still broken by gaps, this happens much less often, and the gaps are smaller. Through observation, we can still find some fine line features that can be identified by human eyes but not by the neural networks with self-adjustment components. This is a very interesting phenomenon. The author specifically extracted the pixels from these features and compared their values with those from other clusters. The result shows that the values stray away from their target cluster and are close to one of the other clusters. As a result, the variation is not captured by the proposed system. The possible reason may be that human eyes have the ability to adjust their distinguishing criteria based on the context, while computer based systems do not.

One limitation of the supervised methods is that training samples have to be prepared manually and a target value has to be assigned to each sample by the user. The learning is supervised to improve results gradually based on the feedback of each epoch. Unsupervised methods do not have the burden to associate desired outputs with input samples. However, an unsupervised method receives no feedback on an outcome. Learning has to rely on the existing intrinsic structure of the sample data to group pixels into a number of clusters. This intrinsic structure does not necessarily reflect the distribution of color clusters, because (1) pixel variations of color images are difficult to predict, and (2) different types of color images show totally different characteristics in pixel variations. By including representative pixels in training samples, a priori knowledge about pixel clusters can be captured. For

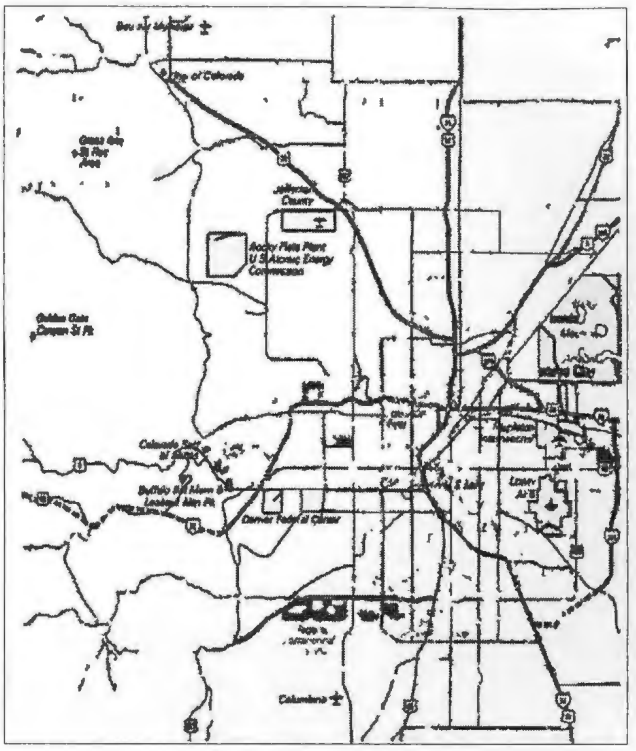


Figure 3.10: The red color layer obtained by taking account of overlapping and boundary pixel variations.

those color images (such as map images) that have very thin line features (one or two pixels wide) and require delicate image segmentation methods, it is believed that a priori knowledge about the color cluster distribution makes a great difference in segmentation results. Another issue with the unsupervised methods is that the proper segmentation resolution (the number of target clusters) has to be known in advance. If the resolution is too high, the pixels of a single color may be scattered into many clusters. On the other hand, if the resolution is too low, one cluster may contain pixels from multiple colors.

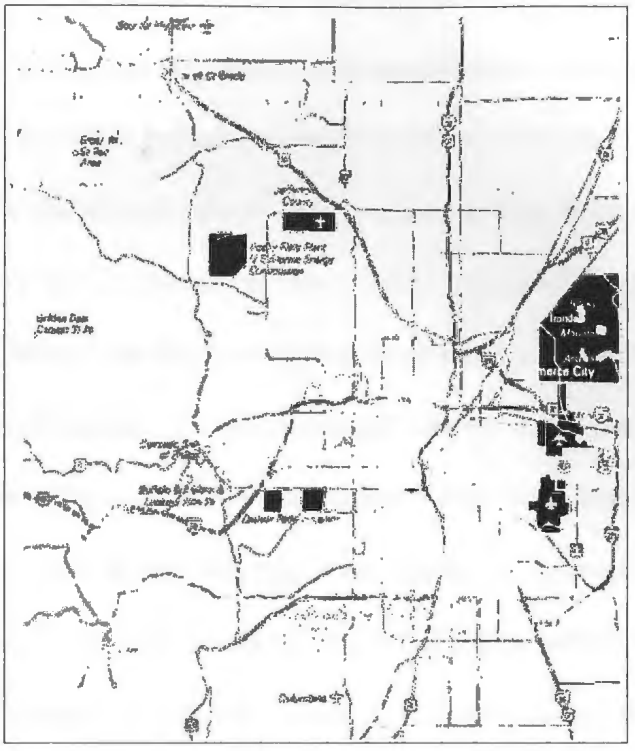


Figure 3.11: The red color layer obtained using fuzzy c-means clustering method.

As a comparison, the fuzzy c-means method was used to extract the red color layer. Experiments were carried out with different predefined cluster numbers, among which the best result is obtained using cluster number 7 (shown in Figure 3.11).

As a further comparison, the color image segmentation algorithm provided by Comaniciu and Meer [25] was applied to separate the map of Denver. This method is a general technique that avoids drawbacks of the traditional clustering methods and achieves efficient and robust color feature extraction. It takes three parameters characterizing three levels of segmentation resolutions: undersegmentation, oversegmentation, and quantization, which mean low, intermediate, and high resolutions,

respectively. 5 clusters are obtained with undersegmentation option, 39 clusters with oversegmentation option, and 50 clusters with quantization option. The quantization result is far from acceptable because, through posterior analysis, it is found that the red color pixels are distributed into at least 8 clusters. The undersegmentation and oversegmentation results are shown in Figure 3.12. Human observation is needed to pick the proper clusters from the oversegmentation result to construct the red layer. The undersegmentation option shows good results except at the overlapping areas. It seems the pink pixels are considered closer to red pixels than those from the overlapping areas, because most of the pink pixels are shown in the result. One interesting phenomenon is that some black pixels are also classified as red (see the word "DENVER" at the bottom-left corner). The reason is that only five colors are determined to be significant ones. Black color is not one of them, since the total number of black pixels is much smaller. Some black pixels are assigned to the red cluster, because the algorithm determines that it is the closest cluster to these black pixels.

In some further experiments, the trained network built for one map image is also applied to some other maps produced using similar methods. This shows that significant retraining time can be saved. Some of the experimental results are presented in Appendix D. The quality of the results depends on the patterns of pixel variations. If a map is printed under a different condition, the neural network usually has to be retrained. By comparing the segmentation results with the original map, it can be observed that except in the downtown areas (yellow colored areas), most red color

Figure 3.12: The red layers obtained based on Comaniciu and Meer's algorithm. (a) Constructed with 1 of 6 clusters produced by undersegmentation; (b) Constructed with 5 of 26 clusters produced by oversegmentation.

pixels can be extracted. The downtown areas happen to be places where multiple features (road, river, city boundary and texts) interweave, so that the patterns of pixel variations are much more difficult to predict. This means to extract such pixels, it is still necessary to retrain the network with samples picked specifically from these areas. Better results may be achieved by training the network with pixel samples from different maps and then applying it to the maps individually. Better results are obtained for blue color features with a width of more than three pixels. However, very thin blue color features (one or two pixels wide) are difficult to extract. The reason is that the intensity values of the pixels from such blue lines are relatively higher that those from other blue areas due to the contrast with the white background. This means that to capture the pixel variations represented by these pixels, it is necessary to pay more attention to such thin lines to make sure pixel samples from these thin lines constitute a significant percentage.

Experiments were also conducted on maps scanned in resolutions 600dpi and 1200dpi. The results are very similar to those obtained from previous experiments with the resolution 300dpi. This shows that the resolution 300dpi is sufficient enough to capture the typical pixel variations.

From the experiments described above, it can be concluded that by taking into consideration overlapping and boundary pixel variations, much more detailed and accurate segmentation can be obtained. The limitation of the method is that some noises are introduced. It can be seen that although the result shown in Figure 3.10 is

much better than that in Figure 3.9, it still includes a few spots that should not be classified as red. To determine what contributes to the misclassification, histogram analysis was performed on the pixel values of those error spots. One reason is that some totally unrelated small areas show very similar pixel values. Because these pixels are usually mixed in their context, human eyes are not able to perceive their difference from the context. Another reason is that areas with overlapping colors usually exhibit more unpredictable variances. While pixel variations in overlapping areas are captured, some other pixels that have similar pixel values but are not from overlapping areas, are picked up. Fortunately these cases do not happen very often, and if they do, they do so at a few small and isolated spots. A large part of the noises can be removed using proper noise removal techniques, such as mathematical morphology operators.

The proposed color segmentation method can be directly applied to large size maps. The processing speed will be proportional to the map size. More dynamic color variations may occur on large maps. However, this will not affect the effectiveness of the proposed method, because the color printing process is strictly controlled and such variations have a narrow range. In addition, the variations can be captured by adopting effective sampling methods.

3.6 Conclusions

In this dissertation, a novel fuzzy neural network combined with self adjustment components has been presented for solving color segmentation problems on maps. The color segmentation system built based on the proposed method can be applied to different color maps printed on paper. This method can also be utilized for other color segmentation tasks, such as segmentation of colors on pictures produced with techniques similar to those used on maps. A heuristic similar to the one that is used to treat areas with two overlapping colors may be developed to deal with areas with three or more overlapping colors. Interactions of multiple pigment layers make the pixel variations in areas where three or more colors overlap more difficult to predict and capture. Fortunately such interactions do not have unlimited effect on color variations, because most of the reflected light energy is contributed by the few pigment layers on the top. The idea of self adjustable fuzzy model can also be used in some feature space analysis problems, where only vague clusters can be obtained. In addition, with the existence of self adjustment structures, the network is a so called " partially " supervised learning network, whose dynamic properties still need to be further explored.

Chapter 4

Conceptual Modeling and Description Logics

In any application, one of the most important tasks is to capture and express the most relevant information of the problem to solve. Frequently the design of applications is described inadequately and equivocally in natural languages. Different methods are developed in an attempt to accurately model application domains in detail. Most models, including Description Logics (DL), view the world as a collection of interrelated components, so that it is easy to build and extend information systems. DL languages push the limits further by introducing terminological and assertional reasoning facilities to automatically reveal subsumption relations between concepts and allow consistency check on DL representations. Thus many of the relationships of concepts that may not be intuitively recognized by model builders can be discovered. The purpose of this chapter is to present DL as a conceptual modeling tool. A brief

review of the motivations of concept modeling in the high-level map understanding will be presented first. Description Logics languages are viewed as the tool for knowledge representation. General issues concerning using DL as the modeling mechanism for map understanding are then discussed. In Section 4.4, a conceptual language $\mathcal{GALC}(\ell, \mathcal{D})$ is proposed by the author, which extends $\mathcal{ALC}(\mathcal{D})$ by allowing roles that represent n-ary relations.

4.1 Conceptual Modeling

All the currently existing techniques for high-level map understanding use domain specific knowledge. However, those knowledges are not readily applicable to map understanding systems. It is necessary to decide what knowledge is needed and how to represent it. It is a well known problem solving strategy to identify a set of objects, their properties and interactions. In this section, an object-relationship graph approach that aims to form a concise description of map features and their relationships will be discussed. As a graphical notation, such a graph offers a natural and flexible specification of map information.

There are two types of map objects: primitive and derivative. Primitive objects are those geometric features obtained from the low-level image processing algorithms. They can be line segments, areas, characters and digits. Primitives are characterized by object properties, such as location, size, direction, and angle. Derivative objects are features derived from primitive objects or other derivative objects in the interpretation

process. A derivative object can be, for example, a road network, a river network, or a map scene. Some relationships represent spatial information, such as "connect", "on", "touch", "near" and "intersect". Others represent aggregation information. For example, the "made of" relationship is used to represent the fact that a map object is made of a number of river networks or road networks.

An object-relationship graph consists of a set of nodes and a set of arcs that interconnect the nodes. There are two types of nodes: object nodes and relationship nodes. A relationship node represents the existence of a certain relationship between two object nodes or among more than two object nodes. An arc only exists between an object node and a relationship node, since the arc represents the participation of the node in a relationship.

Note that the concept of object node refers to a category of objects which possess certain properties in common, not just a single object instance. The concept of relationship node represents the category of relation instances among objects from different categories. Thus a relationship node refers to a set of relationship instances. Each of the object nodes that is connected to a relationship node **R** is said to participate in **R**.

Let us take as an example a relationship node **MADE_OF** among three object nodes **ROAD_NETWORK**, **ROAD_SECTION**, and **ROAD_JOINT**. This relationship node indicates how a road network is built up by a set of road joints and a set of road sections. Each relationship instance associates a road network object, a

set of road joint objects, and a set of road section objects. This example is illustrated in Figure 4.1, where each relationship instance I_i is shown connected to the objects that participate in I_i. In Figure 4.1, road network r_1 is made of road joints j_1, j_2, j_3, j_4, and road sections s_1, s_2, s_3, s_4, s_5, and road network r_2 is made of j_5, j_6, j_7, j_8, j_9 and s_6, s_7, s_8, s_9, s_{10}.

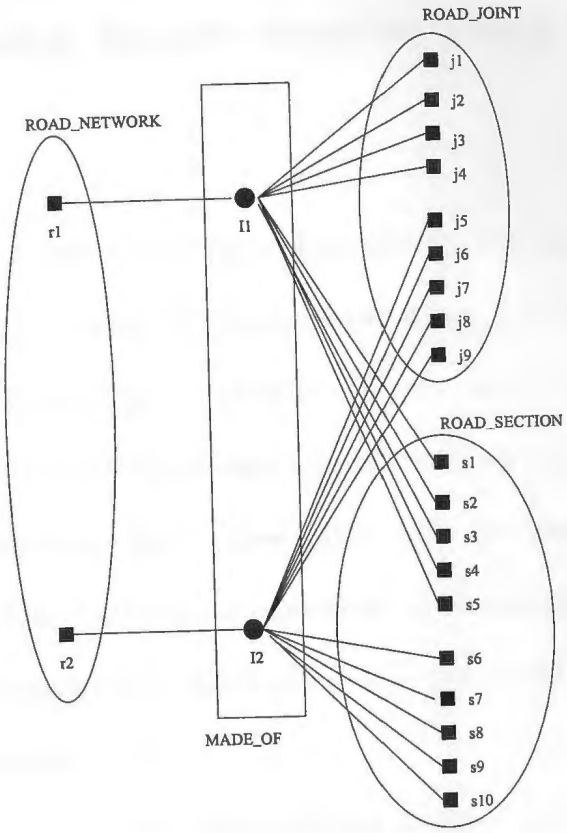


Figure 4.1: Two instances of MADE_OF relationship

In many conceptual representations, the relation edges are drawn as directed. The edge directions indicate the different roles of objects in the relations. For example, we can say that "a map instance contains a road network instance". But the description

from the other direction is not correct —— we cannot say "a road network instance contains a map instance". In some conceptual graphs in this dissertation, the direction information is ignored, because the role of each participating object is specified in the semantics of the relationship node.

4.2 Knowledge Representation Using Description Logics

In order to acquire knowledge for map interpretation, it is essential to study the information contained in maps. We must be aware that it is extremely difficult to produce an ideal semantic representation in one pass. It is always necessary to go through several iterations before sufficient information is captured. The first iteration may result in a somewhat rough representation with the most basic and directly perceivable information. Later on the representation is refined or enhanced through further analysis. Figure 4.2 is an object-relationship graph with some significant map objects and relationships.

In Figure 4.2, spatial information of different abstract levels has been described. Structural objects and relationships are concrete objects that exist on maps; by intuition, they are usually identified and mapped from real world into the object-relationship graph with little abstraction. All the currently existing methods represent this kind of knowledge explicitly in one way or the other. However, there

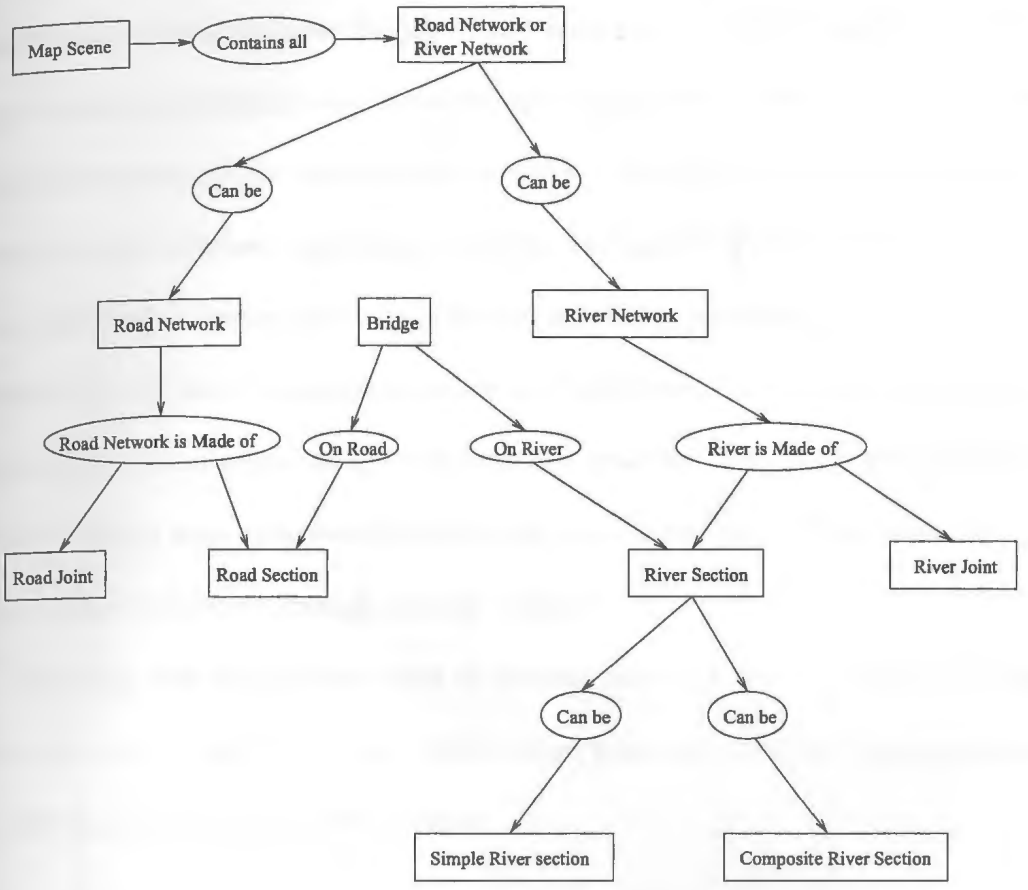


Figure 4.2: An object-relationship graph

are some categories of knowledges which are usually selectively incorporated in algorithms, and thus are not explicitly represented. In the setting of map understanding, this kind of implicitly represented knowledge is hard to be shared and reused in other situations. Consider, for example, the algorithm to make up a gap in a broken line. Such an algorithm is usually implemented with a procedure that examines certain spatial relations, such as line directions, acceptable distances, and the context. The relevant spatial knowledge is implicitly represented in the procedure. The procedure can gain some flexibility by taking user-defined parameters and thresholds. However,

this is not sufficient to make the procedure adaptive to complex scenarios. When a large number of phenomena are involved, it is not possible to encode the situational activities and control strategies into parameters. If the spatial knowledge is separated from the procedure and represented explicitly, the algorithm will become more adaptive. Description Logics provide us a tool to explicitly represent spatial information. Unlike DL, rule based representations are not logic based. They are basically derived from human experiences and generalizations of practical problem solving strategies. The rule based map understanding is one alternative for explicit representation, but it is still not formalized enough and not verifiable.

Although difficulties always exist in determining what kind of objects and relationships may be described in an explicit form, generally the following categories of knowledges deserve to be made explicit:

- *Objects and relationships that have to do with the goals of the interpretation.* For example, consider a request to query the cities near a certain road. The "city" objects and their relations with the "road" objects should be explicit in the object-relationship graph.

- *Concepts that describe partially recognized objects.* During the interpretation process, many instances are made explicit through an incremental recognizing process. For example, the concept of a road network is not established in one step. Part of it is recognized first, then the rest of it is gradually built up over a number of steps.

- *Concepts that represent the state of interpretation.* The interpretation of maps involves complex sequences of inferences to achieve its goals. Sometimes we have to draw conclusions about the status of the currently recognized objects. One example is that we may need to explicitly represent the fact that "no more road sections connected with the current partially recognized road network can be found".

- *Concepts about condition and action information.* For example, when line features are tracked, certain unexpected situations may emerge, such as a gap or a city symbol breaking a line. Explicitly representing the conditions and subsequent actions to be taken will guide the system on how to proceed with the interpretation.

4.3 Formalization of Semantics

Figure 4.2 only gives a scheme of the objects and relationships. It is equally important to properly represent the semantics of those connected nodes. The semantics of object nodes are given by their attributes. The relationship nodes cannot be defined without specifying their participating object nodes. In order to accurately capture their semantics, the relationship nodes have to be formally defined.

In this dissertation, a fully formalized theory, Description Logics, is applied to accurately specify semantics of map features. This section gives a brief introduction

to a particular Description Logics system $\mathcal{ALC}(\mathcal{D})$ (there are a number of variants) first proposed by Baader [38]. An important point is that there exists a sound and complete algorithm that decides the consistency of a knowledge representation based on DL. This distinguishes the proposed method from traditional methods. The maintenance of consistency of the knowledge representation of most existing solutions depends on the designer's personal experience rather than on a reliable and verifiable formalism. In Chapter 5, we extend $\mathcal{ALC}(\mathcal{D})$ with more relation constructs so that some constraints existing among objects can be accurately captured.

4.3.1 Implicit vs. Explicit Representation

In some respects, the power of knowledge based systems lies in their ability to discover the implicit meaning derived from the explicit representations. Nonetheless, solving a problem does not mean to uncover all implicit knowledge in a domain; it is in fact not necessary and not possible. A balance between implicit and explicit representations has to be maintained.

To decide what kind of knowledge is suitable to be represented implicitly or explicitly depends on the nature of the problem. Some knowledges are suitable to be sliced into smaller-grained and manipulatable pieces, since diverse interactions and behaviors exist among the elements, while some other knowledges are better put into the implicit category. This happens when the interactions among the elements are uniform and it is suitable to use some serialized batch processing. For example,

low level image processing tasks are usually carried out using procedural approaches. Declarative representation is not suitable for describing pixel level knowledge. The knowledge representation becomes too granulated if each pixel in an image is specified with a declarative symbol.

Formalized approaches, especially explicit representations, offer the following benefits:

1. Capture of details. Users have easy access to object properties and relations in which they are interested. The information users need will be readily available for manipulation if it is expressed explicitly.

2. Reuse of knowledge. Explicit representations make it easier to compare the vocabulary and terms used to express concepts and relationships, so that the same knowledge representation can be reused in similar situations.

4.3.2 Description Logics

Description Logics can be used to describe an application domain in a structured and well-understood way. A DL representation works by first formalizing the relevant concepts in the application domain. Through *concept expressions*, concepts can be defined using atomic concepts and roles. Concepts can be considered as unary predicates that are interpreted as sets of individuals. Roles are binary predicates that are interpreted as binary relations between individuals. The concepts are used to specify objects and their properties, while the roles specify relationships among concepts.

What makes $\mathcal{ALC}(\mathcal{D})$ advantageous over other pure abstract logic representations is that concrete domains and predicates on those domains are integrated into conceptual languages. This is especially important to map understanding system modeling since it involves a lot of pixel level calculations. As stated in the previous section, if pixel level information and operations are explicitly represented, the resulting definition and model would be inefficient and too complex to manage. The map itself is a multi-abstract level and multi-facade phenomenon. It contains both overview and detailed information.

Readers can refer to Appendix A for the basic definitions and results of Description Logics. Some basic definitions and notations will be recalled. The reader is assumed to be familiar with the non-formalized set theory and theory of first order logic. For a more detailed study of Description Logics, the reader is referred to [38].

To give examples of DL representations, suppose that company and manufacturer are two concepts. Then company $\sqcap$ manufacturer and company $\sqcap \neg$manufacturer are two concept terms describing those companies that are in manufacturing business and those companies that are not, respectively. Suppose, in addition, that a role $\underline{hasProduct_R}$ and a concept car are introduced. The concept terms, company $\sqcap$ $\exists \underline{hasProduct_R}.$car and company $\sqcap$ $\forall \underline{hasProduct_R}.\bot$, can be formed, denoting the companies that produce cars and those companies that do not have any products. If only companies with a large enough size are considered, this situation can also be represented by concept expressions using predicates on concrete domains. For instance,

an expression of the form company $\sqcap \geqslant_{50}$ (EmployeeNumber) can be used to describe companies with no less than 50 employees. Here $\geqslant_{50}$ denotes a unary predicate, $\{n \mid n \geqslant 50\}$, on the natural numbers. Expressing such properties directly with reference to a concrete domain seems to be easier and more natural than encoding them into abstract concept expressions [38]. The introduction of concrete domains and predicates on these domains enable us to incorporate procedural knowledge representations. In the context of map understanding, discoveries of many spatial relations rely on pixel level computations and computations of geometric attributes, which are difficult to handle at the abstract logical level. The set of concrete objects is denoted by **OD**, and the set of abstract objects denoted by **OA**.

Next, two concrete domains to be used in modeling map objects and their relationships are presented. Note that we use the first quadrant of the coordinate system to represent map information, without loss of generality.

1. The concrete domain $\mathcal{POINT}$

   The domain of $\mathcal{POINT}$ is defined as the set of all pairs of coordinates in a two-dimensional plane. That is, $\text{dom}(\mathcal{POINT}) = \mathbf{PT}\ \{(s,t) | s \in \mathbf{I}, t \in \mathbf{I}\}$, where **I** is the set of non-negative numbers.

   pred($\mathcal{POINT}$) = { identical, proximate, not_identical, not_proximate, $Top_{\mathcal{POINT}}$, $Bottom_{\mathcal{POINT}}$ }

   identical is a binary predicate denoting that two points are identical, and proximate is a binary predicate denoting that two points are very near each other.

Usually two points are said to be proximate if they are within a predefined distance of each other. The predefined distance can be tuned according to various situations of maps, such as types of map features and relations. It is not directly dependent on the scale of the map.

$\mathsf{not\_identical} = \overline{\mathsf{identical}}$

$\mathsf{not\_proximate} = \overline{\mathsf{proximate}}$

2. The concrete domain $\mathcal{GEOM}$

We intend to use $\mathcal{GEOM}$ to deal with facts concerning simple geometric objects (points and polylines) on a 2-dimensional plane. Let **PLine** be the set of all possible polylines in a 2-dimensional plane. The domain of $\mathcal{GEOM}$ is defined as the joint set of dom($\mathcal{POINT}$) and **PLine**, in symbols dom($\mathcal{GEOM}$) = dom($\mathcal{POINT}$) $\cup$ **PLine**.

pred($\mathcal{GEOM}$) = { is_point, is_not_point, is_polyline, is_not_polyline, join_at, not_join_at, endpoints_of, not_endpoints_of, intersect, not_intersect, $Top_{\mathcal{GEOM}}$, $Bottom_{GEOM}$ }

is_point: a unary predicate name for the set **PT**.

is_polyline: a unary predicate name for the set **PLine**.

join_at: a ternary predicate name that signifies the fact that two polylines join ends at a point. That is,

$\{(l_1, l_2, pt) \mid \mathsf{is\_polyline}(l_1), \mathsf{is\_polyline}(l_2), \mathsf{is\_point}(\mathrm{pt}), l_1$ and l_2 are not identical,

and pt is the only common endpoint of l_1 and l_2.}

endpoints_of is a ternary predicate name indicating the relation of a polyline and its end points, which is defined as

$\{(l_1, p_1, p_2) \mid p_1$ and p_2 are the two end points of polyline l_1.$\}$

intersect is a binary predicate defined as

$\{(l_1, l_2) \mid l_1$ and l_2 intersect at a common point other than their end points.$\}$

It is easy to verify that domains $\mathcal{POINT}$ and $\mathcal{GEOM}$ are admissible.

4.4 Generalized $\mathcal{ALC}(\mathcal{D})$

The conceptual language $\mathcal{ALC}(\mathcal{D})$ discussed in previous sections is sufficient for modeling applications of different natures. Conceptual models offer more expressive facilities than conventional modeling tools. However, the roles in $\mathcal{ALC}(\mathcal{D})$ are only interpreted as binary relations in models. On many occasions, it is desirable to overcome this restriction. In this section, the author proposes to to extend $\mathcal{ALC}(\mathcal{D})$ with n-ary roles. The extended conceptual language, called $\mathcal{GALC}(\ell, \mathcal{D})$, has more expressive power than $\mathcal{ALC}(\mathcal{D})$ does. Such an extension is necessary because a map phenomenon usually involves more than two types of map objects. N-ary roles are needed to accurately express how multiple map objects are put together to represent a higher level concept.

Definition 4.1 *$\mathcal{GALC}(\ell, \mathcal{D})$ can be obtained by extending the definition of $\mathcal{ALC}(\mathcal{D})$ (i.e., **Definition** A.3) by allowing the forming of concept terms using the following expressions:*

1. $\exists R.C_1 \otimes C_2 \otimes \cdots \otimes C_m$ *((m+1)-ary exist-in restriction), and*
2. $\forall R.C_1 \otimes C_2 \otimes \cdots \otimes C_n$ *((m+1)-ary value restriction).*

*where ℓ is called **maximum role size**, $\ell \in \mathbb{N}$ and $\ell \geqslant 2$, which defines the upper bound of the role size, and we have $m, n \leqslant \ell - 1$.*

Respectively, the syntax and semantics of $\mathcal{GALC}(\ell, \mathcal{D})$ can be defined as follows:

Definition 4.2 *(interpretation and models of $\mathcal{GALC}(\ell, \mathcal{D})$)*

***Definition** A.4 in Appendix A is extended to include the following:*

1. $(\exists R.C_1 \otimes C_2 \otimes \cdots \otimes C_m)^{\mathcal{I}} =$

   $\{x \in dom(\mathcal{I}) \mid$ *there exist* $y_1, y_2, \cdots, y_m$ *such that* $(x, y_1, y_2, \cdots, y_m) \in R^{\mathcal{I}}$ *and* $y_1 \in C_1, y_2 \in C_2, \cdots, y_m \in C_m\}$

2. $(\forall R.C_1 \otimes C_2 \otimes \cdots \otimes C_n)^{\mathcal{I}} =$

   $\{x \in dom(\mathcal{I}) \mid$ *for all* $y_1, y_2, \cdots, y_n$ *such that* $(x, y_1, y_2, \cdots, y_n) \in R^{\mathcal{I}}$, *we have* $y_1 \in C_1, y_2 \in C_2, \cdots, y_n \in C_n\}$

To describe individual concept and role instances, the assertional language for $\mathcal{GALC}(\ell, \mathcal{D})$ is defined by extending that of $\mathcal{ALC}(\mathcal{D})$. In addition to the axioms

allowed in an A-box of $\mathcal{ALC}(\mathcal{D})$, an A-box with respect to $\mathcal{GALC}(\ell, \mathcal{D})$ will also allow assertional axioms of the form $(a_1, \cdots, a_n) : \mathrm{R}$, where R is an n-ary role. An interpretation $\mathcal{I}$ of $\mathcal{GALC}(\ell, \mathcal{D})$ satisfies $(a_1, \cdots, a_n) : \mathrm{R}$ *iff* $(a_1^{\mathcal{I}}, \cdots, a_n^{\mathcal{I}}) \in R^{\mathcal{I}}$.

In general, conceptual modeling of domain knowledge is a subjective process. There are often many possible ways of modeling the map objects and their relations. Different knowledge engineers may use different sets of concepts, roles and features to describe the same map phenomena. The choice of modeling approaches can have a large impact on the applicability and adaptability of a design. One important modeling decision is the arity of roles. For a given scenario, some use binary roles, while others may prefer to use higher arity ones. In many cases, the choice between binary and higher arity only reflects different viewpoints of knowledge representation. However, sometimes higher arity relations supply more direct and natural descriptions. Consider, for example, the scenario shown in Figure 4.3, where highway intersections are explicitly represented using small circles. It is natural to identify three concepts (hw_route, route_section, and hw_intersection) and one ternary role ($\underline{\text{RCHAINED\_WITH}_R}$). The construction relationship among these three concepts can be defined using the following terminological axiom:

$$\mathsf{hw\_route} = \exists\ \underline{\text{RCHAINED\_WITH}_R}\ .\ \mathsf{route\_section} \otimes \mathsf{hw\_intersection}$$

It is apparent that the ternary role $\underline{\text{RCHAINED\_WITH}_R}$ is not simply the addition of the three binary roles between any two of these three concepts. It is possible to

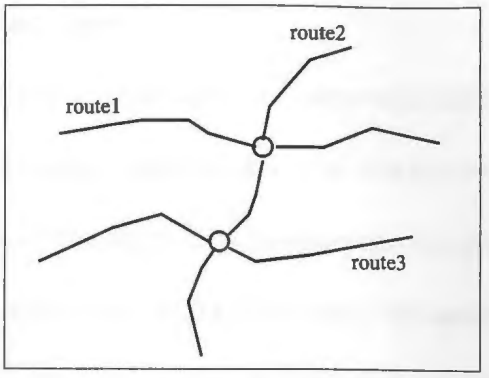


Figure 4.3: Three intersecting highway routes.

substitute a higher arity role with a number of binary roles. However, this is accomplished by looking at the scenario from a different standpoint. In particular, a new concept routesec_with_intersection and a new role $\underline{\text{RI\_CHAINED\_WITH}_R}$ are identified, and the "highway route" concept is defined as

$$\text{hw\_route} = \exists\ \underline{\text{RI\_CHAINED\_WITH}_R}\ .\ \text{routesec\_with\_intersection}$$

This means that the two types of map objects described earlier by the two concepts route_section and hw_intersection are in fact defined as the building blocks of the new concept named routesec_with_intersection. From this perspective, a highway route is made of a series of entities, and each of such entities is a highway section with intersection symbols at one or both of its end points. From this example, it can be seen that the new concepts and binary roles may not reflect our natural way of thinking. In many cases, the subtle semantic differences between the binary and n-ary roles may not be recognized easily. Allowing engineers to use higher arity roles can enable them to identify concepts and roles in an intuitive and flexible way, and thus

give them more expressing power.

In $\mathcal{ALC}(\mathcal{D})$, the reasoning algorithm has been established to decide the consistency of an A-box, which means the consistency of the knowledge representation can be verified at design time. This calls for a mechanism to transform a knowledge representation in $\mathcal{GALC}(\ell, \mathcal{D})$ to that in $\mathcal{ALC}(\mathcal{D})$, since the analysis of an $\mathcal{ALC}(\mathcal{D})$ representation is easier and readily available. In the following section, we will show that for any $\mathcal{GALC}(\ell, \mathcal{D})$, there exists an $\mathcal{ALC}(\mathcal{D})$ such that all models of the $\mathcal{GALC}(\ell, \mathcal{D})$ are also models of the corresponding $\mathcal{ALC}(\mathcal{D})$. The transformation rules described in the next section can be applied to obtain an $\mathcal{ALC}(\mathcal{D})$ from a $\mathcal{GALC}(\ell, \mathcal{D})$.

Before the formal definition of the transformation rules is presented, we shall give an example of mapping a $\mathcal{GALC}(\ell, \mathcal{D})$ representation to an $\mathcal{ALC}(\mathcal{D})$ one. Consider the ternary role **supply** among these three concepts: manufacturer, product and customer. The assertional language of $\mathcal{GALC}(\ell, \mathcal{D})$ can be exploited to express facts regarding the manufacturers, products, customers, and their relationships. For example, the fact that "manufacturer m_1 supplies product p_1 to customer p_1" can be represented using the following axioms:

$\{ (m_1, p_1, c_1) : \textbf{supply}, m_1 : \mathsf{manufacturer}, p_1 : \mathsf{product}, c_1 : \mathsf{customer} \}$

Axiom $(m_1, p_1, c_1) :$ **supply** can be viewed as a relation instance of the ternary relationship expressed by **supply**. It is obvious that **supply** naturally is not a binary role. In order to take advantage of the fruitful research on $\mathcal{ALC}(\mathcal{D})$, it is desirable to study the relationship between $\mathcal{ALC}(\mathcal{D})$ and $\mathcal{GALC}(\ell, \mathcal{D})$. An intuitive treatment

may be to replace the role **supply** with two binary roles; one role (**provide**) describes the relationship between manufacturer and product, the other one (**used_by**) describes the relationship between product and customer. In other words, an instance of role **supply**, (m_1, p_1, c_1) : **supply**, is substituted by (m_1, p_1) : **provide** and (p_1, c_1) : **used_by**, which mean facts "m_1 provides p_1" and "p_1 is used by c_1", respectively. It is easy to tell that semantically **provide** and **used_by** in combination are not equivalent to **supply**. However, the two binary roles partially preserve the semantics of **supply**, and thus can be considered a weak form of substitution. This type of substitution is regarded as a process of semantic generalization due to the fact that some specific information is missing in the process. One the one hand, the role instance (m_1, p_1, c_1) : **supply** implies (m_1, p_1) : **provide** and (p_1, c_1) : **used_by**. On the other hand, the two facts (m_1, p_1) : **provide** and (p_1, c_1) : **used_by** do not imply that the fact (m_1, p_1, c_1) : **supply** is true, instead they may be the result of (m_1, p_1, c_2) : **supply** and (m_2, p_1, c_1) : **supply**. In other words, the product p_1 used by c_1 is not necessarily provided by m_1.

In general, an n-ary role can be replaced with $n-1$ binary roles in the same way we treat role **supply**. Although this means some information loss, it is beneficial in terms of analysis of knowledge representation. Next we will introduce the concept of S-equivalency, which means in order to check the validity of the model design, we do not need to perform consistency test directly on a $\mathcal{GALC}(\ell, \mathcal{D})$ representation. A consistency test based on the corresponding $\mathcal{ALC}(\mathcal{D})$ representation is sufficient.

Mapping a $\mathcal{GALC}(\ell, \mathcal{D})$ language into an $\mathcal{ALC}(\mathcal{D})$ language

Let $\mathcal{GL} =< \mathbf{C}, \mathbf{R}, \mathbf{F}, \mathcal{D}, \mathcal{T} >$ be a $\mathcal{GALC}(\ell, \mathcal{D})$ language, where $\mathbf{C}$ and $\mathbf{F}$ are concepts and features in $\mathcal{ALC}(\mathcal{D})$, $\mathcal{D}$ is the concrete domain, $\mathbf{R}$ is the set of roles, and $\mathcal{T}$ is the set of axioms.

$\mathbf{R}$ is written as follows:

$\mathbf{R} = \{R_i | i = 1, 2, \cdots, n\}$, where R_i is a $\mu(i)$-ary role. Function μ is defined as a mapping:

μ: $\mathbb{I} \rightarrow \mathbb{N}$, where $\mathbb{I}$ is the set of integers and $\mathbb{N}$ is the set of natural numbers.

Since $\mathcal{GL}$ has an upper bound ℓ, we have $\mu(i) \leqslant \ell$.

We can obtain an $\mathcal{ALC}(\mathcal{D})$ language $\mathcal{L}$, expressed as $<\mathbf{C}^\dagger, \mathbf{F}^\dagger, \mathcal{D}^\dagger, \mathbf{R}^\dagger, \mathcal{T}^\dagger>$, from $\mathcal{GL}$, where

$$\mathbf{C}^\dagger = \mathbf{C}, \mathbf{R}^\dagger = \Psi(\mathbf{R}), \mathbf{F}^\dagger = \mathbf{F}, \mathcal{D}^\dagger = \mathcal{D}, \mathcal{T}^\dagger = \Theta(\mathcal{T})$$

$\mathcal{L}$ directly uses the concepts, features, and concrete domain of $\mathcal{GL}$. Functions Ψ and Θ map $\mathcal{GL}$'s role set and axiom set to those of $\mathcal{L}$'s respectively. Given $\mathbf{R} = \{R_1, R_2, \cdots, R_n\}$, n $\in \mathbb{N}$.

$$\mathbf{R}^\dagger = \Psi(\mathbf{R}) = \bigcup_{i=1}^{n} \psi(R_i)$$

Let $\mathfrak{R}$ be the set of all possible roles. The mapping ψ: $\mathfrak{R} \rightarrow 2^{\mathfrak{R}}$ in turn is defined as follows:

$$\psi(R_i) = \begin{cases} \{R_i\} & \text{if } R_i \text{ is a binary role,} \\ \{R_{i,1}, R_{i,2}, \cdots, R_{i,m}\} & \text{if R is an (m+1)-ary role, and m} \geqslant 2. \end{cases}$$

where $R_{i,1}$, $\cdots$, $R_{i,m}$ are binary role names.

Given $\mathcal{T} = \{T_1, T_2, \cdots, T_k\}$, k $\in \mathbb{N}$.

$$\Theta(\mathcal{T}) = \{\theta(T_i) | \text{i=1, 2, } \cdots, k\}$$

Assume that ψ(R) = $\{R'_1, R'_2, \cdots, R'_m\}$, and let $\mathfrak{T}$ be the set of all possible axioms. Then the mapping θ: $\mathfrak{T} \rightarrow \mathfrak{T}$ performs the following substitutions on an axiom, say T_i:

For each occurrence of $\exists$R.$C_1 \otimes \cdots \otimes C_m$ and $\forall$R.$C_1 \otimes \cdots \otimes C_m$ in T_i, where m $\geqslant$ 2,

- replace $\exists$R .$C_1 \otimes \cdots \otimes C_m$ with $\exists R'_1$.$(C_1 \sqcap \exists\ R'_2$.$(C_2 \sqcap \cdots (C_{m-1} \sqcap \exists\ R'_m$. $C_m) \cdots))$,
- replace $\forall$R .$C_1 \otimes \cdots \otimes C_m$ with $\forall R'_1$.$(C_1 \sqcap \forall\ R'_2$.$(C_2 \sqcap \cdots (C_{m-1} \sqcap \forall\ R'_m$. $C_m) \cdots))$.

It has been seen syntactically how a $\mathcal{GALC}(\mathcal{D})$ is mapped to an $\mathcal{ALC}(\mathcal{D})$. Now we are going to consider how a $\mathcal{GALC}(\mathcal{D})$ interpretation is mapped to an $\mathcal{ALC}(\mathcal{D})$

one. Let $\mathcal{I}$ be an interpretation of $\mathcal{GL}$, then $\mathcal{I}'$ is considered as an interpretation of $\mathcal{L}$ with respect to mapping (Φ, Θ), as long as the following conditions are satisfied:

1. For those concepts, roles and functions that are taken directly from $\mathcal{GL}$, their interpretations are the same as those in $\mathcal{L}$.

2. For those roles that are derived from the m-ary roles (m $\geqslant$ 3) in $\mathcal{GL}$, their interpretations are obtained as follows:

   Given an m-ary role (m $\geqslant$ 3) R. Then $\psi(\mathrm{R}) = \{R'_1, R'_2, \cdots, R'_{m-1}\}$. Let $R^{\mathcal{I}} = \{(c_1, c_2, \cdots, c_m) \mid c_1 \in C_1, c_2 \in C_2, \cdots c_m \in C_m\}$.

   Then we have

   $R_k^{\mathcal{I}'} = \{(c'_k, c'_{k+1}) |$ there exist $c'_1, \cdots, c'_{k-1}, c'_{k+2}, \cdots, c'_m$ so that $(c'_1, c'_2, \cdots, c'_m) \in \mathrm{R}\}$, where k = 1, 2, $\cdots$, m-1.

   It can be seen that the outcome of this operation is to substitute each n-ary relation in $\mathcal{GL}$ with a set of binary relations in $\mathcal{L}$, which is called **relation upcast**.

A relation upcast can be viewed as a process of generalization, as shown in Figure 4.4.

Therefore, we can define the (Φ, Θ) **morphism** as follows:

Let the (Φ, Θ) morphism of A-box A be denoted as A^M.

For axiom **x** that does not have any occurrence of a higher arity ($\geqslant$ 3) role, **x** $\in$ A *iff* **x** $\in A^M$.

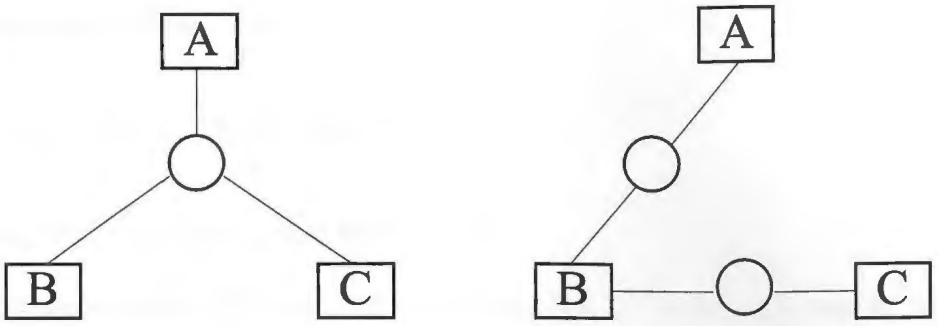


Figure 4.4: Upcast viewed as relation generalization.

For any occurrence of a higher arity role, it is substituted with its corresponding concept term defined by mapping θ. This is to say, each axiom of the form

$$\text{a} : \exists \text{R}.C_1 \otimes \cdots \otimes C_m$$

corresponds to the axiom

$$\text{a} : \exists R'_1.(C_1 \sqcap \exists R'_2(C_2 \sqcap \cdots(\exists R'_m.C_m))).$$

Each axiom of the form $(c_1, \cdots, c_m)$: R, m $\geqslant$ 3, is replaced by $m-1$ axioms $(c_1, c_2) : R'_1, \cdots, (c_{m-1}, c_m) : R'_{m-1}$, where $\Phi(\text{R}) = \{R'_1, \cdots, R'_{m-1}\}$.

It is important to note that $\mathcal{L}$ is not equivalent to $\mathcal{GL}$. As an example, let us consider mapping a $\mathcal{GALC}(\mathcal{D})$ language

$\mathcal{GM}$ = <{A, B, C}, {R}, **F**, **D**, T>

to an $\mathcal{ALC}(\mathcal{D})$ language

$$\mathcal{M} = <\{\text{A, B, C}\}; \{R_1, R_2\}, \mathbf{F}, \mathbf{D}, T'>$$

using $\Phi(R) = \{R_1, R_2\}$ and $\Theta(T) = T'$.

Suppose we have $A^{\mathcal{I}} = \{a_1, a_2\}$, $B^{\mathcal{I}} = \{b_1, b_2\}$, $C^{\mathcal{I}} = \{c_1, c_2\}$, and $R^{\mathcal{I}} = \{(a_1, b_1, c_1), (a_2, b_2, c_2), (a_1, b_2, c_1), (a_2, b_1, c_2)\}$. Function Θ will map the interpretation of R to $R_1^{\mathcal{I}} = \{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)\}$ and $R_2^{\mathcal{I}} = (b_1, c_1), (b_1, c_2), (b_2, c_1), (b_2, c_2)\}$.

However, Θ can also map a different interpretation of R to the same R_1 and R_2. Given $R^{\mathcal{I}'} = \{ (a_1, b_1, c_2), (a_2, b_2, c_1), (a_1, b_2, c_2), (a_2, b_1, c_1), (a_1, b_1, c_2)\}$. It can be seen that $\mathcal{I}$ and $\mathcal{I}'$ are obvious different interpretations of R. The function Θ can still map them to the same set of binary roles. That is, $\Theta(R^{\mathcal{I}}) = \Theta(R^{\mathcal{I}'})$.

In fact, the model of $\mathcal{L}$ can be looked at as a generalization of $\mathcal{GL}$. One feature of the mappings (Φ, Θ) is that it preserves the property of verifiability. Let $\mathcal{A}_0$ be an A-box for $\mathcal{GALC}(\ell, \mathcal{D})$, $\mathcal{A}_1$ be an A-box obtained through mappings (Φ, Θ). If $\mathcal{A}_1$ is not consistent, $\mathcal{A}_0$ is also inconsistent, and vice versa.

An A-box $\mathcal{A}$ is said to be inconsistent *iff* it contains at least one of the following clash rules:

1. $\mathcal{A}$ contains axioms $(a, x) : f$ and $(a, b) : f$ for a feature name f. This is an obvious contradiction because an abstract object (b) and a concrete object (x) cannot be identical.

2. $\mathcal{A}$ contains axioms $(a, x) : f$ and $a : \forall f.C$. This is an obvious contradiction because the concrete object (x) cannot be an element of a concept.

3. $\mathcal{A}$ contains axioms $a : A$ and $a : \neg A$ for a concept name A.

4. $\mathcal{A}$ contains axioms $(x_1^{(1)}, \cdots, x_{n_1}^{(1)}) : P_1, \cdots, (x_1^{(k)}, \cdots, x_{n_k}^{(k)}) : P_k$ and the corresponding conjunction $\wedge_{i=1}^{k} P_i(\underline{x}^{(i)})$ is not satisfiable in $\mathcal{D}$. This is a contradiction because $\mathcal{D}$ is admissible.

We only need to prove that if there exists one clash in $\mathcal{A}_0$, the same axioms involved in this clash can also be found in $\mathcal{A}_1$.

Definition 4.3 *(Transformation Rules)*

Let $\mathcal{M}$ be a finite set of A-boxes, and let $\mathcal{A}$ be an element of $\mathcal{M}$.

1. *The conjunction rule. Assume that $a : (C \sqcap D)$ is in $\mathcal{A}$ and $a : C$ or $a : D$ is not in $\mathcal{A}$. The A-box $\mathcal{A}'$ is obtained from $\mathcal{A}$ by adding the two axioms $a : C$, $a : D$ to $\mathcal{A}$.*

2. *The disjunction rule. Assume that $a : (C \sqcup D)$ is in $\mathcal{A}$ and neither $a : C$ nor $a : D$ is in $\mathcal{A}$. The A-box $\mathcal{A}'$ is obtained from $\mathcal{A}$ by adding $a : C$ to $\mathcal{A}$, and A-box $\mathcal{A}''$ is obtained from $\mathcal{A}$ by adding axiom $a : D$ to $\mathcal{A}$.*

3. *The exists-in restriction rule. Assume that $a : \exists R.C_1 \otimes \cdots \otimes C_m$ is in $\mathcal{A}$ and that there are no object names $c_1, c_2, \cdots, c_m$ in OA such that the axioms $(a, c_1, c_2, \cdots, c_m) : R$ and $c_1 : C_1, \cdots, c_m : C_m$ are in $\mathcal{A}$. Let $b_1, \cdots, b_m \in OA$ be*

"new" abstract names (i.e, names not occurring in $\mathcal{A}$). Then we add the m+1 new axioms to $\mathcal{A}$.

4. *The value restriction rule. Assume that* $a : \forall R.C_1 \otimes \cdots \otimes C_m$ *and* $(a, b_1, \cdots, b_m) : R$ *are in* $\mathcal{A}$ *and that* $b_1 : C_1, \cdots, b_m : C_m$ *are not in* $\mathcal{A}$. *The A-box* $\mathcal{A}'$ *is obtained from* $\mathcal{A}$ *by adding the axioms* $b_1 : C_1, \cdots, b_m : C_m$.

5. *The predicate restriction rule. Assume that* $a : P(u_1, \cdots, u_n)$ *is in* $\mathcal{A}$ *and that the following does not hold:*

   For the feature chains $u_i = f_{i1} \cdots f_{in_i}$, $i = 1, 2, \cdots n$, *there are object names* $b_{i1}, \cdots, b_{in_i-1} \in OA$ *and* $x_i \in OC$ *such that the A-box* $\mathcal{A}$ *contains axioms* $(a, b_{i1}) : f_{i1}$, $(b_{i1}, b_{i2}) : f_{i2}, \cdots, (b_{in_i-1}, x_i) : i_{n_i}$, *and* $(x_1, \cdots, x_n) : P$.

   For each of the feature chains u_i, *we choose new object names* $b_{i1}, \cdots, b_{in_i-1} \in OA$ *and* $x_i \in OC$, *and augment the A-box by the axioms* $(a, b_{i1}) : f_{i1}$, $(b_{i1}, b_{i2}) : f_{i2}, \cdots, (b_{in_i-1}, x_i) : f_{in_i}$. *New forks may be created, which means two different names are assigned to the same object. They are deleted by using a unique name for each object. Finally we add* $(x_1, \cdots, x_n) : P$ *to obtain the A-box* $\mathcal{A}'$.

Algorithm 4.1 *(Consistency Test)*

The following procedure takes an A-box $\mathcal{A}_0$ *as an argument and checks whether it is consistent or not.*

define procedure check_consistency($\mathcal{A}_0$)

$\mathcal{A}_1$ = *eliminate_forks*($\mathcal{A}_0$)

$r = 1$

$M[1] = \{\mathcal{A}_1\}$

while *'a transformation rule is applicable to* $M[r]$*'* **do**

$r = r + 1$

$M[r]$ = *apply-a-transformation-rule*($M[r-1]$)

done

if *'there is an* $\mathcal{A} \in M[r]$ *that does not contain a clash'*

then return *consistent*

else return *inconsistent*

Proposition 4.1 *Let $\mathcal{A}_0$ be an A-box of $\mathcal{GALC}(\ell, \mathcal{D})$, and $\mathcal{A}_1$ be an A-box of $\mathcal{ALC}(\mathcal{D})$, which is obtained from $\mathcal{M}_0$ through the morphism (Φ, Θ). Then $\mathcal{A}_0$ is inconsistent iff $\mathcal{A}_1$ is inconsistent.*

Proof. Suppose the consistency test applied on $\mathcal{A}_0$ suggests it has a clash. We just need to show that $\mathcal{A}_1$ also contains a clash. Conversely, we need to show that if $\mathcal{A}_1$ contains a clash, $\mathcal{A}_0$ does as well.

The proposition is the consequence of Lemma 4.1.

■

Before we move on to prove Lemma 4.1, let us introduce the concept *synonymous axiom* and the concept *S-equivalence between A-boxes.*

Definition 4.4 *(Synonymous Assertional Axioms)*

We say two axioms $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are* ***Synonyms*** *iff* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are identical or* $\boldsymbol{y}$ *is the* (Φ, Θ) *morphism form of* $\boldsymbol{x}$*. Two sets of axioms are* ***synonymous*** *iff for each axiom of one set we can find a corresponding synonymous axiom in the other set.*

Definition 4.5 *(S-equivalence)*

Let A_1 *and* A_2 *be two A-boxes.* A_1 *is said to be S-equivalent to* A_2 *iff for each axiom of the form* $(a, b) : f$*,* $(a, y) : f$*,* $a : C$*, or* $(y_1, \cdots, y_n) : P$ *in* A_1*, there exists a synonymous counterpart in* A_2*.*

Lemma 4.1 $\mathcal{A}_0$ *violates clash rules 1, 2, 3, and 4 iff* $\mathcal{A}_1$ *violates clash rules 1, 2, 3, and 4, respectively.*

Proof. Let $M_{\mathcal{A}_0}$ be the complete set of A-boxes obtained from $\mathcal{A}_0$ by applying the consistency test, and let $M_{\mathcal{A}_1}$ be that obtained from $\mathcal{A}_1$.

Let B_0 be a $\mathcal{GALC}$ A-box, B_1 be an $\mathcal{ALC}$ one. Let B_0 and B_1 be S-equivalent.

By applying a transformation rule on B_0 and B_1, new A-boxes B_0' (and B_0'') and B_1' (and B_1'') are obtained. We will see that B_0' and B_1' are also S-equivalent. To prove this, we just need to show that for each new axiom added to B_0, the transformation rule also requires its synonymous form be added to B_1, since B_0' and B_1' are obtained by augmentation of B_0 and B_1 respectively.

If an axiom $(a, e) : f$ (a, e are abstract objects) is newly added to B_0', it can be determined that it results from either the fact that $a : \exists f.E$ is in B_0 and there is no

$e : E$ so that $(a, e) : f$, or the fact that $a : P$ is in B_0, where P is such a predicate that $a : P$ implies $(a, e) : f$. Such a fact also applies to B_1, since B_0 and B_1 are S-equivalent. Consequently, $(a, e) : f$ is also an axiom added to B_1'.

If $(a, y) : f$ (y is a concrete object) in B_0' is newly added, it must be the result of the following case:

> $a : \mathrm{P}(u_1, \cdots, f, \cdots, u_n)$ is in B_0 and there exists no axiom $(a, y) : f$ in B_0.

Since B_1 is S-equivalent to B_0, the axiom $a : \mathrm{P}(u_1, \cdots, f, \cdots, u_n)$ also appears in B_1, and $(a, y) : f$ is not in B_1. As a result, $(a, y) : f$ appears in B_1'.

If an axiom $a : C$ is newly added to B_0', according to the transformation rules, this may be the result of the following cases:

1. An axiom $a : (C \sqcap D)$ is in B_0 and $a : C$ is not in B_0. Thus we have $a : \Phi(C \sqcap D)$ (i.e., $a : \Phi(C) \sqcap \Phi(D)$) in B_1, and a : $\Phi(C)$ is not in B_1. According to the transformation rules, $a : \Phi(C)$ will be a newly added axiom in B_1', which is synonymous to $a : C$.

2. An axiom $a : (C \sqcup D)$ is in B_0 and $a : C$ is not. This requires that the axiom $a : \Phi(C) \sqcup \Phi(D)$ be in B_1 and $a : \Phi(D)$ not be in B_1. Therefore one of the two augmented A-boxes, say, B_1', is S-equivalent to B_0'.

3. $a : \exists R.C_1 \otimes \cdots \otimes C_m$ is in B_0, and one of the concept terms, say, C_k, is actually C. The synonymous counterpart of this axiom in B_1 will be $a : \exists R_1'.(\Phi(C_1) \sqcap$

$\exists\ R'_2.(\Phi(C_2) \sqcap \cdots \exists\ R'_m.\Phi(C_m)))$. If $b_k : C$ is not in B_0 and is added as a new axiom, its synonymous form, $b_k : \Phi(C)$, is also not in B_1. To obtain B'_1, the transformation rules are applied a number of times and $b_k : \Phi(C)$ is added.

4. $a : \forall\ R.C_1 \otimes \cdots \exists f.C \cdots \otimes C_m$ and $(a, b_1, \cdots, b_m) : R$ are in B_0, and $b_k : C$ is not in B_0. According to the definition of S-equivalence, we will have axioms $a : \forall\ R'_1.(\Phi(C_1) \sqcap \forall\ R'_2.(\Phi(C_2) \sqcap \cdots \forall\ R'_m.\ \Phi(C_m) \cdots))$ and $(a, b_1) : R'_1, \cdots,$ $(b_{m-1}, b_m) : R'_m$ in B_1. But $b_k : \Phi(C)$ is not in B_1. It is obvious $b_k : \Phi(C)$ also has to be added to B_1 to obtain B'_1.

If the transformation rules add new axioms to B_0, their synonymous items are also added to B_1. Thus the resulting B'_0 and B'_1 are S-equivalent.

Now consider the scenarios when clash rules are violated. From the above discussion, we have known that the consistency test algorithm maintains the S-equivalence between $M_{\mathcal{A}_0}$ and $M_{\mathcal{A}_1}$ in the process of A-box augmentation. Consequently we can draw the conclusion that $M_{\mathcal{A}_0}$ violates a clash rule *iff* $M_{\mathcal{A}_1}$ violates the same clash rule as well.

■

Chapter 5

Representing Knowledge for Map Understanding

In the previous chapter a Description Logics language was introduced to capture knowledge of the world. Such a representation language is very useful in the design of knowledge based applications. Various applications have been developed based on Description Logics. The close interaction between theory and practice is the key to exploiting the expressive power of Description Logics. Description Logics can also be utilized in the knowledge representation of map understanding systems. The success of the system design relies on a precise characterization of the knowledge base.

This chapter concerns how to model map knowledge based on Description Logics. In particular, examples are presented to demonstrate the modeling process.

5.1 Modeling with Description Logics

Up to this point, we have been concentrating on the theoretical aspects of the conceptual language $\mathcal{ALC}(\mathcal{D})$ and its assertional language. The two closely related formal languages provide us a special viewpoint to treat object instances and their relations in an application domain. Terminological axioms (T-box) and assertional axioms (A-box) are used to describe knowledge about the real world.

Next, an example is used to summarize the approach to apply the Description Logics theory to represent spatial data such as topographic maps. The philosophy of the modeling of map knowledge and subsequent applications is illustrated in Figure 5.1.

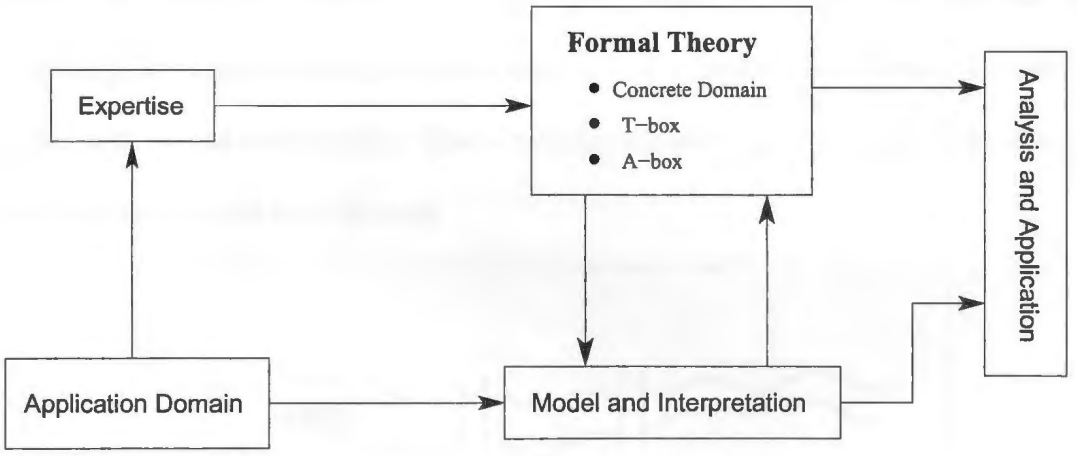


Figure 5.1: The DL conceptual modeling philosophy.

Expertise is obtained from application domains through inquiry, research and development activities. It is in the form of intuitive knowledge, which is acquired from the application domain being researched. Through recognition and analysis, representations based on a formal theory are established. These representations (concrete

domain, T-box, and A-box) capture the essential properties and relations of the elements in the application. A model is regarded as an interpretation of the formal conceptual and assertional languages. It can also be seen as an abstraction of the application domain, since the model does not try to capture all aspects of the real world (which is impossible), but those specific aspects that are most representative. The model also affects the evolution of the formal theory (verification, modification, and extension).

The conceptual language $\mathcal{GALC}(\ell, \mathcal{D})$ can be used to represent the following map knowledge: (1) map object types and object instances; (2) relationships among map objects and relation instances; and (3) abstract concept hierarchy. For the sake of simplicity, we choose a simple map scene —— a complex river section shown in Figure 5.2 —— as our example. The terminological and assertional knowledge about the river section will be presented.

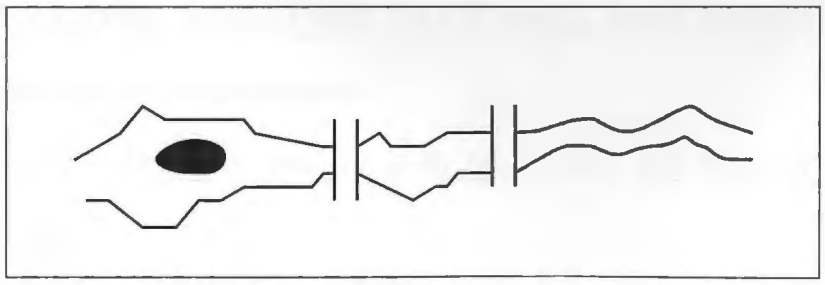


Figure 5.2: A complex river section with an island and two bridges.

Any map scene is an integrated entity which is composed of a number of building parts. There exist innately complex interactions among those parts. An initial step toward the construction of a map scene model from a Description Logics viewpoint is

to identify the concrete domains. For the example in Figure 5.2, the concrete domain $\mathcal{GEOM}$ defined in Section 4.3.2 of Chapter 4 is used.

A T-box $\mathcal{T}_{\mathcal{ACC}(\mathcal{GEOM})}$ is then defined, which is the result of considering how a complex river section is constructed using other defined concepts. A T-box contains two different kinds of concepts: *derived* and *primitive*. Derived concepts occur on the left hand side of a terminological axiom, while all other concepts are primitive. In this example, the primitive concepts are defined as follows

{bridge, island, elementary_river_portion}

The derived concepts are

{ complex_river_section, part_of_complex_river_section, river_section_with_bridge, complex_river_segment, river_portion_with_island }

The following roles are defined:

{CHAINED_WITH$_R$, HAS_B_PART$_R$, HAS_R_PART$_R$, BUILD_WITH$_R$}

The following features are defined:

{B_PART_OF$_F$, R_PART_OF_ONE$_F$, R_PART_OF_OTHER$_F$, ERP_PART_OF$_F$, IS_PART_OF$_F$}

The predicates defined are

{*PRE_form_riversbridge_connection*, *PRE_contain_in_middle*}

Mathematically, the following terminological axioms in $\mathcal{T}_{\mathcal{ACC}(\mathcal{GEOM})}$ are given.

complex_river_section = $\forall$CHAINED_WITH$_R$.part_of_complex_river_section

part_of_complex_river_section = river_section_with_bridge

river_section_with_bridge = $\exists$ HAS_B_PART$_R$.bridge $\sqcap$ $\exists$

HAS_R_PART$_R$.complex_river_segment $\sqcap$

PRE_form_riversbridge_connection(B_PART_OF$_F$, R_PART_OF_ONE$_F$,

R_PART_OF_OTHER$_F$)

complex_river_segment = river_portion_with_island $\sqcup$ elementary_river_portion

river_portion_with_island = $\exists$ BUILD_WITH$_R$ .elementary_river_portion $\sqcap$ $\exists$

BUILD_WITH$_R$ .island $\sqcap$ *PRE_contain_in_middle*(ERP_PART_OF$_F$,

IS_PART_OF$_F$)

The T-box allows us to define a new concept based on other previously defined concepts. The operations used to build the new concept provide both sufficient and necessary conditions for classifying an individual as a member of the new concept. The T-box's main task is to show how groups of instances with common properties are related with one another. However, interrelated individuals are encoded in the semantics of roles; that is, a subset of the product dom($\mathcal{D}$) $\times$ dom($\mathcal{D}$). In the real world, when it comes to specifying the semantics of roles, the following two approaches are usually used:

- Enumerate all the ordered pairs (relation instances).

- Establish a set of constraints or properties that determines whether a given relation instance belongs to the role.

Enumerating through the whole set of relation instances is not practical since it may be a huge set depending on the types of maps. For a given map, the number of relation instances is finite. However, the semantic constraints have to be defined on the domain of all the map instances that may participate in the relation, which may be a huge collection of instances. The better way to apprehend the meaning of a role is to specify what kind of scenarios may suggest the happening of the role. For instance, in map understanding applications, the role "inside" can be represented as a truth-value function (or algorithm). It is not necessary to list all the individuals that are inside another instance. Sometimes, special constraints are put forward to be incorporated into the syntax of the roles. For example, the role name $R_{\leqslant 2}$ indicates that given a, $\sharp\{b \mid (a, b) \in R^{\mathcal{I}}\} \leqslant 2$.[1]

The A-box contains extensional knowledge of individuals in both abstract and concrete domains. It includes *membership assertions* and *role assertions*. To distinguish one individual from another, an "extrinsic" identification scheme that assigns a number to each individual is used. Suppose there is a need to express the following facts: (1) An instance of the complex river section, **crs_1**, is composed of two attached parts: **pcs_1** and **pcs_2**; (2) **pcs_1** and **pcs_2** are also instances of complex sections with bridges on them; and (3) **rpwi_1** is a portion of river with an island

[1]The operation $\sharp$ returns the number of elements in a set.

inside. Figure 5.3 is a graphical illustration of the facts. With assertional axioms, these facts can be expressed as follows:

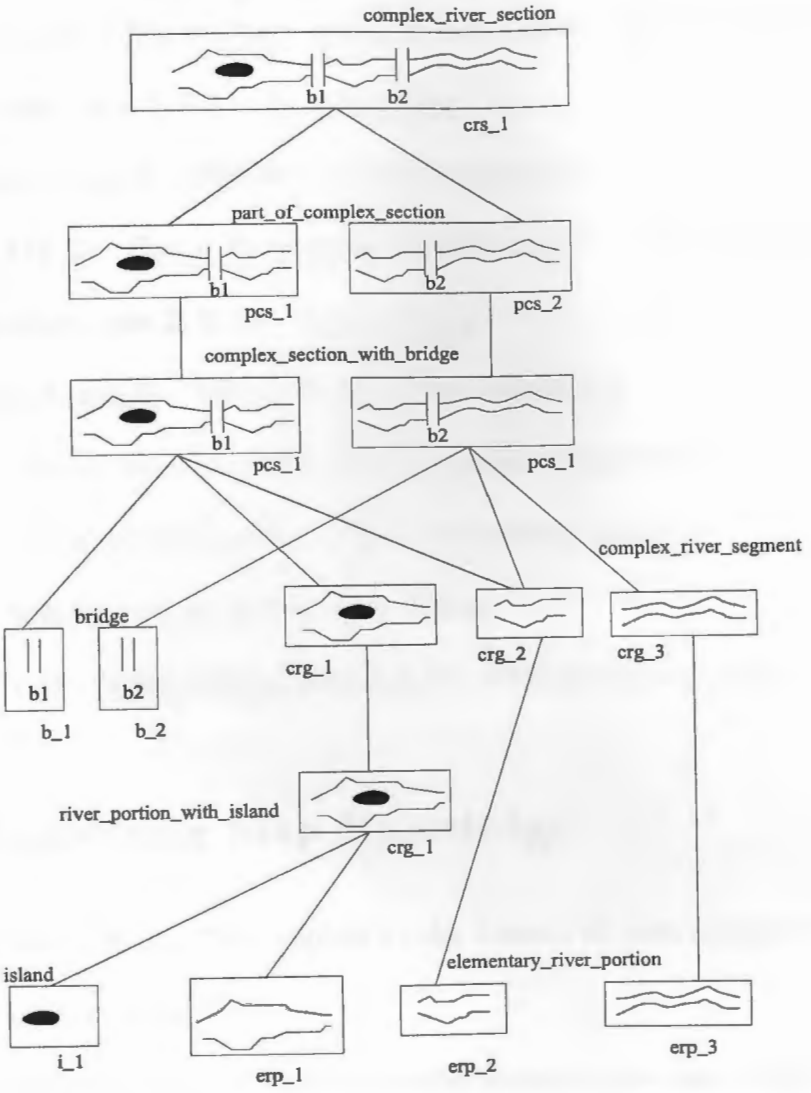


Figure 5.3: Concepts involved in a complex river section.

{ **crs_1** : complex_river_section, **pcs_1** : part_of_complex_section,

pcs_2 : part_of_complex_section, (**crs_1**, **pcs_1**) : $\underline{\text{CHAINED\_WITH}_R}$,

(**crs_1**, **pcs_2**) : $\underline{\text{CHAINED\_WITH}_R}$, **pcs_1** : river_section_with_bridge,

pcs_2 : river_section_with_bridge, **crg_1** : complex_river_segment,

crg_2 : complex_river_segment, **crg_3** : complex_river_segment,

(**pcs_1**, **crg_1**) : $\underline{\text{HAS\_R\_PART\_ONE}_R}$, (**pcs_1**, **crg_2**) : $\underline{\text{HAS\_R\_PART\_OTHER}_R}$,

b_1 : bridge, (**pcs_1**, **b_1**) : $\underline{\text{HAS\_B\_PART}_R}$,

(**b_1**, **crg_1**, **crg_2**) : *PRE_form_riverbridge_connection*

(**pcs_2**, **crg_2**) : $\underline{\text{HAS\_R\_PART\_ONE}_R}$, (**pcs_2**, **crg_3**) : $\underline{\text{HAS\_R\_PART\_OTHER}_R}$,

b_2 : **bridge**, (**pcs_2**, **b_2**) : $\underline{\text{HAS\_B\_PART}_R}$,

(**b_2**, **crg_2**, **crg_3**) : *PRE_form_riverbridge_connection.*

crg_1 : river_portion_with_island, **crg_2** : elementary_river_portion,

crg_3 : elementary_river_portion, **erp_1** : elementary_river_portion,

(**crg_1**, **erp_1**) : $\underline{\text{BUILD\_WITH}_R}$, **i_1** : **island,**

(**crg_1**, **i_1**) : $\underline{\text{BUILD\_WITH}_R}$, (**erp_1**, **i_1**) : *PRE_contain_in_middle* }

5.2 Expressing Map Knowledge

In this section $\mathcal{GALC}(\ell, \mathcal{D})$ is applied to the domain of map understanding. The whole process is expressed below.

Let dom($\mathcal{I}$) be an arbitrary set of all possible instances on a map in scope. We start with the definition of the primitive concepts used in map knowledge representations.

{ city, bar_line, road_segment, island, elementary_river_portion, simple_river_segment }

A concrete domain can be viewed as a mechanism for modeling primitive objects

and predicates, which are indispensable in a knowledge engineering system. The predicates represent procedural reasoning algorithms that work on concrete objects. It is obvious that all sorts of spatial objects and symbols can be built up with pixel level objects, such as points and chained line segments. The predicates are specified over the concrete objects. To decide whether a predicate is satisfied, the DL systems take various algorithms to compute positions of and relations among the objects. In this dissertation we define an admissible domain $\mathcal{D}_{map\_understanding}$, which is the union of a number of "smaller" admissible domains. One such domain is $\mathcal{D}_{geometry}$, which consists of a set of points and continuous line segments, and a set of predicates defining positions, geometrical properties, and spatial relations of objects.

The following is a set of axioms which make up the T-box, where concept names and predicate names are written in Sans serif font and role names written in Small caps font. Predicate names are prefixed with *PRE_*.

1. map $= \exists \underline{\text{CONTAINS}_R}$.roadnetwork_or_rivernetwork $\sqcap \exists \underline{\text{CONTAINS}_R}$.other_feature

   This axiom means that a map contains a set of instances, each of which can be either a road network or a river network, and other map features. Role $\underline{\text{CONTAINS}_R}$ is defined as such:

   a,b $\in$ dom($\mathcal{I}$); (a,b) $\in \underline{\text{CONTAINS}_R^{\mathcal{I}}}$ *iff* b $\in$ roadnetwork_or_rivernetwork

   or b $\in$ other_feature.

2. other_feature = isolated_lake $\sqcup$ buildings

The concept other_feature includes isolated lakes and buildings.

3. roadnetwork_or_rivernetwork = road_network ⊔ river_network

   This means a roadnetwork_or_rivernetwork instance is either a road network or a river network.

4. road_network = ∃ $\underline{Road\_Network\_Has_R}$. road_section

   A road_network can be viewed as being made of a collection of road_section instances. Note that there are two types of map phenomena; one is the physically visible objects, such as line segments and city symbols, the other is the invisible ones, such as gaps between broken line segments. The existence of invisible objects depends on other objects' existence, or the context. Ideally, we think of a joint as a simple point where more than two legs join. However, in our application, a road joint carries a meaning that takes into account the complex situations on real maps. For example, as shown in Figure 5.4, sometimes road legs join at a city, while other times roads are not exactly joined at a point (this occurs frequently). Therefore, the concept *joint* is an indispensable part in the proposed DL system.

   It may be asked why road_section instances are identified as the building blocks of a road network. Usually, a road network is made of a set of road legs and a set of joints. The role $\underline{Road\_Network\_Has_R}$ may be considered a ternary relation among road networks, joints and road sections. However, $\underline{Road\_Network\_Has_R}$

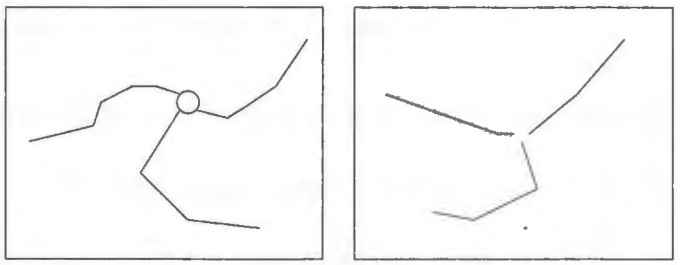


Figure 5.4: Two instances of road joints.

is defined as a binary role in this representation. It is better to view the **joint** instances as attributes of road networks, since the existence of a joint depends on the intersecting road sections.

5. road_section $= \exists$ $\underline{\texttt{Has\_As\_TwoEndPoints}_F}$. endpoints $\sqcap \exists$ $\underline{\texttt{Has\_As\_Leg}_F}$. leg $\sqcap$ *PRE_leg_joint*($\underline{\texttt{Has\_As\_Joint}_F}$ $\underline{\texttt{Pos}_F}$, $\underline{\texttt{Has\_As\_Leg}_F}$ $\underline{\texttt{First\_End}_F}$ $\underline{\texttt{Pos}_F}$, $\underline{\texttt{Has\_As\_Leg}_F}$ $\underline{\texttt{Other\_End}_F}$ $\underline{\texttt{Pos}_F}$)

   Feature $\underline{\texttt{Has\_As\_Joint}_F}$ refers to the joint part of a joint_leg_pair. $\underline{\texttt{Has\_As\_Leg}_F}$ refers to the leg part of a joint_leg_pair. Note that *PRE_Leg_Joint* is a predicate restriction. The feature chain $\underline{\texttt{Has\_As\_Joint}_F}$ $\underline{\texttt{Pos}_F}$ picks up the position of a joint. $\underline{\texttt{Has\_As\_Leg}_F}$ $\underline{\texttt{First\_End}_F}$ $\underline{\texttt{Pos}_F}$ means the position for one of the end points of the leg. The predicate *PRE_Leg_Joint* means that the position of the joint of a joint_leg_pair should coincide with one of the end points of its leg.

   Note that we can bestow on this predicate different meanings according to what types of constrictions we intend to have. For example, it can be defined as

   ($\underline{\texttt{Has\_As\_Joint}_F}$ $\underline{\texttt{Pos}_F}$) = ($\underline{\texttt{Has\_As\_Leg}_F}$ $\underline{\texttt{First\_End}_F}$ $\underline{\texttt{Pos}_F}$) $\vee$ ($\underline{\textit{Has\_As\_Joint}_F}$

$\underline{\texttt{Pos}_F}$) = ($\underline{\texttt{Has\_As\_Leg}_F}$ $\underline{\texttt{Other\_End}_F}$ $\underline{\texttt{Pos}_F}$)

Unfortunately, this is only the case with ideal situations. In most cases, it is important to allow for some variations. On maps, it is possible that a joint actually does not touch its connecting legs. Thus we have

dist((HAS_JOINT POS)-(HAS_LEG FIRST_END POS)) $\leqslant$ **threshold**

$\vee$ **dist**((HAS_JOINT POS)-(HAS_LEG OTHER_END POS)) $\leqslant$ **threshold**

This is a good example of separating representation from implementation. We can consider the application's declarative aspects first, and delay implementation to a later stage. As a result, flexibility is achieved in adapting representations to different situations.

6. leg = ($\exists$ $\underline{\texttt{Compose\_String\_Of}_R}$. part_of_leg $\sqcup$ $\exists$ $\underline{\texttt{Is\_Simple}_F}$. road_segment) $\sqcap$ $\forall$ $\underline{\texttt{More\_Parts}_R}$. $\bot$

   This means that a "leg" is composed of a string of leg parts, or just a simple road segment. The concept term $\underline{\texttt{More\_Parts}_R}$.$\bot$ means that the part_of_leg instance cannot connect to any more leg parts (already the longest among its class). Next we define what exactly these leg parts are.

7. part_of_leg = leg_part_with_gap $\sqcup$ leg_part_with_city $\sqcup$ leg_part_with_bridge

   Obviously this axiom defines that a leg part can be a leg_part_with_gap, or a leg_part_with_city, or a leg_part_with_bridge.

8. leg_part_with_gap = $\exists$ $\underline{\texttt{Has\_One\_End}_F}$. road_segments $\sqcap$ $\exists$ $\underline{\texttt{Has\_Other\_End}_F}$.

road_segment $\sqcap$ *PRE_forming_gap_inbetween*(Has_One_End$_F$, Has_Other_End$_F$)

This describes the scenario that a "leg part" instance in turn has one "road segment" instance at one end and another "road segment" at the other end, forming a "gap" in between. The predicate *PRE_forming_gap_inbetween* takes the corresponding geometric forms of the two "road segment" instances and checks whether they satisfy the criteria for forming a gap. Such criteria may require that the two road segments meet the following conditions: (1) They are close enough to each other; and (2) They follow each other's curve.

9. leg_part_with_city $= \exists$ Has_One_End$_F$. road_segment $\sqcap \exists$ Has_Other_End$_F$. road_segment $\sqcap \exists$ Has_In_Between$_F$. city $\sqcap$ *PRE_with_city_inbetween* (Has_One_End$_F$ Geom_Form$_F$, Has_Other_End$_F$ Geom_Form$_F$, Has_In_Between$_F$ Geom_Form$_F$)

   This axiom defines that a "leg part with city" instance is built with two "road segment" instances and one "city" instance that satisfy certain criteria.

10. leg_part_with_bridge $= \exists$ Has_One_End$_F$. road_segment $\sqcap \exists$ Has_Other_End$_F$. road_segment $\sqcap \exists$ Has_In_Between$_F$. bridge $\sqcap$ *PRE_with_bridge_inbetween* (Has_One_End$_F$, Has_Other_End$_F$, Has_In_Between$_F$)

    Axioms 6 to 10 share the same structure. The basic meanings behind them are more easily illustrated by a diagram in Figure 5.5. Note that we look at a part of a leg as having three smaller parts: one road segment at one end, another

road segment at the other end, and a bridge symbol in between.

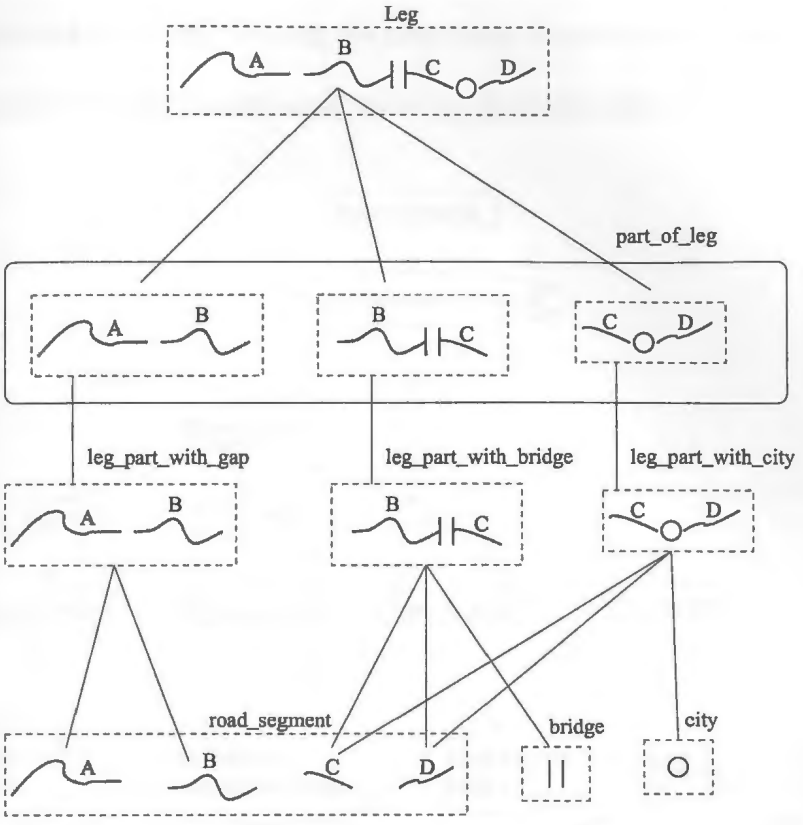


Figure 5.5: Meaning of a "leg" instance.

11. bridge $= \exists$ $\underline{\texttt{First\_Bar}_F}$. bar_line $\sqcap \exists$ $\underline{\texttt{Other\_Bar}_F}$. bar_line $\sqcap$ *PRE_Bridge* ($\underline{\texttt{First\_Bar}_F}$, $\underline{\texttt{Other\_Bar}_F}$)

    It is obvious that a bridge on a map is drawn with two bar lines. The predicate *PRE_Bridge* states the restriction that only two bar lines that are reasonably short and approximately parallel to each other can make a bridge symbol.

12. river_network $= \exists$ $\underline{\texttt{River\_Network\_Has}_F}$. joint_section_pair

    Just like road networks, a river network is made up of joints and sections. The

object-relationship graph in Figure 5.6 gives a general sketch of the concepts and relationships involved. During the knowledge engineering process, the accurate semantics of those concepts and roles are gradually refined.

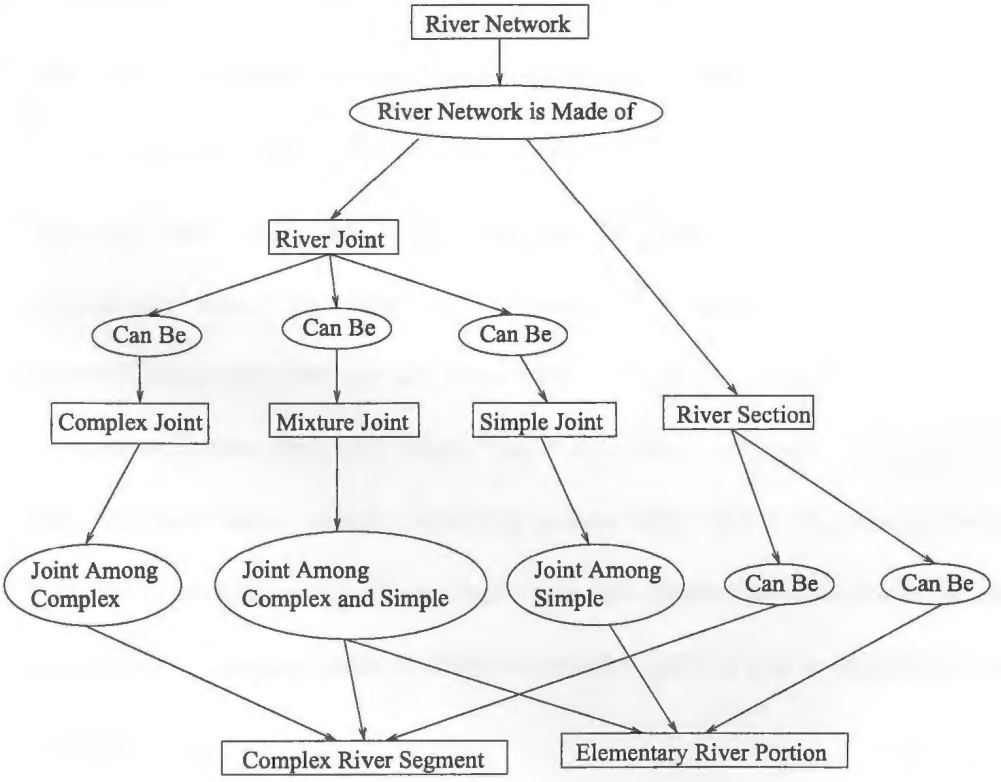


Figure 5.6: The object-relationship graph of river networks.

13. joint_section_pair $= \exists$ $\underline{\texttt{Has\_Joint}_F}$. river_joint $\sqcap \exists$ $\underline{\texttt{Has\_Section}_F}$. river_section $\sqcap$ (($\mathit{PRE\_Is\_Complex\_Joint}(\underline{\texttt{Has\_Joint}_F}) \sqcap \mathit{PRE\_Is\_Complex\_Section}$ $(\underline{\texttt{Has\_Section}_F}) \sqcap \mathit{PRE\_Complex\_Section\_Joint}(\underline{\texttt{Has\_Joint}_F}\ \underline{\texttt{Geom\_Form}_F}$, $\underline{\texttt{Has\_Section}_F}\ \underline{\texttt{Geom\_Form}_F})) \sqcup (\mathit{PRE\_Is\_Mix\_Joint}(\underline{\texttt{Has\_Joint}_F}) \sqcap$ $\mathit{PRE\_Is\_Mix\_Section}(\underline{\texttt{Has\_Section}_F}) \sqcap \mathit{PRE\_Mix\_Section\_Joint}(\underline{\texttt{Has\_Joint}_F}$

$\underline{\texttt{Geom\_Form}_F}$, $\underline{\texttt{Has\_Section}_F}$ $\underline{\texttt{Geom\_Form}_F}$))

$\sqcup$ (*PRE_Is_Simple_Joint*($\underline{\texttt{Has\_Joint}_F}$) $\sqcap$ *PRE_Is_Simple_Section*($\underline{\texttt{Has\_Section}_F}$)

$\sqcap$ *PRE_Simple_Section_Joint*($\underline{\texttt{Has\_Joint}_F}$ $\underline{\texttt{Geom\_Form}_F}$, $\underline{\texttt{Has\_Section}_F}$

$\underline{\texttt{Geom\_Form}_F}$)))

This axiom describes how a "joint section pair" instance can be constructed from "river joint" and "river section" instances.

Through role names $\underline{\texttt{Has\_Joint}_F}$ and $\underline{\texttt{Has\_Section}_F}$, it states that a "joint section pair" has a "river joint" and a "river section" as its parts. Three different types of joints and sections are considered: *complex*, *mixed*, and *simple*. River sections with two explicitly drawn banks are called *complex*, while those with only one bank called *simple*. Similarly, a spot where two complex sections join is called a complex joint. Two simple ones are connected with a simple joint. In particular, a mixed joint is where a complex section and a simple one come together.

Further details of the semantics involved in this axiom are defined by the axioms below.

14. river_joint = complex_joint $\sqcup$ mixed_joint $\sqcup$ simple_joint

To capture the meaning of river networks in detail, we first analyze different forms of intersections of a river system. Shown in Figure 5.7 are some typical river intersections.

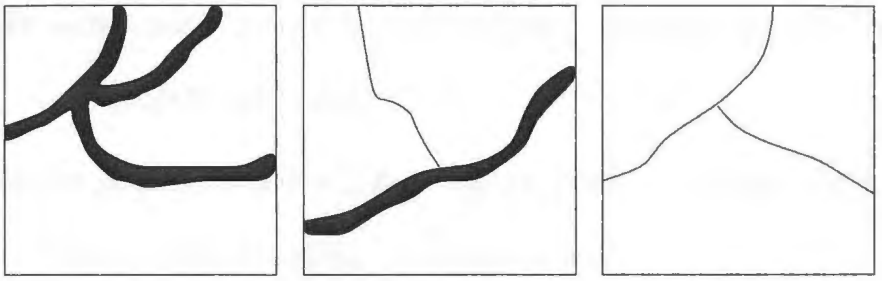


Figure 5.7: Different river intersections.

15. complex_joint $= \exists$ $\underline{\texttt{Complex\_Joint\_Connect}_R}$. complex_river_section

This axiom states that a "complex joint" instance connects a number of "complex river section" instances.

It is convenient to think that complex river sections are connected by pairs of articulations, as illustrated in Figure 5.8. We have to explicitly symbolize the end points (A_1, A_1', B_1, B_1', C_1, C_1') of the complex river sections.

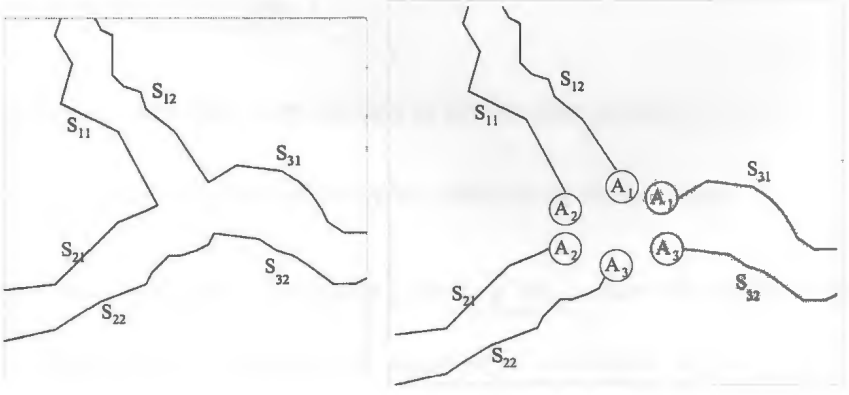


Figure 5.8: Articulation pairs.

Alternative terminological axioms can be used to represent the same map knowledge. For example, axioms 13 and 14 can be substituted with the following:

joint_section_pair = complex_joint_section_pair ⊔ mixture_joint_section_pair

⊔ simple_joint_section_pair

complex_joint_section_pair = ∃ $\underline{\texttt{Has\_Complex\_Joint}_F}$. complex_joint ⊓

∃ $\underline{\texttt{Has\_Complex\_Section}_F}$. complex_section ⊓

PRE_Complex_Section_Joint($\underline{\texttt{Has\_Complex\_Joint}_F}$,

$\underline{\texttt{Has\_Complex\_Section}_F}$)

mixture_joint_section_pair = ∃ $\underline{\texttt{Has\_Mixture\_Joint}_F}$. mixture_joint

⊓ *PRE_Mixture_Section_Joint*($\underline{\texttt{Has\_Mixture\_Joint}_F}$,

$\underline{\texttt{Has\_Mixture\_Section}_F}$)

simple_joint_section_pair = ∃ $\underline{\texttt{Has\_Simple\_Joint}_F}$. simple_joint

⊓ *PRE_Simple_Section_Joint*($\underline{\texttt{Has\_Simple\_Joint}_F}$,

$\underline{\texttt{Has\_Simple\_Section}_F}$)

16. river_section = complex_river_section ⊔ simple_river_section

River sections can be either complex sections or simple ones.

17. complex_river_section = (∃ $\underline{\texttt{Compose\_String\_Of}_R}$. part_of_complex_section

⊔ ∃ $\underline{\texttt{Is\_Complex}_F}$. complex_river_segment) ⊓ ¬ ∃ $\underline{\texttt{More\_Parts}_R}$. ⊤

This axiom states that a "complex river section" instance is composed of a string of "part of complex section" instances. The role name $\underline{\texttt{Compose\_String\_Of}_R}$ requires that the building parts of a complex river section be connected together in a consecutive manner.

The concept term $\neg \exists$ More_Parts$_R$. $\top$ describes those complex river sections that do not have any more other parts.

18. part_of_complex_section = complex_section_with_bridge_case1 $\sqcup$ complex_section_with_bridge_case2

    This axiom distinguishes two different ways in which a bridge is drawn: (1) with an explicit bridge symbol; and (2) implied in the context.

19. complex_section_with_bridge_case1 = $\exists$ Has_First_End$_F$. complex_river_segment $\sqcap \exists$ Has_Other_End$_F$. complex_river_segment $\sqcap \exists$ Has_In_Between$_F$. bridge $\sqcap$ *PRE_with_bridge_in_between1*(Has_One_End$_F$, Has_Other_End$_F$, Has_In_Between$_F$)

    This axiom states that a "complex river section with a bridge" instance should be made of three parts: a complex river segment at one end, another complex river segment at the other end, and a "bridge" symbol between the two river segments.

    Sometimes there is no bridge symbol drawn at the intersection of a road section and a river section. This situation is dealt with by the next axiom.

20. complex_section_with_bridge_case2 = $\exists$ Has_One_End$_F$. complex_river_segment $\sqcap \exists$ Has_Other_End$_F$. river_segment $\sqcap \exists$ Has_Road_Cross$_F$. road_segment $\sqcap$ *PRE_with_bridge_in_between2*(Has_One_End$_F$, Has_Other_End$_F$, Has_Road_Cross$_F$)

This axiom deals with the scenario when an explicit "bridge" instance is not drawn on the map.

21. complex_river_segment = river_portion_with_island $\sqcup$ elementary_river_portion

    A "complex river segment" instance appears either as a river portion with an island symbol in the middle or as an elementary river portion.

22. river_portion_with_island = $\exists$ $\underline{\texttt{Has\_River\_Portion}_F}$. elementary_river_portion
    $\sqcap \exists$ $\underline{\texttt{Has\_Island}_F}$.island $\sqcap$ $PRE\_island\_in(\underline{\texttt{Has\_River\_Portion}_F}, \underline{\texttt{Has\_Island}_F})$

    A "river portion with island" instance in turn is made of an elementary river portion with an island in the region covered by the river portion.

23. simple_river_section = $\exists$ $\underline{\texttt{Compose\_String\_Of}_R}$. part_of_simple_section
    $\sqcap \neg \exists$ $\underline{\texttt{More\_Parts}_R}$. $\top$

    This axiom tells us a simple river section is assembled with a string of parts and contains no other parts.

24. part_of_simple_section = simple_section_with_bridge $\sqcup$ simple_section_with_gap

    A "part of simple section" instance is either a "simple section with bridge" or a "simple section with gap".

25. simple_section_with_bridge = simple_section_with_bridge_case1
    $\sqcup$ simple_section_with_bridge_case2

We distinguish two cases in the next two axioms: (1) the presence of a bridge on the river section is explicitly represented by a bridge symbol; and (2) the presence of a bridge is implied in the context.

26. simple_section_with_bridge_case1 $= \exists$ $\underline{\texttt{Has\_One\_End}_F}$. simple_river_segment $\sqcap \exists$ $\underline{\texttt{Has\_Other\_End}_F}$. simple_river_segment $\sqcap \exists$ $\underline{\texttt{Has\_In\_Between}_F}$. bridge $\sqcap$ *PRE_with_bridge_in_between*($\underline{\texttt{Has\_One\_End}_F}$, $\underline{\texttt{Has\_Other\_End}_F}$, $\underline{\texttt{Has\_In\_Between}_F}$)

    In this scenario, a simple river section is made of two simple river segments with an explicitly drawn bridge symbol sitting in between.

27. simple_section_with_bridge_case2 $= \exists$ $\underline{\texttt{Has\_One\_End}_F}$. simple_river_segment $\sqcap \exists$ $\underline{\texttt{Has\_Other\_End}_F}$.simple_river_segment $\sqcap \exists$ $\underline{\texttt{Has\_Road\_Cross}_F}$.road_segment $\sqcap$ *PRE_with_bridge_in_between*($\underline{\texttt{Has\_One\_End}_F}$, $\underline{\texttt{Has\_Other\_End}_F}$, $\underline{\texttt{Has\_Road\_Cross}_F}$)

    In this scenario, there is no explicitly drawn bridge symbol in between the two simple river segments. However, since the simple river segments intersect with a road segment, a reasonable conclusion can be drawn about the presence of a bridge.

28. simple_section_with_gap $= \exists$ $\underline{\texttt{Has\_One\_End}_F}$. simple_river_segment $\sqcap \exists$ $\underline{\texttt{Has\_Other\_End}_F}$. simple_river_segment $\sqcap$ *PRE_forming_gap_between*($\underline{\texttt{Has\_One\_End}_F}$, $\underline{\texttt{Has\_Other\_End}_F}$)

Duo to the quality of low-level image processing algorithms, a continuous road may appear as a number of separated line segments. To overcome this issue, we consider two simple river segments as belonging to the same river section as long as they comply with the restrictions implied in the predicate *PRE_forming_gap_between*.

5.3 Summary of Typical Concepts and Roles

From the above discussions, it is seen that in order to deal with spatial objects and their relations, features and phenomena of interest have to be explicitly symbolized. In this section, the semantics of some typical concepts and roles used for modeling spatial knowledge will be summarized.

- $\underline{\texttt{Chained\_With}_R}$: a finite set of objects (called chaining-objects) $\{\gamma_1, \cdots, \gamma_w\}$, is said to form a chain *iff* these objects $(\gamma_1, \cdots, \gamma_w)$ can be arranged into an ordered series $\gamma_1', \cdots, \gamma_w'$ (where $\gamma_i' \in \{\gamma_1, \cdots, \gamma_w\}$) so that

  1. each chaining-object γ_i' is associated with a pair $(h_1(\gamma_i'), h_2(\gamma_i'))$ (called hooks), and
  2. there exists a predicate $\mathbf{P}$ (link condition) such that $\mathbf{P}(h_2(\gamma_1'), h_1(\gamma_2'))$, $\mathbf{P}(h_2(\gamma_2'), h_1(\gamma_3')), \cdots, \mathbf{P}(h_2(\gamma_{w-1}'), h_1(\gamma_w'))$.

  We define $\underline{\texttt{Chained\_With}_R}^{\mathcal{I}}$ as $\{(\mathrm{c}, \gamma) \mid \gamma$ is one of the chaining objects that form the chain c.$\}$

- $\underline{\texttt{Contains}_R}$: this is a one-to-many relation, which represents the relation among a collection instance and its member instances. An instance e is viewed as a collection of some other instances, then

  $\underline{\texttt{Contains}_R}^{\mathcal{I}} = \{(\text{e, y}) \mid \text{y belongs to the collection represented by e.}\}$

- $\underline{\texttt{Networked\_With}_R}$: a number of instances $o_1, \cdots, o_v$ are said to be networked *iff* they meet the following constraints:

  1. Each o_i, i=1, $\cdots$, v, has a pair $\langle\, p_1(o_i), p_2(o_i)\,\rangle$ (end points).

  2. No two such instances are identical, i.e., $o_i \neq o_j$ *iff* $i \neq j$, $i,j = 1, \cdots, v$.

  3. The instances are connected either directly or indirectly with one another. Two different instances are directly connected with each other *iff* they have joined end points, i.e.,

     $$\text{CONNECTED}(o_i,\ o_j) \equiv \text{JOINED}(p_1(o_i),\ p_1(o_j)) \cup \text{JOINED}(p_1(o_i),\ p_2(o_j)) \cup \text{JOINED}(p_2(o_i),\ p_1(o_j)) \cup \text{JOINED}(p_2(o_i),\ p_2(o_j))$$

     Two different instances o_i and o_j are indirectly connected with each other *iff* there exists a (non-empty) set of instances o_{k+1}, o_{k+2}, $\cdots$, o_{k+q} such that

     $$\text{CONNECTED}(o_i,\ o_{k+1}) \cup \text{CONNECTED}(o_{k+1},\ o_{k+2}) \cup \cdots \cup \text{CONNECTED}(o_{k+q},\ o_j).$$

Chapter 6

Map Understanding Process

In previous chapters we have shown how to formalize map knowledges. However, to achieve the task of map understanding, we have to employ an appropriate reasoning mechanism that works effectively on the basis of DL representations. One of the desirable practices used by knowledge-based system designers is to separate the knowledge representation as much as possible from the reasoning mechanism. In this chapter, the reasoning mechanism based on a DL specification will be discussed. It will be shown that just like the formalization of knowledge representations, the reasoning process of map understanding can be formally defined. In particular, it will be shown that the map understanding process is actually a process that builds a so called **complete** object-instance-relation (**OIR**) graph.

6.1 Understanding of Maps

In order to understand maps, it takes a considerable effort to process raw image data and draw inferences based on the domain knowledge. Understanding maps means being able to do more than simply finding the facts, such as, "How big is the area covered by the map?" or "How many colors are used in the map making?". Most frequently asked questions about a map are these: (1) Where is the location of a specific object? (2) How to travel there? and (3) What is the distance from here to that place? The visual information presented to a map reader does not answer all the questions directly. He has to identify, search, match, index, correlate, analyze, guess, confirm, and summarize the facts implicitly shown on the map. In a sense, a map may be viewed as a special kind of database. However, it is not organized in lists of data records, but presented in an analog format that needs to be processed by human eyes, a sophisticated biological vision system. This database holds spatial information and serves as a guidance tool. It will be frequently referred to when answering those questions mentioned above. One thing to note is that each of these questions concerns just a portion of a map. To answer them does not require dealing with all the information stored in the map. This means that map understanding depends on the tasks it intends to achieve. Different understanding tasks may involve different instances, attributes, and relations.

In this section, the term "understanding" will be defined in a formal fashion first. The word "understanding" is widely used in human communication. Yet it is very

difficult to define and evaluate. There is no generally agreed-upon scientific meaning. However, it is possible to explore some of the significant aspects of what are involved. Instead of trying to define a general meaning, we concentrate our attention on what it means by understanding in the context of GIS and similar computer applications. An understanding process can be described by telling what kind of output it produces. A map is understood by a computer system when its output has adequately revealed the information (either explicit or implicit) that the maker of the map intends to convey. The output can give a highly structured and classified description, based on which other applications can carry out further analysis. It will be shown that a complete **OIR** graph introduced later in this section can be such an understanding result. In general, an understanding of a map is nothing more than revealing map objects and relationships.

We formalize the term *understanding* through the following definition.

Definition 6.1 (Understanding task)

An understanding task **T** *is a pair (T, OIR(T)), where T is a concept term, and OIR(T) is a complete object-instance-relation graph that covers T.*

Since a new concept name can be defined using a concept term, sometimes we also call T the task concept. We will first show that any understanding task centers around a concept T, whose formal definition has been given in Appendix A. The output of an understanding process is a graph structure OIR(T). In the next section, the formal definition of **OIR** will be given.

In Description Logics systems, a concept is defined as a subset of $\mathrm{dom}(\mathcal{I})$, and thus it virtually can refer to any set of instances with some common properties. Any phenomenon (or phenomena) on a map can be viewed as a concept. Therefore, it can be said that to understand a map phenomenon (or a group of phenomena) is actually to understand a concept in DL systems in an abstract sense. Understanding a concept means revealing the instances of this concept as well as the possible instances that are related to it either directly or indirectly. The following examples are used to illustrate this idea.

Task 6.1 *Suppose we intend to store a map into a GIS in digital format, which means we have to understand the phenomenon "map". From discussions in Section 5.2, we know that all sorts of map phenomena of interest can be represented in concepts or concept terms.*

Such a task can be represented by a task concept *Map* and a complete **OIR** that covers *Map*. The **OIR** graph includes different abstract levels of map concepts and roles that are related to the concept *Map*, where all the relevant concept and role instances are explicitly represented. Due to the varying usages of digital maps, different GIS systems may put their emphasis on different types of information to capture.

Task 6.2 *Suppose we are given a map and want to drive from New York city to Los Angles by the shortest route. We have to understand relevant phenomena on the*

map. All those meaningful things involved in our understanding can be abstracted into concepts or concept terms. The concepts we identify for this task are these: city, road section, path, path between New York to Los Angles, and the shortest path between them.

Note that we also take into consideration those concepts which are usually computed using procedural algorithms, such as *the shortest path between two cities.*

6.2 Object-Instance-Relation Graphs

In this section, an introduction to **OIR** graphs is presented, which is a special structure we propose to be used for representing a map understanding process, as well as for describing and analyzing map understanding systems.

Definition 6.2 *(Object-Instance-Relation graph).*
*An object-instance-relation graph (***OIR***) is a graph* $\mathbf{T} = \langle \mathbf{O}, \mathbf{I}, \mathbf{G_I}, \mathbf{N_R}, \boldsymbol{Prim} \rangle$, *where*

1. $\mathbf{O} = \{ o_1, o_2, ..., o_m \}$ *is a set of* ***instance holders.***

2. $\mathbf{I} = \{ i_1, i_2, ..., i_n \}$ *is a set of* ***instances.***

3. *A* ***placement function*** $\mathbf{G_I}$ *is a partial mapping:*

   $\mathbf{G_I}$: $\mathbf{I} \rightarrow \mathbf{O}$

4. $\mathbf{N_R}$ *is a set of* ***relationships****, and each of its elements* $\mathbf{R} \subseteq \mathbf{I} \times \mathbf{I}$ *is a set of* ***edges****.*

5. $\boldsymbol{Prim} \subset \boldsymbol{O}$ *is a set of primitive instance holders.*

An **OIR** graph is actually a graphical illustration of object classes and relationships among them. Each element in **O** is called an instance holder, which corresponds to a concept name. The individuals in a given map world compose the set **I**. The placement function $\mathbf{G_I}$ is a many-to-one mapping that assigns instances to each instance holder in the **OIR** graph. $\mathbf{N_R}$ may be viewed as the set of role names in a DL system. In a graphical illustration of an **OIR** graph, instance holders are drawn as ovals, and object instances drawn as small black boxes. The relation instances are represented by edges between object instances. The primitive instance holders are represented by ovals with bold lines. Figure 6.1 gives an example of an **OIR** graph.

OIR graphs can be viewed as an alternative representation of maps. According to an **OIR**, a new map can be created as long as those map features and relations of interest are specified in the **OIR**. Multiple **OIR**s that describe different regions can be merged to form a larger map. Take the simplified map in Figure 6.2 as an example. The **OIR** graph in Figure 6.1 can depict, in a declarative manner, the object instances on this map and their relations at different abstract levels. At this stage, we contend ourselves with the current **OIR** depiction by just demonstrating the existence of certain instances and relations. The detailed properties or the nature of individual instances and relations are absent. In the proposed methodology, an **OIR**

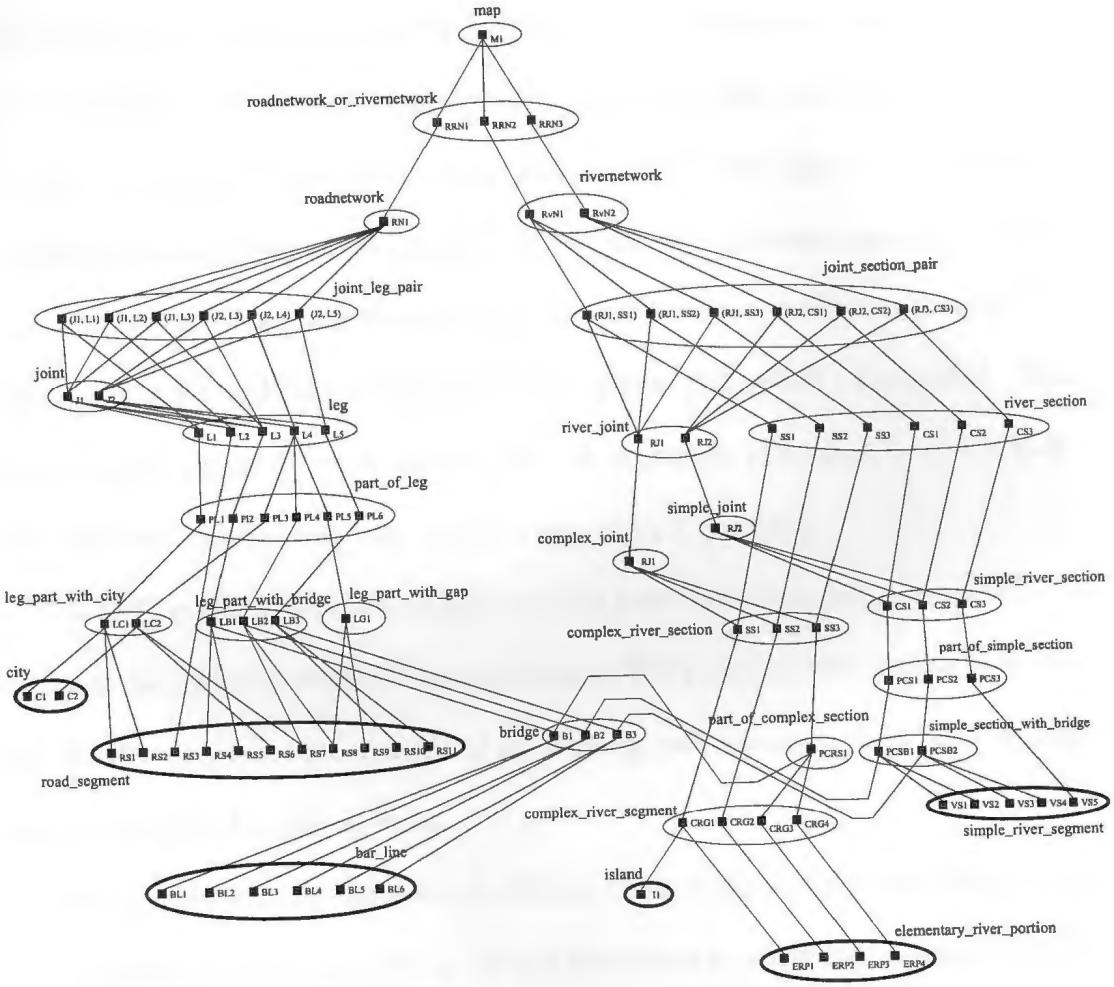


Figure 6.1: An example of an **OIR** graph.

only serves as a tool to capture proceedings of a map understanding process. The internal representation of a DL system deals with details of instances and relations. As can be seen in Figure 6.1, the instances are drawn inside the concept holders, indicating that they are assigned to these holders. Each concept holder has an associated *concept name*, indicating its corresponding concept in the DL system. Each instance is also associated with a name, which corresponds to its instance name in the

assertional language. Hence, an **OIR** graph can be looked at as a visual version of a set of assertional axioms. For example, an instance **B1** in the instance holder labeled **bridge** can be represented as an axiom: **B1** : **bridge**. Each edge in an **OIR** graph corresponds to an axiom of the form (a, b) : **R**, where a, b are two concept instances connected by the edge, and the edge is a relation instance that belongs to **R**. The edge between **B1** and **BL1** corresponds to the axiom (**B1**, **BL1**) : ONE_BAR[1]. Note that a terminological axiom of the form $\mathbf{A} = \mathbf{B}$ is actually equivalent to $\mathbf{A} = \forall\, \mathbf{R}.\mathbf{B}$, where **R** is an identity role, that is, $\mathbf{R}^{\mathcal{I}} = \{(s, s) \mid s \in \mathrm{dom}(\mathcal{I})\}$.

The **OIR** graph reveals facts of interest to us from the simple map in Figure 6.2. It indicates that the map contains one road network **RN1** and two river networks **RvN1** and **RvN2**. **RN1** in turn is made of six joint_leg_pair instances. Each joint_leg_pair has a joint and a leg, and so forth.

Solving a problem by computer is different from doing so by human beings. For a human being to solve a problem, he can take steps to reach the solution without a complete specification of the problem. Computers can only achieve a solution by manipulating clearly defined representations. A complete **OIR** graph has the following characteristics:

1. It is a symbolic representation, which provides a basis for computing solutions.

2. It provides an explicit and tangible description of concept and role instances

[1]For the sake of simplicity, the edges in Figure 6.1 are not labeled with corresponding role names. However, the role names can be figured out based on the participating concepts.

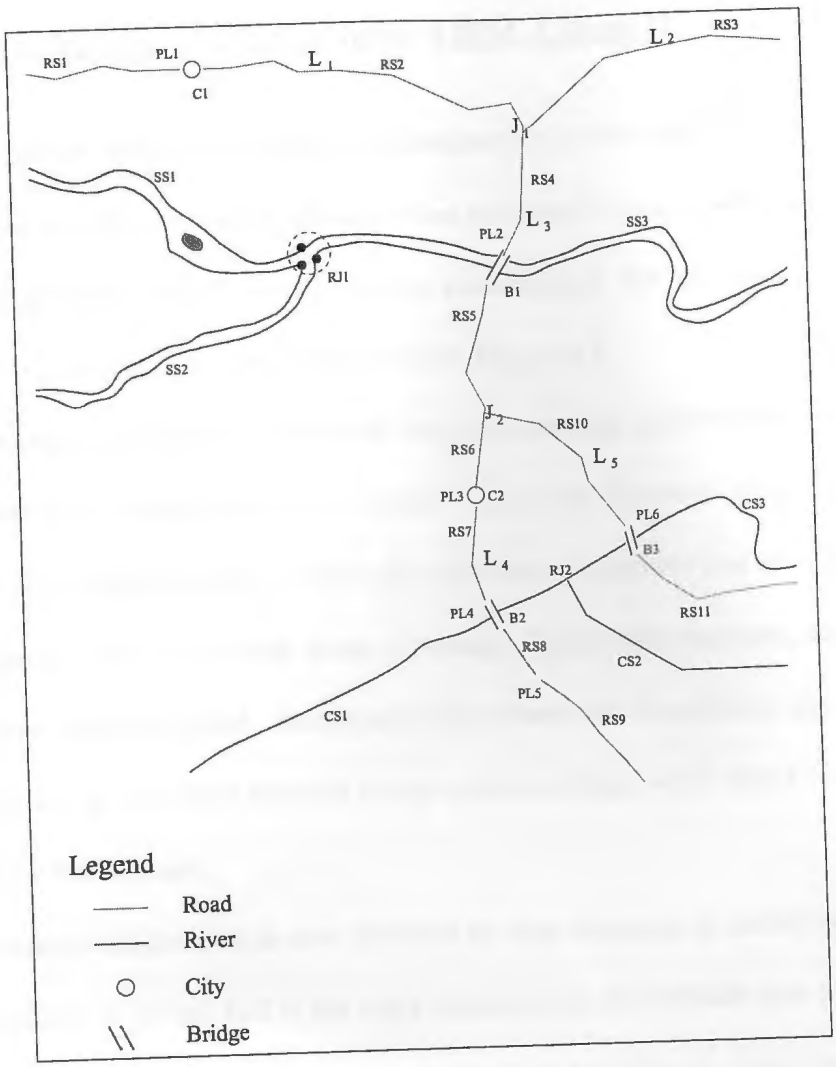


Figure 6.2: A simple test map.

that are implicitly encoded in a map.

The **OIR** graph in Figure 6.1 is a complete **OIR** graph, which is formally defined in the next section.

6.3 Building a Complete OIR Graph

In this section we shall give an informal introduction to the map understanding process. This is done by means of an example that performs the task of storing the simple map in Figure 6.2 in digital format. Such a task requires the explicit representation of the map objects and relations illustrated in Figure 6.1.

As discussed in Chapter 1, low-level image processing methods are first applied against raw map images in order to obtain elementary features. Before we start a high-level map understanding process, all other derived features and relations remain encoded in the raw map except those instances of primitive concepts, such as city, bar_line, road_segment, island, elementary_river_portion, and simple_river_segment. This can be viewed as the initial state of a map understanding, which can be represented by an initial **OIR** graph.

An instance holder that is not occupied by any instances is called **empty**. An instance holder is called **full** if no more instances in the domain can be added to it. An **OIR** graph whose primitive instance holders are full and whose non-primitive holders are empty is called an **initial OIR**.

As can be seen from the preceding discussion, the output of a map understanding process is a complete **OIR** graph in Figure 6.1. From an abstract viewpoint, the understanding of the simple map is indeed a process that starts from an initial **OIR** graph, goes through a series of understanding steps, and finally, builds up the complete **OIR** graph. This can also be considered an incremental structuring process.

In Figure 6.1, there are thirty **instance holders**, among which six are primitive instance holders. In the beginning, only primitive instance holders have instances inside, while all the other instance holders, which are also called derived instance holders, are empty. During the understanding process, the number of instances in derived instance holders may increase, which indicates that various derived instances are made explicit and assigned to the corresponding holders.

An understanding step can be defined as an action that reveals new information. Such an action can put a "new" instance into an instance holder, which indicates a concept instance being made explicit. It can also be one that adds an edge between two existing instances, which indicates that a relation between the two instances has been established. Each understanding step decides an action to take based on the current state of the **OIR** graph. Consider, for example, the six bar_line instances, **BL1**, $\cdots$, **BL6**, shown in Figure 6.2. Suppose that an algorithm is used to decide whether two bar lines constitute a bridge[2]. **BL1** and **BL2** are determined to be such a pair of bar lines. This fact can be denoted by

$$(\mathsf{BL1}, \mathsf{BL2}) \in \mathit{PRE\_Bridge}(\underline{\mathtt{One\_Bar}_F}, \underline{\mathtt{Other\_Bar}_F})$$

According to axiom 11, we know that a bridge is defined as something that has two bar lines which fulfill the predicate $\mathit{PRE\_Bridge}(\underline{\mathtt{One\_Bar}_F}, \underline{\mathtt{Other\_Bar}_F})$. Thus we can put a new instance **B1** in the holder **bridge**, and add two edges (**B1**, **BL1**)

[2]There are different algorithms for this purpose, depending on how bridge symbols are drawn on maps. One method is to check their relative positions and their parallelism.

and (**B1**, **BL2**) to the graph, indicating that bridge **B1** is made of **BL1** and **BL2**. This understanding step produces new instances and edges. Following this step, suppose that bridge **B1** is connected to a road segment **RS4** at one side and **RS5** at the other side. It is known that **B1**, **RS4** and **RS5** make up a leg_part_with_bridge instance. Then an instance **LB1** is placed in the **leg_part_with_bridge** holder, and three edges, (**LB1**, **B1**), (**LB1**, **RS4**), and (**LB1**, **RS5**), are added. Therefore, we can keep on exploring for new instances and relations, until the **OIR** graph gradually evolves into a complete **OIR** graph in Figure 6.1.

In DL systems, each concept participates in at least one relationship (role). Some concepts play an indispensable part in other concepts. Given a terminological axiom $C = D$, let DefC(C) denote the set of concepts occurring in D, and let DefR(C) denote the set of roles occurring in D. The concepts in DefC(C) are called the ***defining concepts*** of C, and the roles in DefR(C) are said to be the ***defining roles*** of C. Let **a** be an instance of C. The ***defining relations*** of **a** are the role instances in DefR(C) that specify relations among **a** and certain instances in DefC(C). The instances in DefC(C) that are related to **a** are called its ***defining instances***.

Initially, the instance holder of C is empty, since it is a derived concept. In many cases, the instance holders of its defining concepts are also empty. After a number of understanding steps, the holders of its defining concepts are starting to get populated with explicitly recognized instances. Typically, when certain instances of the defining concepts are available, they may suggest the existence of some instances of the derived

concept. For example, when a line segment in red color is recognized, a conclusion can be drawn that there exists a road section instance. There is no need to wait until all the line segments that compose the road section are recognized. The recognition of a new instance requires that all its defining instances and relations be explicitly represented. A number of understanding steps are usually needed. We distinguish two recognition states for the explicitly represented instances in an **OIR**: full and partial. If an instance is placed in a holder without having all its defining instances and relations explicitly shown, it is labeled as a partially recognized instance; otherwise it is called fully recognized. In an **OIR**, the relations among a derived instance and its defining instances are characterized by a set of edges that are called ***assembly edges***. A fully recognized instance is therefore an instance with all its assembly edges explicitly shown.

OIR with respect to task C

Each map understanding task concerns a specific set of concepts and roles. It is not necessary to discover the individuals and relations irrelevant to the task concept. In other words, only those concepts and roles that directly or indirectly define the task concept need to be considered. $\mathsf{DefC}^{+}(C)$ is called the ***defining concept closure*** of C, which is obtained by recursively substituting any concept in $\mathsf{DefC}(C)$ with its direct defining concepts until no new concept name can be added. All those terminological axioms relevant to C form a T[C]-box, which is a subset of the T-box.

Given a task concept C, an **OIR** graph with respect to a T[C]-box is constructed

as follows:

- Start with creating a new instance holder C_H in correspondence to C.
- For each newly added instance holder C_H, repeat the following procedure until no new holders are available:

  Consider its associated terminological axiom. If it is in the form $\mathrm{C} = E_1 \sqcup \cdots \sqcup E_k$ or $\mathrm{C} = E_1 \sqcap \cdots \sqcap E_k$, add new instance holders corresponding to concept terms $E_1, \cdots, E_k$, $\mathrm{k} > 1$.

  If it is in the form $\mathrm{C} = \exists\, \mathrm{R}.D_1 \otimes \cdots \otimes D_k$, add new instance holders corresponding to concept terms $D_1, \cdots, D_k$.

An understanding task **C** can be regarded as a special concept, that is, an instance holder in an **OIR**. Based on the above discussion, the term "complete **OIR** graph" with respect to a task is defined. Given a task **C**, the complete **OIR** with respect to **C** is defined as an **OIR** graph $\mathbf{G} = \langle \mathbf{C} \cup \mathsf{DefC}^{+}(\mathbf{C}), \mathbf{I}, \mathbf{G_I}, \mathbf{N_R}, \mathbf{Prim} \rangle$, where

1. All the instance holders are full.
2. All the instances in **I** are fully recognized.

This means that all the instances and their assembly edges of interest are explicitly shown in a complete **OIR**.

6.4 Augmentation Rules

One advantage of **OIR** approaches is that representation and reasoning are separated and both are provided explicit descriptions. Providing explicit descriptions for both knowledge representation and reasoning is very important for capturing complex semantics of an application. **OIR** graphs provide a satisfactory representation of instances and their relations, which define the state of an understanding process. The understanding process starts from the initial state with only primitive instances recognized, followed by a series of understanding steps, each of which recognizes some new instances and relations. This results in the **OIR** graph representing the current state of understanding being transformed to a new **OIR** graph with some new instances and edges added. The transformations of **OIR** graphs reflect the track of an understanding process. Up to this point the remaining issue is how to explicitly describe the understanding process.

Each understanding step is regarded as an augmentation action. The *augmentation rules* for **OIR** graphs, which are used to guide the understanding process, are presented below.

Definition 6.3 *(**OIR** augmentation rule)*

*Let **OA** be the set of abstract objects, let $\mathcal{A}$ be an initial **OIR**, C, D be instance holders, R be a role name, and a, b, c be instances. Also suppose there exist derived instance holders corresponding to $C \sqcup D$, $C \sqcap D$, $\exists\, R.C$, and $\forall\, R.C$. The set of **augmentation rules** are presented as follows:*

1. *If $a : C$, $a : D$ are in $\mathcal{A}$, and $a : C \sqcap D$ is not in $\mathcal{A}$, then add $a : C \sqcap D$ to $\mathcal{A}$.*

2. *If $a : C$ or $a : D$ is in $\mathcal{A}$ and $a : C \sqcup D$ is not in $\mathcal{A}$, then $a : C \sqcup D$ is added to $\mathcal{A}$.*

3. *If axioms $b : C$, $(a, b) : R$ are in $\mathcal{A}$, and $a : \exists R.C$ is not in $\mathcal{A}$, then add $a : \exists R.C$ to $\mathcal{A}$.*

4. *Let rel(a, R) be defined by $\{x \mid x \in dom(\mathcal{I})$ and $(a, x) \in R\}$. If $rel(a, R) \subseteq C$, then add $a : \forall R.C$ to $\mathcal{A}$.*

5. *If the following holds:*

   *For the feature chains $u_i = f_{i1} \cdots f_{in_i}$, for $i = 1, \cdots, n$, there are instance names $b_{i1}, \cdots, b_{in_i-1} \in$ **OA** and x_i, for $i = 1, \cdots, n$, such that the A-box $\mathcal{A}$ contains axioms $(a, b_{i1}) : f_{i1}$, $(b_{i1}, b_{i2}) : f_{i2}, \cdots$, $(b_{in_i-1}, x_i) : f_{in_i}$, and $(x_1, \cdots, x_n) : P$.*

   then add axiom $a : P(u_1, \cdots, u_n)$ to $\mathcal{A}$.

The above augmentation rules are derived from the definitions of concept terms. Take rule 3 as an example. Assume that $b : C$ and $(a, b) : R$ are in $\mathcal{A}$ and $a : \exists R.C$ is not in $\mathcal{A}$, according to the definition of concept term $\exists R.C$, we have $a : \exists R.C$. Note that rules 1 to 5 state the construction of **OIR** graphs in a "bottom-up" fashion. Current research papers only discuss the "top-down" rules, which are used

in the consistency test algorithm. In this dissertation, we introduce the "bottom-up" rules, which are useful in constructing concepts from other defined concepts.

An important feature of DL systems is that there exists a *sound* and *complete* algorithm which is able to verify the *consistency* of an A-box of $\mathcal{ALC}(\mathcal{D})$. An A-box $\mathcal{A}$ is *consistent* if it does not have contradictions. One question may be asked: Do the augmentation rules in **Definition** 6.3 preserve consistency? We call an augmentation rule *consistency preserving* if, after applying the rule to an A-box $\mathcal{A}$, the resulting A-box is still consistent. Next we shall prove that they do preserve consistency.

Theorem 6.1 *The augmentation rules in* **Definition** *6.3 are consistency preserving.*

Proof. Let $\mathcal{A}$ be a consistent A-box of $\mathcal{ALC}(\mathcal{D})$, and $\mathcal{A}_1$ be the A-box obtained from $\mathcal{A}$ by applying a given augmentation rule. To prove that $\mathcal{A}_1$ is also a consistent A-box, we have to demonstrate that no contradictory concept terms are implied. The conclusion can be reached by showing that the contrary leads to a contradiction. Since each augmentation rule adds some concept terms to $\mathcal{A}$, we just need to prove those added terms do not contradict to other concepts terms in $\mathcal{A}$.

(a). Augmentation rule 1 adds $a : (C \sqcap D)$ to $\mathcal{A}$. If $a : (C \sqcap D)$ contradicts with any concept term in $\mathcal{A}$, then $a : \neg(C \sqcap D)$ is implied by $\mathcal{A}$. Since $a : C$ and $a : D$ are also in $\mathcal{A}$, they are in contradiction with $a : \neg(C \sqcap D)$. Thus, $a : \neg(C \sqcap D)$ should not be implied in $\mathcal{A}$. Then the case with rule 1 is proved.

(b). Augmentation rule 2 adds $a : (C \sqcup D)$ to $\mathcal{A}$. If it contradicts with any concept term in $\mathcal{A}$, then $a : \neg(C \sqcup D)$ must be implied by $\mathcal{A}$. If either $a : C$ or $a : D$

is in $\mathcal{A}$, it will contradict with $a : \neg(C \sqcup D)$. Thus $a : \neg(C \sqcap D)$ should not be implied in $\mathcal{A}$.

(c). Augmentation rule 3 adds $a : \exists R.C$ to $\mathcal{A}$. If it contradicts with any concept term in $\mathcal{A}$, then $a : \neg\exists R.C$ is implied in $\mathcal{A}$. $a : \neg\exists R.C$ is equivalent to $a : \forall R.\neg C$. Since $(a, b) \in R$ is also in $\mathcal{A}$, according to the definition of $\forall R.\neg C$, we have $b \in \neg C$. This is contradictory to $b : C$, which is also in $\mathcal{A}$.

(d). Augmentation rule 4 adds $a : \forall R.C$ to $\mathcal{A}$. If it contradicts with any concept term in $\mathcal{A}$, then $a : \neg\forall R.C$ is implied in $\mathcal{A}$. $a : \neg\forall R.C$ is equivalent to $a : \exists R.\neg C$. This means that there exists an instance c such that $(a, c) \in \neg C$. This contradicts with the fact $\text{rel}(a, R) \subseteq C$.

(e). Augmentation rule 5 adds $a : \text{P}(u_1, \cdots, u_n)$ to $\mathcal{A}$. If it contradicts with any concept term in $\mathcal{A}$, then we need to consider the following two cases:

(1) the conjunction

$$\bigwedge_{i=1}^{k} P_i(\underline{x^{(i)}})$$

is not satisfiable in $\mathcal{D}$. Here $\underline{x^{(i)}}$ denotes $(x_1^{(i)}, \cdots, (x_{n_i}^{(i)})$, and $P_1(\underline{x^{(1)}}) = \text{P}(u_1, \cdots, u_n)$. This is not possible since by definition the satisfiability problem for finite conjunctions of the above mentioned form is decidable.

(2) $a : \neg\text{P}(u_1, \cdots, u_n)$ is implied in $\mathcal{A}$. By definition this means that there exist $r_1, \cdots, r_n \in \text{dom}(\mathcal{D})$ such that $u_1^{\mathcal{I}}(a) = r_1, \cdots, u_n^{\mathcal{I}}(a) = r_n$ and $(r_1, \cdots, r_n) \in \neg P$. Since the following condition holds:

For the feature chains $u_i = f_{i1} \cdots f_{in_i}$, for $i = 1, \cdots, n$, there are instance names $b_{i1}, \cdots, b_{in_i-1} \in \mathbf{OA}$ and x_i, for $i = 1, \cdots, n$, such that the A-box $\mathcal{A}$ contains axioms $(a, b_{i1}) : f_{i1}$, $(b_{i1}, b_{i2}) : f_{i2}, \cdots$, $(b_{in_i-1}, x_i) : f_{in_i}$, and $(x_1, \cdots, x_n) : \mathrm{P}$.

By definition, the feature chains $u_1, \cdots, u_n$ are partial functions, so they uniquely decide $r_1, \cdots, r_n$. Then we have $r_1 = u_1, \cdots, r_n = u_n$. So $(x_1, \cdots, x_n) : \mathrm{P}$ contradicts with $(r_1, \cdots, r_n) \in \neg P$.

It can be seen that the concept terms produced by these rules do not contradict with those already existing in $\mathcal{A}$. Thus, their consistency preserving property has been proved.

■

Theorem 6.1 plays a substantial role in guaranteeing a consistent understanding process. It has been mentioned that an understanding process starts with a set of explicitly represented primitive instances. It is obvious that these primitive instances correspond to an A-box $\mathcal{A}_{prim}$, which contains only axioms of the form $a_i : C$, where a_i is a primitive instance. According to the consistency checking algorithm (Algorithm 4.1 in Chapter 4), $\mathcal{A}_{prim}$ is a complete A-box. By applying the augmentation rules, various instances and relations are recognized successively, until no further instances and relations can be added. Since each augmentation preserves consistency,

the outcome after a series of augmentations is still consistent. This means a complete **OIR** graph, which is built as a result of the map understanding process, is consistent.

Chapter 7

Mapping Description Logics to Grammatical Representation

In the previous chapters, we have concentrated on issues concerning knowledge representation for map understanding. An equally important problem is how to conduct the reasoning process on the basis of explicitly represented knowledge. One important point to note is that the knowledge represented is of little use if it does not support an efficient mechanism for reasoning. It is not enough to only express the facts and truth about map objects and their relationships. The knowledge that is made explicit must be inference-oriented.

The primary objective is then to design a mechanism to perform a variety of inference operations and particularly the sequences of inferences that lead us to fulfill the desired understanding task. As introduced in the previous chapter, **OIR** graphs serve as a well-defined representation in the context of map understanding. It has

been known that an understanding process is one during which instances and relations are gradually recognized and made explicit in an **OIR** graph, until a complete **OIR** graph is reached. Since the **OIR** graph is an abstract structure, designing a reasoning mechanism for map understanding is actually designing such a mechanism that works on **OIR** graphs. To obtain actions taken for reasoning, we just need to focus on new instances and relations to be produced and under what kind of circumstances they are produced.

In this chapter, a method to transfer a Description Logics representation to a grammatical representation is proposed. The knowledge reasoning mechanism is built on the resulted grammatical representation instead of on the DL representation directly. Therefore, the map understanding process is treated as a grammar parsing process. The reasons to adopt such an approach are presented as follows. First, grammar has been successfully used in a lot of other applications, especially language processing applications. Second, a grammar parser can be automatically generated according to the given grammar. There exist a large number of grammar analysis methods that are readily available. Third, a DL representation can be converted to a grammar which preserves the semantics of the DL representation. Fourth, map understanding is a bottom-up process that can be characterized by a grammar parsing process, because grammar parsing is also a bottom-up process that discovers higher level symbols based on the currently recognized symbols.

7.1 Role Types

Augmentation rules in **Definition** 6.3 govern the understanding process in general terms. They are not enough to guide the understanding process, since they do not reveal domain specific relationships. In other words, they specify what can be added to the **OIR** when certain new instances and relations are discovered based on the current state of the **OIR**, but not how to explicate the facts. Take the role name $\underline{\texttt{Road\_Network\_Has}_R}$ as an example. In Section 5.2, axiom 4 states that the concept roadnetwork is defined as having a relationship with the concept joint_leg_pair. Such a relationship in Description Logic systems is formally treated as a subset of $\text{dom}(\mathcal{I}) \times \text{dom}(\mathcal{I})$. However, this subset is encoded in the raw map image, not in the **OIR**. Exactly what instance pairs make up this subset has to be revealed during the understanding process. This is a question that can only be answered with the domain knowledge. Since a joint_leg_pair is always a part of a road network, we know that for a joint_leg_pair instance, there is an edge connecting it with one and only one road_network instance. These actually are the constraints that the role $\underline{\texttt{Road\_Network\_Has}_R}$ should obey, which can be expressed as follows:

(**G1**) For all $a \in$ joint_leg_pair, there exists $b \in$ road_network such that $(a, b) \in$ ROAD_NETWORK_HAS.

(**G2**) For all $a \in$ joint_leg_pair and all $b \in$ road_network, $(a, b) \in$ ROAD_NETWORK_HAS and $(a, c) \in$ ROAD_NETWORK_HAS imply $b = c$.

(**G3**) Let a, $b \in$ joint_leg_pair, v, $w \in$ road_network, $(v, a) \in$ ROAD_NETWORK_HAS, $(w, b) \in$ ROAD_NETWORK_HAS, $(a, j_1) \in$ HAS_JOINT, $(a, l_1) \in$ HAS_LEG, $(b, j_2) \in$ HAS_JOINT, and $(b, l_2) \in$ HAS_LEG. If $j_1 = j_2$ or $l_1 = l_2$, then $v = w$.

These constraints mean that two joint_leg_pair instances are two parts that belong to the same road network if they have a common leg or a common joint. Thus in the **OIR** graph, they are connected to the same road_network instance. It is said that **a** is connected to **b** if there is an edge between **a** and **b**.

In general, each role name represents certain information and defines particular constraints that should hold among multiple concepts. Next we are going to summarize some typical role types used in map understanding and specify the constraints for each of them.

1. CONTAINS

   This is a *one to many* role. Just as shown in Figure 6.1, there is only one instance in instance holder "map", and all the instances in holder "roadnetwork_or_rivernetwork" should be connected with it. Let **A** and **B** be two concepts, let CONTAINS $\subseteq \mathbf{A} \times \mathbf{B}$ be such a role, and let a, b, c be instances. The following are the constraints CONTAINS should hold.

   - If $b \in \mathbf{B}$, then there exists a such that $a \in \mathbf{A}$.
   - If $b \in \mathbf{B}$ and $c \in \mathbf{B}$, then $a = b$.

2. COMPOSE_STRING_OF_LEG

This is *a sequence* constraint. Let $\mathbf{A} = \exists$COMPOSE_STRING_OF.$\mathbf{B}$ be an axiom. Then its constraint can be expressed as

- If $a \in \mathbf{A}$, then there exists a subset $\{b_1, b_2, \cdots, b_n\} \subseteq \mathbf{B}$ such that the following hold:

  $(a, b_1) \in$ COMPOSE_STRING_OF, $\cdots$, $(a, b_n) \in$ COMPOSE_STRING_OF;

  $h(b_1, b_2)$, $h(b_2, b_3)$, $\cdots$, $h(b_{n-1}, b_n)$, and there do not exist b_0, $b_{n+1} \in \mathbf{B}$ such that $h(b_0, b_1)$ and $h(b_n, b_{n+1})$.

  where h is a predicate denoting that b_i and b_{i+1} are two part_of_leg instances connecting to each other.

- If $b \in \mathbf{B}$, then there exists a such that $a \in \mathbf{A}$.

3. RIVER_NETWORK_HAS

   Let $\mathbf{A} = \exists$RIVER_NETWORK_HAS.$\mathbf{B}$ be an axiom.

   - For all $b \in \mathbf{A}$, there exists $a \in \mathbf{A}$, such that $(a, b) \in$ RIVER_NETWORK_HAS.
   - For all a : $\mathbf{A}$ and b : $\mathbf{B}$, $(b, a) \in$ RIVER_NETWORK_HAS and $(b, c) \in$ RIVER_NETWORK_HAS imply $a = c$.
   - Let $a, b \in \mathbf{A}$, $v, w \in \mathbf{B}$, $(a, v) \in$ RIVER_NETWORK_HAS, $(b, w) \in$ RIVER_NETWORK_HAS, $v \in$ Pre_Section_Joint(HAS_JOINT D_JOINT, HAS_SECTION D_SECTION) (i.e., $(v, j_v) \in$ HAS_JOINT, $(j_v, jv_d) \in$ D_JOINT, $(v, s_v) \in$ HAS_SECTION, $(s_v, js_d) \in$ D_SECTION), $w \in$ Pre_Section_Joint(HAS_JOINT D_JOINT, HAS_SECTION D_SECTION) (i.e.,

$(w, j_w) \in$ HAS_JOINT, $(j_w, jw_d) \in$ D_JOINT, $(w, s_w) \in$ HAS_SECTION, $(s_w, sw_d) \in$ D_SECTION), $j_v = j_w$ or $s_v = s_w$, then $a = b$.

Note that features D_JOINT and D_SECTION are subsets of dom($\mathcal{I}$) $\times$ dom($\mathcal{D}_{id}$), where dom($\mathcal{D}_{id}$) is a concrete domain of the unique ID numbers of abstract objects.

4. COMPLEX_JOINT_HAS

The role should satisfy the *circular* constraint. Let $\mathbf{A} = \exists$COMPLEX_JOINT_HAS.$\mathbf{B}$ be an axiom, $a \in \mathbf{A}$, and $b_1, \cdots b_n \in \mathbf{B}$. Then

$(a, b_1) \in$ COMPLEX_JOINT_HAS, $\cdots$, $(a, b_n) \in$ COMPLEX_JOINT_HAS, *iff* $b_1, \cdots, b_n$ are connected together in a circular form.

Let there exist

- $ep_1, \cdots, ep_n \in$ end_pair such that $(b_1, ep_1) \in$ HAS_END_PAIR, $\cdots$, $(b_n, ep_n) \in$ HAS_END_PAIR,
- $p_{1,1}, p_{1,2} \in$ position such that $(ep_1, p_{1,1}) \in$ POSITIONED_AT, and $(ep_1, p_{1,2}) \in$ POSITIONED_AT, $\cdots$, $p_{n,1}, p_{n,2} \in$ position such that $(ep_n, p_{n,1}) \in$ POSITIONED_AT, $(ep_n, p_{n,2}) \in$ POSITIONED_AT,

then $p_{i,2} = p_{i+1,1}$, $i = 1, \cdots, n$, and $p_{n,2} = p_{1,1}$.

5. COMPOSE_STRING_OF_RSECTION

This role also specifies a *sequence* constraint. The only difference between COMPOSE_STRING_OF_RSECTION and COMPOSE_STRING_OF_LEG is that their

definitions of predicate h are different. In the constraint definition of COMPOSE_STRING_OF_RSECTION, h decides whether two part_of_complex_section instances are connected.

In a general sense, features are considered special roles. For each feature name f in $\mathcal{ALC}(\mathcal{D})$, its constraint can be derived from its definition:

If $(x, y) \in f^{\mathcal{I}}$ and $(x, z) \in f^{\mathcal{I}}$, then $y = z$.

7.2 Specification of Grammar

In the following, an approach to convert Description Logics semantics to a set of grammar productions will be shown in detail. It has been mentioned that the understanding process is the evolving of an initial **OIR** graph into a complete **OIR** graph, and that the augmentation rules give the actions that incrementally build the complete **OIR** graph. To implement the understanding system, a control mechanism is needed to interweave and collaborate the understanding steps. The idea is to derive a set of grammar productions from a Description Logics representation. Thus the map understanding process can be carried out based on the grammar rules, and a parser will be automatically generated. This parser serves as such a control mechanism.

The grammar derivation takes two phases. The first phase is to map T[S], a T-box with respect to the task S, into a set of grammar rules, while the second phase enhances and refines the grammar production set obtained from the first phase.

Each element of the production set $\widetilde{P} = \{P_i \mid i = 1, 2, \cdots, m\}$ has the form

$$P_i ::= S_{i,1} \quad S_{i,2} \quad \ldots \quad S_{i,n_i}$$

The left hand side P_i represents a piece of knowledge such as $a : C$ in $\mathcal{A}$. The right hand side represents a list of symbols, each of which also represents a piece of knowledge.

The author proposes to consider first how each single terminological axiom corresponds to a set of productions. Then the corresponding production sets of all axioms in the T-box are put together to obtain the so called map understanding grammar. Recall that in **Definition** 6.3 a set of augmentation rules is given. These augmentation rules state general semantics of terminological axioms that are used in the understanding process. They therefore are the general reasoning rules that the map understanding process should follow. Additionally, the particular semantics of various instances and their relations should be taken into consideration in the formation of the reasoning mechanism as well. In the previous chapter, we studied the semantics of various role types and features and brought about the constraints of each role in the understanding process. Since an understanding step takes actions to explore and reveal "new" instances and relations based on the existing instances and relations, the general augmentation rules together with the role-specific constraints can serve as reasoning rules for the understanding process. However, although these reasoning rules are sufficient to build a reasoning algorithm, they are not expressed in a compact and unified form. Users have to provide a knowledge inference mechanism to guide the process to select and apply them. The inference mechanism has the rules work

in concert to solve problems. By mapping a knowledge representation to a grammar, the automatically generated grammar parser can serve as a reasoning control module. Knowledge control, collaboration, subgoal decision, and optimization can be incorporated in the parser.

Before the definition of the derivation rules is given, an equivalent rewriting of a T-box $\mathcal{T}$ is introduced.

Definition 7.1 *(Well-formed T-box)*

Not losing generality, assume that all concept terms are in negation normal form.

- *For those non-atomic concept terms in axioms of the form* $A = B_1 \sqcup \cdots \sqcup B_k$, *say,* $B_{i_1}, \cdots, B_{i_q}$, *substitute them with newly created atomic concept terms* $B^*_{i_1}$, $\cdots$, $B^*_{i_1}$ *respectively, and add to* $\mathcal{T}$ *the following new axioms:* $B^*_{i_1} = B_{i_1}, \cdots, B^*_{i_q} = B_{i_q}$.

- *Consider those concept terms in the right hand side of axioms of the form* $A = B_1 \sqcap \cdots \sqcap B_k$. *If* B_i, $1 < i < k$, *is not atomic and appears in the form* $B_{i_1} \sqcap \cdots \sqcap B_{i_w}$, *substitute* B_i *with an atomic concept term* B^*_i *and add to* $\mathcal{T}$ *an axiom* $B^*_i = B_i$. *If* B_i, $1 < i < k$, *appears in the form* $\exists R.B_{i'}$ *or* $\forall R.B_{i'}$, *and* $B_{i'}$ *is not an atomic concept term, substitute* $B_{i'}$ *with newly created atomic concept term* $B^*_{i'}$, *and add to* $\mathcal{T}$ *a new axiom* $B^*_{i'} = B_{i'}$.

- *For the non-atomic concept term* B *in axioms of the form* $A = \exists R.B$ *or* $A = \forall R.B$, *substitute* B *with newly created atomic concept term* B^*, *and add* $B^* =$

B to $\mathcal{T}$.

These rewriting procedures are applied repeatedly until the T-box no longer changes. The resulting T-box is called a well-formed T-box.

A well-formed T-box assures that in an axiom, any role related restriction ($\exists R.C$ or $\forall R.C$) is imposed on an atomic concept term; that is, C is required to be atomic. Based on a well-formed T-box with respect to task S, the derivation rules can be defined.

Definition 7.2 *(Derivation Rules)*

For each terminological axiom **a**, *a set* $\tilde{\mathbf{P}}$ *of productions can be derived.*

The derived grammar $\mathcal{G}_T$ *is defined as follows:*

- *If* **a** *is in the form* $A = B_1 \sqcup \cdots \sqcup B_k$, *a set of production rules are added into* $\tilde{\mathbf{P}}$: $\tilde{A} ::= \tilde{B}_1, \cdots,$ *and* $\tilde{A} ::= \tilde{B}_k$, *where* $\tilde{A}, \tilde{B}_1, \cdots, \tilde{B}_k$ *are grammar symbols corresponding to* $A, B_1, \cdots, B_k$.

- *If* **a** *is in the form* $A = B_1 \sqcap \cdots \sqcap B_k$, *and* $B_1, \cdots, B_k$ *are atomic concept terms, a production rule* $\tilde{A} ::= \tilde{B}_1 \cdots \tilde{B}_k$ *is created and added into* $\tilde{\mathbf{P}}$, *where* $\tilde{A}$, $\tilde{B}_1, \cdots, \tilde{B}_k$ *are grammar symbols corresponding to* $A, B_1, \cdots B_k$.

Each of the production rules constructed in this way describes a single understanding step that examines the current state of the interpretation process (**OIR** graph), and tries to discover certain pieces of knowledge (facts on instance properties

and existence of relations). For example, a production rule $A^* ::= B^*\ C^*$, which is derived from axiom $A = B \sqcap C$, can indicate that if $b : B^{\mathcal{I}}$ and c: $C^{\mathcal{I}}$, then put a new fact $a : A^{\mathcal{I}}$ in the **OIR**.

In the next section, we proceed to the second phase of production rule derivation.

7.3 Symbols Representing Constraints

It is known that roles in DL systems can be treated as constraints on the product $\mathrm{dom}(\mathcal{I}) \times \cdots \times \mathrm{dom}(\mathcal{I})$. One important task is to use production rules to specify those constraints. The grammatical representation has to describe not only information about object instances, but also relative spatial constraints over them. It is necessary to introduce some special grammar symbols to deal with the relative constraints.

The grammatical restriction symbols are now defined more precisely. First, some related terms have to be defined.

Definition 7.3 *(membership, property and constraint symbols)*

Three types of symbols used in building production rules are distinguished. The first one is the ***membership symbol*** *(m-symbol), which represents the knowledge of a certain instance's membership. The second one is the* ***property symbol*** *(p-symbol), which represents one of the properties of another instance. A p-symbol itself may be an instance of either an abstract or a concrete domain. Membership and property symbols together are called* ***existential symbols****. The third one is the* ***constraint***

symbol *(c-symbol), which specifies the knowledge of a single relation instance among two or more individuals.*

For example, a membership symbol can stand for the fact specified by an assertional axiom $a : C$. Constraint symbols can be used to represent facts, such as $(a, b) : R$, since the role R can be specified as a set of restrictions on $\text{dom}(\mathcal{I}) \times \cdots \times \text{dom}(\mathcal{I})$. If such a constraint symbol occurs in a production rule, the interpretation action taken is usually associated with an algorithm to examine whether (a, b) is a relation instance of R. Therefore, we can also use a predicate logical formula $P_R(a, b)$ to represent the constraints. That is, $P_R(a, b)$ is an alternative representation of the semantics of this constraint symbol.

Since special symbols are taken into consideration in the construction of grammatical rules used in map understanding, it is obvious there are some syntactical differences between the map grammar and the traditional ones. Based on the above discussion, the definition of a grammar for map understanding can be given.

Definition 7.4 *(Map grammar)*

Given $n \geqslant 1$, a map grammar is a collection of production rules in the form

$$A ::= \alpha_1 \, \dot{r}_1 \, \alpha_2 \, \dot{r}_2 \cdots \alpha_n \, \dot{r}_n$$

where the left hand side is a single membership symbol, and the right hand side is a string of one or more symbols, each of which is either an existential or a constraint symbol. In particular, $\alpha_1, \cdots, \alpha_n$ represent continuous strings of existential symbols,

which we refer to as ***existential symbol chunks****.* $\dot{r}_1, \cdots \dot{r}_n$ *are strings of one or more constraint symbols, referred to as* ***constraint symbol chunks****. We also define the membership chunk* (α_p) *appearing immediately before a certain constraint symbol* $\dot{r}_p$ *as its* ***restriction scope****. In other words, each c-symbol in* $\dot{r}_p$ *only specifies constraints among membership instances in its participant set. It does not specify any restrictions over m-symbols or p-symbols at its right hand side.*

An example of a simple map grammar production rule is given below.

$$\text{Q} :== \underbrace{\text{A}\ \ \text{B}\ \ \text{C}}_{\alpha_1}\ \ \underbrace{\dot{R}_1\ \ \dot{R}_2}_{\dot{r}_1}\ \ \underbrace{\text{D}\ \ \text{E}}_{\alpha_2}\ \ \underbrace{\dot{R}_3}_{\dot{r}_2}$$

where Q at the left hand side is a membership symbol. A, B, C, D, and E are also m-symbols. $\dot{R}_1$, $\dot{R}_2$ and $\dot{R}_3$ are constraint symbols. $\dot{R}_1$ and $\dot{R}_2$'s restriction scope includes A, B and C, while $\dot{R}_3$'s restriction scope is made of D and E.

The symbols Q, A, B, C, D, and E may have been derived from assertional axioms q : Q, a : A, b : B, c : C, d : D, and e : E. What is important is the semantics of constraints $\dot{R}_1$, $\dot{R}_2$ and $\dot{R}_3$. The constraint predicate symbolized by $\dot{R}_1$ may be applied on both A and B, or on A alone. C is included in $\dot{R}_1$'s restriction scope, but $\dot{R}_1$ does not specify any restrictions over it. $\dot{R}_2$ can impose restrictions on all three of A, B and C. $\dot{R}_1$ can be used to specify facts like $(a, b) : R_1$.

The presence of a constraint symbol $\dot{r}_m$ in a production rule indicates that after a sequence of symbols $\alpha_1\ \dot{r}_1\ \alpha_2\ \dot{r}_2 \cdots \alpha_m$ are *shifted in* during the parsing process, a situation is met such that we have to determine whether $\dot{r}_m$ will impose certain

restrictions on some of the precedent symbols already shifted, that is, restrictions on symbols in $\dot{r}_m$'s scope. The actions that deal with this situation and examine whether certain conditions hold are abstracted into a symbol $\dot{r}_m$. The problem then becomes to determine whether the next token to shift in is an instance of $\dot{r}_m$. To decide whether a $\dot{r}_m$ token exists and what kind of restrictions to impose on symbols in its scope, a so called restriction evaluation function associated with $\dot{r}_m$ is evaluated. Suppose that the result of the evaluation is a true/false value, which decides whether there exists a constraint symbol corresponding to the evaluation function. Such an evaluation function is obtained according to the semantics implied in the constraint symbol. Consider, for example, a constraint symbol, attaching, which has two m-symbols RS_1 and RS_2 in its scope. RS_1 and RS_2 indicate the existence of two "road section" instances. The evaluation function associated with attaching is then used to decide whether RS_1 and RS_2 are attached to each other. Therefore, the constraint is discovered by the evaluation function.

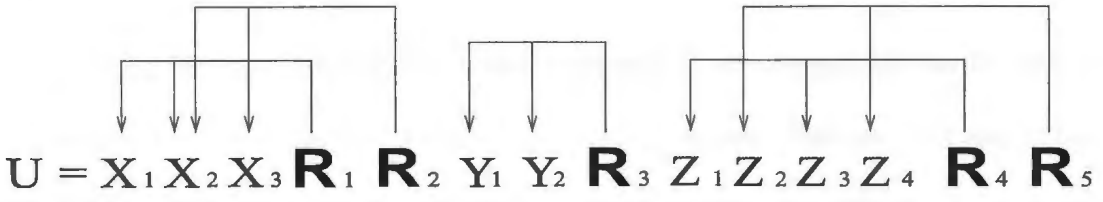


Figure 7.1: Constraint symbols and their scopes.

Consider Figure 7.1. It shows a production rule with 9 membership symbols and 5 constraint symbols. Also shown are the m-symbols on which the c-symbols impose restrictions. Note that it is not necessary for each c-symbol to enforce constraints

over all m-symbols in its scope.

Since an understanding step is not expected to take on excessive semantics other than specifying facts about the membership of a few individuals and the relations that they participate, typically the production rule shown in Figure 7.1 may also be presented as follows:

U(u : U) ::= X_1[b : X_1] X_2[c : X_2] X_3[d : X_3] R_1[b, c] R_2[b, c, d] Y_1[b : Y_1] Y_2[e : Y_2] R_3[b, e] Z_1[b : Z_1] Z_2[b : Z_2] Z_3[f : Z_3] Z_4[g : Z_4] R_4[b, f] R_5[b, g]

The semantics of the above production can be given as:

Given individuals b, c, d, e, f, and g, concepts X_1, X_2, X_3, Y_1, Y_2, Z_1, Z_2, Z_3, and Z_4, and roles R_1, R_2, R_3, R_4, and R_5,

$b \in X_1^{\mathcal{I}} \cap c \in X_2^{\mathcal{I}} \cap d \in X_3^{\mathcal{I}} \cap (b, c) \in R_1^{\mathcal{I}} \cap (b, c, d) \in R_2^{\mathcal{I}} \cap b \in Y_1^{\mathcal{I}} \cap e \in Y_2^{\mathcal{I}} \cap (b, e) \in R_3^{\mathcal{I}} \cap b \in Z_1^{\mathcal{I}} \cap b \in Z_2^{\mathcal{I}} \cap f \in Z_3^{\mathcal{I}} \cap g \in Z_4^{\mathcal{I}} \cap (b, f) \in R_4^{\mathcal{I}} \cap (b, g) \in R_5^{\mathcal{I}} \Longrightarrow u \in U.$

Usually, to decide whether the token representing an m-symbol t can be shifted in, we just need to determine whether $typeof(t, Y_i)$ is true. However, if a restriction symbol has derived a restriction R_i on it, to decide whether to shift it in, we have to evaluate $typeof(t, Y_i) \wedge R_i(Y_i)$.

For example, consider the c-symbol p_form_bridge[l,m] in the production below:

BRIG[b:brig] ::=

T_LNS[l:lns] p_is_barline[l] T_LNS[m:lns] p_is_barline[m] p_form_bridge[l,m]

It represents whether there exists a second bar line that forms a bridge together with the first bar line token already shifted in. If there exists such a bar line, it will impose a restriction on the two "bar line" symbols | and m.

The following additional rules are followed to deal with the translation of terminological axioms into a set of production rules. According to the derivation rules in **Definition** 7.2, an axiom which is a disjunction of several concept terms corresponds to a set of production rules, each of which deals with a disjunctive term. Therefore, we just need to consider those axioms in the form $A = B_1 \sqcap \cdots \sqcap B_q$, where B_i is either an atomic concept term or an **atomic restrictive concept term**, which refers to a concept term in the form $\exists$ R . $C_1 \otimes \cdots \otimes C_m$ or $\forall$ R . $C_1 \otimes \cdots \otimes C_m$, where $C_1, \cdots$, and C_m in turn are atomic concept terms (referred to as **participating concepts**).

Before further conversion, we have to rearrange the order of the right hand side of an axiom A, if necessary. Let B_i and B_j, $1 \leqslant i, j \leqslant q$, be two concept terms that occur in A, where B_i is an atomic concept term, and B_j is an atomic restrictive concept term. If B_i is at a place after B_j, it will be removed from its current place and put in a place before B_j. For example, $A = B \sqcap \exists$R.$C \sqcap D$ will become $A = B \sqcap D \sqcap \exists$R.$C$. The idea behind this is to make sure those facts about the existence of instances are examined first, then restrictions imposed on them can be checked. Obviously such permutations will not change the axiom's semantics.

Before moving forward, two types of roles in Description Logics have to be distin-

guished.

Definition 7.5 *(Roles with fixed or varied number of participants)*

***f-roles** are the type of roles with fixed number of participants, while **v-roles** are those roles with variable number of participants.*

In this study, we will not consider roles without an upper bound on the number of participating instances, because any map is a collection of a finite number of map symbols.

Let us first consider those atomic concept terms. In the corresponding production rule, each of them will appear as a symbol like SYM[a : S].

Let us consider each restrictive concept term $B_t = \exists$ (or $\forall$) $R.C_1 \otimes \cdots \otimes C_m$ and its preceding m-symbol chunk. If R is an f-role, we will map each restricted atomic concept term C in the scope to a symbol CSYM[$c : C$]. If any of $C_1, \cdots, C_m$, say C_q, is not in the scope, we will add a new symbol (like CSYM$_q$[$c : C_q$]) to the rule. For B_t itself, we will use a symbol $R[a_1, \cdots, a_p]$ as its representation in the rule, where a_i, $1 \leqslant i \leqslant p$ corresponds to an m-symbol ZSYM[a_i : Z] in front of $R[a_1, \cdots, a_p]$.

For terms with v-roles, we will resort to the derivation rules discussed in the next section.

7.4 Refinement of the Derived Grammar Rules

The previous section deals with the syntax level derivation of production rules from terminological axioms. To obtain a full-fledged grammar that is appropriate for a grammatical analysis process, semantic level derivations have to be conducted as well. Aggregation roles, which are an important class of roles in map understanding, are discussed.

A typical example of an aggregation-entity relation is role $\underline{\texttt{Contains}_R}$. Recall the terminological axiom 1 presented in Section 5.2 of Chapter 5, which states that each instance in the concept map corresponds to a set of a varying number of instances of concept roadnetwork_or_rivernetwork. Two production rules can be derived according to the semantics of the concepts and the roles involved:

P1 MAP[a : map] ::= PartialMAP[b : partial_map] Pre_no_more[b]

P2 PartialMap[a : partial_map] ::= PartialMap[b : partial_map] RNRN[c : roadnetwork_or_rivernetwork]

Note that an intermediary m-symbol "PartialMap" with a corresponding concept partial_map is introduced. A partial_map instance refers to a partially interpreted map (i.e., the collection of roadnetwork_or_rivernetwork instances already shown in the **OIR**). Rule **P1** states that a map instance is built with a collection of road or river networks to which no more instances of concept roadnetwork_or_rivernetwork can be added. The symbol "Pre_no_more[b]" stands for a predicate indicating the fact that

there exists no individual in dom($\mathcal{I}$) that can be added into b to form a new collection. This is an example of symbolizing an action to check the true/false value of a fact.

Also note that recursive definitions are used in these rules. However, we will limit the use of recursive treatment to terminological axioms with aggregation roles. Since the aggregation roles are applied on concepts with a finite number of elements, there exists an upper bound on recursive cycles.

7.5 Implementation of the Map Grammar Parser

For the standard form of a parsing procedure [37] of a programming language, the scanner first takes in a source program and generates a string of tokens, then the parser inputs the tokens one by one and produces the parse tree. In some respects, the proposed syntax-based map interpretation system, a map parser, is similar to a classic parser. The input of the map parser is not a program, but a map; that is, map data obtained from low-level image processing procedures. Given the complex nature of map information, we need to construct a parsing/interpretation system to address issues encountered during a map interpretation process. Figure 7.2 gives an overview of the proposed interpretation system.

Just like a language parsing system, the map interpretation system has a parser generator as well. The parser generator, or grammatical analyzer, takes in a context-free map interpretation grammar as input and generates the parser. The map interpretation grammar consists of the productions derived from a DL representation

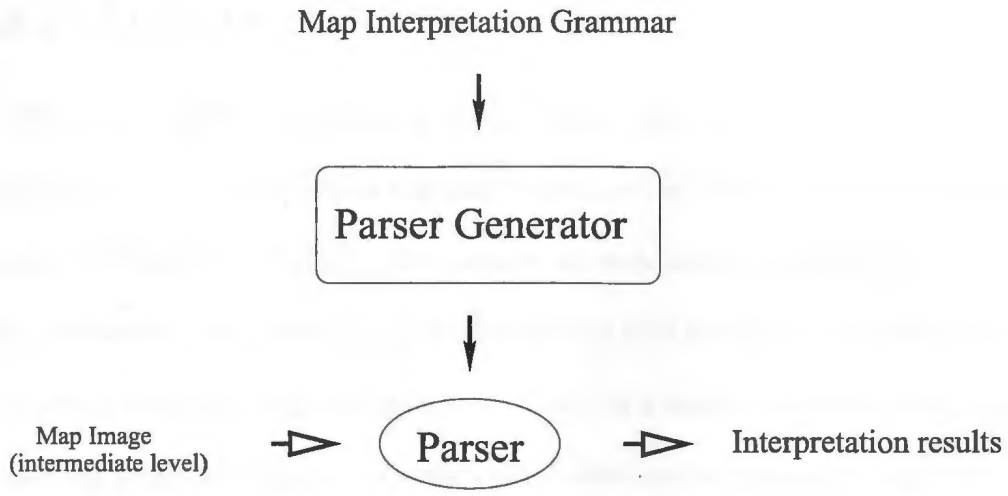


Figure 7.2: An overview of a parser system.

using the method described in Section 6.3. The input to the parser is the map image to be interpreted. The map image is represented by a collection of map elements, such as line segments and city symbols, each of which is in raster or vector format. The output of the parser is the interpretation result.

In this section, a unique parsing mechanism for the map grammar is introduced. The map grammar parser can be viewed as a generalization of the standard LR(k) parser because they share the same basic concepts of shift and reduce. Since map grammars have grammatical features (such as constraint symbols) different from those of standard context-free grammars, their parsing algorithms call for special data structures and control structures. The standard LR(1) parsing mechanism will be briefly reviewed first, then the unique concepts involved in a map grammar parser are introduced, followed by a discussion of the **multiple path stack (MPS)**, which is the basic structure for holding parsing states and helping with the control mechanism.

7.5.1 LR(1) Parser

A shift-reduce parser of a grammar puts its main concern on determining when the right hand side of a production rule can be replaced by its left hand side symbol. It works in a bottom-up fashion. The process of map understanding fits naturally in this mechanism. Comparably, map understanding also works in a bottom-up fashion by starting from a set of primitive objects. Just like a traditional parser using a series of parsing actions to recognize a sentence, its inference mechanism of exploring facts and relations and discovering new instances and relations in an **OIR** can be viewed as a shift-reduce parsing process.

In the following, the mechanism of the shift-reduce parser will be reviewed first. It has an initially empty parser stack, which will contain symbols already parsed. At the beginning, the input queue has a string of all the lexical elements (called tokens) of a sentence. The following actions are repeatedly taken: shift and reduce. The basic idea behind this is to shift tokens from input onto the parse stack until the top of the stack matches a right-hand side (handle) of a production. Then the handle is reduced by substituting it with the left-hand side symbol of the production. The process will end with either the goal state reached or syntax errors reported.

To find out the parsing actions, two tables have to be constructed based on the grammar: ACTION and GOTO. The ACTION table is exploited to decide the next action to take. The GOTO table tells which parse state we reach after a token is shifted in. It is also necessary to look up the next parse state from the GOTO table

when a reduce action takes place and a left-hand side nonterminal is put on the parse stack. At first, the initial state S_0 is pushed onto the empty parse stack. Then the parsing process repeats the following operations: (1) Obtain S, the state at the top of parse stack (the current parse state, usually an integer), and T, the current input token, use S and T to index into the ACTION table to get the next action ACTION(S, T); (2) If the next action is shift, push GOTO(S, T) into the parse stack; and (3) If the next action is reduce, pop up those states on top of the stack that correspond to the right hand side of the matching production, and then use the resulting parse stack top, $\tilde{S}$, and the left hand side of the matching production, A, to index into the GOTO table to obtain the new parse state and push it into the stack. The ACTION table also contains a special "Success" value and a number of "Error" values. If "Success" is met, the parse process finishes successfully. If "Error", the parsing has to be aborted due to a syntax error. If a different language has to be parsed, we only need to reconstruct the ACTION and GOTO tables without changing the parser algorithm itself.

7.5.2 Characteristics of the Map Grammar Parser

Unlike the traditional parsers [37], the input to a map parser is not an ordered list of elements, but an unordered set of map primitives. In classical parsing, there is always a "next" element waiting in the input stream. It will be shifted in to the parse stack after the "current" element has been processed. Since map primitives do not have a

linear structure, there is not a readily-prepared next primitive from the input. Instead, the input to the parser is an unstructured collection of map primitives. However, it is inevitable that we assign an appropriate order in which those in the primitive collection should be fed into the interpretation system. Fortunately this problem can be solved by taking advantage of the formal grammatical representation. It is known that for a context-free grammar there exists a finite state machine characterizing the transitions of parse states. For each such parse state, there is a set of input symbols through which other parse states can be reached. Such a set contains the possible symbols that can be chosen as the "next" symbols.

Another characteristic of the map parser is its ability to deal with restriction function symbols. In the standard parsing of languages, the existence of a symbol, or token, does not depend on the parse state. In the map grammar, restriction function symbols indicate that in one production, the symbols recognized earlier may impose constraints on the following symbols to be recognized; that is, the existence of certain symbols is dependent on the parsing context. Therefore, when the system has to decide what kind of symbol to feed into the parser, it has to make sure the symbol satisfies the constraints.

One more characteristic of our parser to note is that it allows for ambiguities in the grammar. Ambiguities indicate that there exists more than one interpretation for certain parts of a map. There is no necessity to eliminate the ambiguities. Typical parsers usually work in a deterministic style. Parsing actions like "shift" and "reduce"

have to be uniquely decided at each step. Tracking back to previous parsing states is not allowed. However, the map grammar parsing mechanism allows such back tracking. The next section will present the data structure used for back tracking.

7.5.3 Multiple Path Stack (MPS)

The **multiple path stack (MPS)** is a central concept in map grammar parsing. It is the device that we use to handle and organize pivotal actions, such as candidate preparation and selection, shifting, reducing, and tracking back.

An MPS is an acyclic directed graph with a root node. At the very beginning of a map grammar parsing process, the graph does not contain any nodes other than the root. With the advance of the map understanding process, an MPS will grow into a tree like structure with multiple nodes.

Two different nodes occur in an MPS: **parse state nodes** (p-node) and **candidate nodes** (c-node). The information stored at each p-node can be regarded as a summary of the context of the "current" parsing state (or map understanding state). Since a map understanding process is also reflected in the process to build a complete **OIR** from its initial state, the parse state represented by each p-node is also a summary of facts of map instances and their relations marked as explicitly recognized. In addition, a list of c-nodes called *candidate node list* is attached to each p-node.

Candidate Node List

The p-nodes are connected by **path arrows**. Except for the root node, each p-

node in an MPS has one and only one incoming path arrow from another p-node (its parent) and a set of outgoing path arrows to other p-nodes (its children). Starting from a root p-node, the MPS branches out by adding new p-nodes connected with path arrows. Another type of arrow, which points from a p-node to its attached candidate node list, is called a **candidate arrow**. Each p-node in an MPS can have more than one **path** arrow and one and only one **candidate arrow**. It serves as a parse state, which indicates a state of an ongoing map understanding process (**OIR** building process). Such a state can also be treated as a representation encoded in the partially recognized map phenomena (instances and relations). Figure 7.3 illustrates a p-node N and its incoming and outgoing arrows, where path arrows are drawn with concrete lines, and candidate arrows with dotted lines. A, B_1, $\cdots$, B_x are p-nodes, and CN is the candidate node list of N.

The children of a p-node refer to its "next" parse states. Just like in a standard parsing algorithm, a next state is a state reached when a shift or a reduce action is taken based on the "current" parse state. Note that unlike the parse stack of the traditional parsers, where a "next" parse state is decided uniquely, the map grammar parser works in a non-deterministic manner.

The candidate arrows are a unique feature of the MPS. Each of them points to a list of candidate nodes. In Figure 7.3, the candidate node list CN is represented by an array of boxes. While in the candidate list, a candidate node does not represent any parse state. Later on, we will talk about how to remove a c-node from a candidate

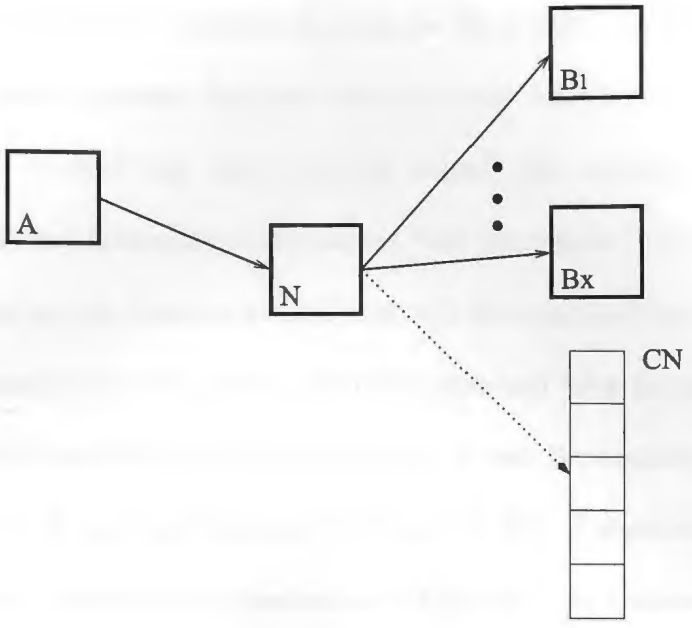


Figure 7.3: An MPS node N and its parent and children.

list and make it a p-node. A candidate node is associated with a type attribute. The type attribute defines the type of a terminal symbol.

Obtaining the Candidate List

The candidate node list associated with a p-node is obtained through the help of the ACTION table. Read across the row corresponding to the parse state represented by the p-node and select the occupied table cells first, then find the column headings corresponding to these occupied cells. The candidate list is made of the terminal types represented by the selected column headings. Take the action table shown in Table 7.1 on page 187 as an example. A p-node with parse state 1 has a candidate list (full_prs (T5), T_SRS (T6), T_CITY (T12), T_LNS (T14)). This tells that when a "single road segment" instance is shifted in (parse state 1 is reached), terminals of

the four types listed in this candidate list may be the possible lookahead symbols.

During the parsing process, the candidate list of each MPS node is further divided into two lists: a "visited" list and a "not yet visited" list. Initially, the visited list is empty. All the candidate nodes are marked "not yet visited". At a certain parse state, the nodes in the "not yet visited" list will be examined one by one to see whether an instance from the input collection (explained later in the next section) matches one of the candidate nodes. If no match is found, the c-node is removed from the "not yet visited" list and placed in the "visited" list. If a matching instance is found, the current c-node under examination is filled with the instance and becomes the current lookahead token. If this c-node is shifted in, it is converted to a p-node. It will remain at the top of the "not yet visited" list. The parser will stop examining, at least temporarily, the rest of the "not yet visited" list at this point, because the parsing has reached a new parse state. The parser considers this new parse state and tries to repeat the process. In this way the parser constantly pushes the parsing process by expanding the MPS. However, with the progress of the parsing procedure, it may come to a point that a back tracking action has to be triggered. That is, the parser has to go back to the previous active top in the history and try alternative paths other than those taken before (Refer to Section 7.7 for an example on how the back tracking operation works and how the previous active top is determined).

Input Selector

Similar to the standard parser, a map grammar parser employs a device called

input selector to feed the input tokens one by one into the parser. The function of the input selector is similar to the scanner of a traditional parser. The difference is that a standard scanner does nothing more than fetch tokens from an ordered string, whereas the input selector of the map parser does not work on an ordered string of tokens. The input selector of the map parser sifts through a collection of primitive map instances and the collection of already recognized instances and relations (**OIR**). One by one it picks up the appropriate object or relation instance as the "next" token. The **input collection** is made up of two parts: the **OIR** and the **primitive collection**. The primitive collection contains those primitive map objects extracted from raw images using low-level image processing algorithms and systems. Line segments, route numbers, cities, and bridges are examples of instances in an input collection.

A map grammar parsing process is characterized by a series of parsing steps. Each parse step pushes the process from one parsing state to another. Based on a particular parse state, the parser has to decide what type of tokens to look for. Then it instructs the input selector to produce a token of this type for it. As discussed earlier, in a map grammar we consider not only existential facts (m-symbols, p-symbols), but also relational facts (c-symbols). For example, if the input selector has been instructed to find an m-symbol, it can search through either the input collection or the **OIR** to retrieve the symbol. If a c-symbol is requested to be discovered by the selector, the constraint represented by the c-symbol may already have been explicitly encoded

in the **OIR**, or a special fact checking algorithm is invoked by the selector. A fact checking algorithm is used to check whether certain constraints are satisfied by certain c-symbols.

When the map size is small, the input selector can search through the whole set of primitive collection to pick the next token. However, when dealing with large size maps, it is not possible for the selector to cover all the primitives in the collection. The performance of the system will degrade dramatically when the map size increases. One solution is to search only the neighboring areas of the current active top of the MPS (discussed in the next section). This will produce a much smaller candidate list each time a next token is needed. A quadtree data structure can be used to represent the primitive collection. Since the primitive collection can be viewed as a representation of the map, it can be recursively decomposed into four equal area blocks until the leaf area blocks are small enough. Given a primitive, it is straightforward to retrieve the neighboring map objects of the primitive by locating its siblings in the quadtree representation. As a result, the input selector's performance can be greatly improved.

Active Top of MPS

Unlike the stack structure that expands or shrinks in one direction, the MPS is a tree structure that grows or shrinks along multiple paths. Therefore, it may have multiple stack tops, which are p-nodes that do not have child p-nodes. Among the stack tops, there is one that is called the **active stack top**. The active stack top indicates the current parse state. Similar to the traditional parser, the parse state

represented by the active top and the next input token are used to determine the parse action to take. At any given moment, there is only one active top. Note that the parent of the active top is not necessary its previous active top. Initially, only the root p-node is present in the MPS and the root is marked as the current active top.

Suppose that the current active top T and the lookahead symbol L suggest a shift action, which requires a new parse state be arrived and a new p-node Q be introduced. Q is "pushed" into the MPS. In other words, Q becomes a child of T and the current active top. T is no longer the active top, but the previous active top. A path arrow is drawn from T to Q. In the mean time, another type of arrow, called a **back tracking** arrow, is drawn from Q to T. Such an arrow points to the previous active top each time a new active top comes into existence. Therefore, a mechanism to keep record of the transitions of active tops is established.

Back Tracking

When a new parse state, whose corresponding MPS node is labeled as the active top, is reached and its candidate list obtained, the parser needs to pick a terminal from the primitive collection as the current lookahead token. It is possible that no candidate lookahead tokens can be found; that is, no currently available terminals have a matching type in the candidate list. In standard parsing, this usually means an error. However, in the context of map parsing, this does not necessarily mean an error, but that a **back tracking** action has to be triggered. Such a back tracking action will mark the active top as a "visited" one, which means that no further parsing

based on this parse state is needed. The control is given back to the previous active top. That is, it becomes the active top again. Algorithm 7.1 shows the pseudo code for back tracking.

Algorithm 7.1 ***(Backtracking Function)***

procedure *regress_to_previous_active_top()* ***is***

begin

If the current active top is root, indicate error.

If there does not exist a previous active top, indicate error.

Put the current active top into the "visited" list of its parent.

Set the previous active top as the current active top.

end

Reduce Action

Let us consider the behavior of the MPS when the input token currently under examination suggests a reduce action. Figure 7.4 illustrates the part of an MPS affected when a reduce action is needed. The state of the MPS before the reduce action is carried out is contained inside the dashed lines. N_0, N_1, $\cdots$, and N_k are a string of p-nodes connected by path arrows, which indicates the MPS has progressed through a series of parse states. Note that each of the p-nodes should have an attached candidate node list. To save space, only N_k's is shown.

Assume that at this point N_k is the current active top of the MPS. **V** and **I** together compose the candidate node list associated with N_k, where **V** is the list that

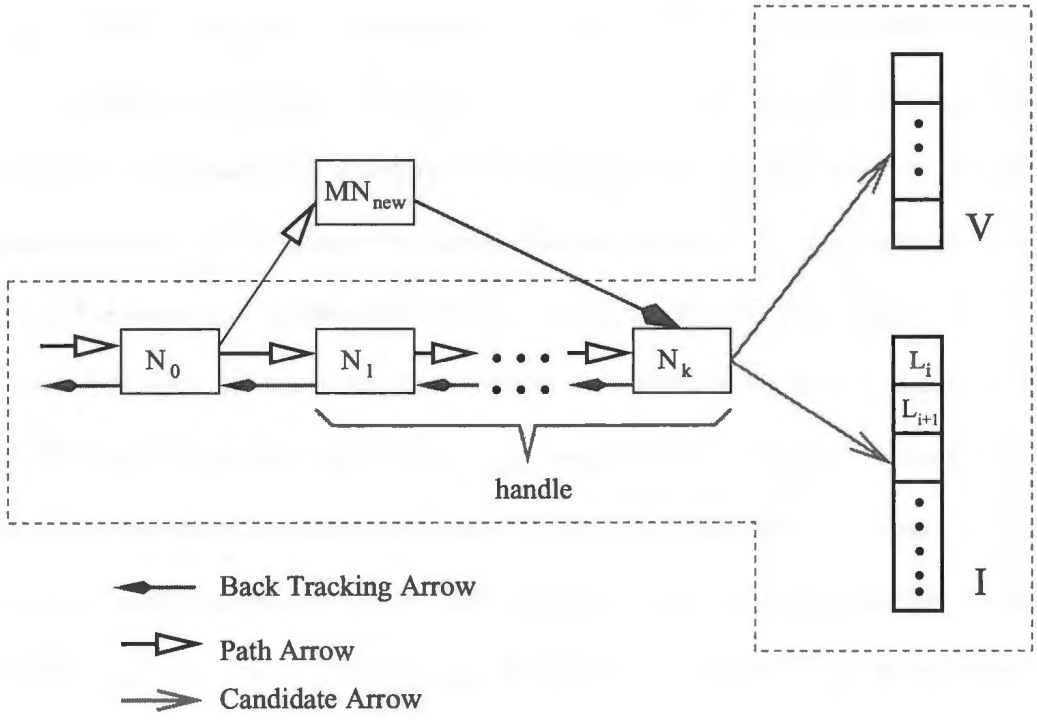


Figure 7.4: Part of the MPS involved in a reduce action.

holds the c-nodes marked as "visited", and **I** holds all those not yet visited. Let L_i be the c-node holding the current lookahead symbol. The next action is determined by looking up the ACTION table using N_k and L_i.

Suppose a reduce action is suggested and the production to reduce is decided. The map parser will behave much differently from the standard parser in the way it deals with a handle (the string of shifted-in symbols on the top of the parse stack that matches the right hand side of a production). Rather than popping the handle from the parse stack, the map parser will keep those MPS nodes corresponding to the handle in the MPS. The reason is that these MPS nodes still hold important history information. Later on we will see that the parser may need to track back to N_k again.

In an MPS, the handle corresponds to a series of MPS nodes (p-nodes) chained with path arrows. In Figure 7.4, nodes $N_1, \cdots, N_k$ are such a node chain referred to as a handle. It is obtained by following in the reverse direction of path arrows from the current active top. To achieve the same effect as popping up the handle, the parser locates the immediate ancestor of the node chain, which is N_0 from Figure 7.4. Then a new MPS node MN_{new} that represents the left hand side symbol is created. The new MPS node represents a new parse state reached after the reduce action is carried out. The new parse state is given by GOTO(S(N_0), NT(MN_{new})), where S(N_0) is the parse state represented by N_0, and NT(MN_{new}) is the parse state by MN_{new}. N_0 will treat the newly added MN_{new} as one of its children. This is indicated in Figure 7.4 through a path arrow pointing from N_0 to MN_{new}. Moreover, MN_{new} is put immediately before N_1, which was the top item of N_0's "not yet visited" candidate list. Therefore, MN_{new} becomes the current top item of N_0's "not yet visited" candidate list. Next, it is made the current active top, indicating that it represents the latest parser state. Subsequent parse actions take place based on the parse state represented by MN_{new}. Note that a back-tracking arrow pointing from MN_{new} to N_k has to be established. This indicates that N_k is the previous active top. If later on MN_{new} is deemed "not explorable", the control has to be returned to N_k. Then N_k can pick the next top item in its "not yet visited" candidate list, L_{i+1}, as the next alternative, to expand the MPS.

Candidate Node Trimming

When processing maps of large sizes, the size of the MPS will also become very large. In order to improve the performance of the grammar parser, it is necessary to study how the MPS grows as the map understanding process advances.

Let P be the number of the productions in the map grammar, let M_P be the maximum possible size of the right hand side of a production, and let S be the number of primitives in the primitive collection. S represents the size of the map. In the extreme case, an MPS path may need to expand all the productions. Therefore the upper bound of the lengths of MPS paths is $P \times M_P$. In the worst case scenario, each MPS node may include all the primitives in the collection in its candidate list. This means that each MPS node may have S child MPS nodes. Consequently, the number of the possible paths in the MPS may be $S^{P \times M_P - 1}$. This indicates that the size of the MPS grows exponentially as the size of the primitive collection increases. In addition, the upper bound of the sizes of the candidate lists is equal to the size of the primitive collection. This will consume a huge amount of memory space and computing power.

To improve the performance of the parsing algorithm, the following have to be addressed: (1) How to reduce the size of a candidate list? and (2) How to reduce the size of the MPS? The size of the candidate list associated with each MPS node can be dramatically cut down by enabling the input selector with ability to choose candidate primitives from a small neighboring region. The MPS can also be shrunk to a much smaller size by trimming some of the paths that are traveled earlier. As has been

mentioned earlier, each p-node is associated with a back-tracking arrow in case the parser needs to track back into the history and start a new path by choosing a new candidate token from the candidate list. Fortunately, it is not always necessary to track back to the parent p-node. When a p-node representing a higher-level feature is shifted in, this feature has been recognized with a high degree of confidence and therefore there is no need to tracking back to the history. Take a look at the **OIR** shown in Figure 6.1. If **simple_river_section** instances $CS1$, $CS2$ and $CS3$ are recognized with a high degree of confidence, the possibility that $CS1$, $CS2$ and $CS3$ be popped from the MPS is low enough so that it is not necessary to maintain the back-tracking arrows when these **simple_river_section** instances are shifted in. This means that the candidate list associated with the MPS node pointed to by the back-tracking arrow can be trimmed. Recall that a back-tracking arrow traces its way back to the root MPS node. Therefore, all the MPS nodes along the back tracking path, as well as their associated candidate lists, can be removed.

7.6 The Map Parsing Algorithm

The parsing algorithm in a pseudo programming language is shown in Algorithm 7.2.

Algorithm 7.2 ***(Map Parsing Algorithm)***

procedure *Parsing()* ***is***

begin

// Initially the MPS is empty. The first p-node pushed in is the root.

```
// Push the start state S0 onto the MPS. And make it the
// current active top.
push(S0, MPS);

// Prepare candidate list of the current stack top.
prepare_candidates();

// Continue until told to stop
L1 :
loop while parsing_not_done
  // Ask the input selector to pick up the next input token from
  // the input collection.
  scan();

  // If no appropriate lookahead token is found from the
  // candidate list, we need to track back to the previous
  // active top.
  if no_proper_candidate_is_found
    regress_to_previous_active_top();
    goto L1;
  end if
```

```
// Based on the lookahead, look up the ACTION table.
// Let S be the current parse state, T be the input token.
act = get_action(S, T);

if no proper action can be determined
  an error occurred, stop parsing.
end if

if act indicates shift action
  // Create an MPS node N0 which holds the next parse
  // state and the shifted in symbol.
  N0 = create_node(GOTO(S, T), T)
  // Push the input token to the MPS. That is, make the current
  // active top (Top) the parent of N0.
  push(N0);
  set_active_top(N0);
  // Point a backtracking arrow to the previous active top.
  set_previous_active_top(Top);
end if
if act indicates reduce action
```

// Obtain the left hand side symbol "lhs" and the size

// "size_of_handle" of the handle of the production to reduce.

get_lhs_and_handle_size();

// Let TC_0 point to the current active top, that is,

// remember it for later reference.

TC_0 = get_active_top_of_mps(MPS);

// Remove the node that holds the current input token and.

// mark it as "visited".

put_token_to_visited(TC_0);

// let pt_0 point to the current top.

pt_0 = TC_0;

// Loop for size_of_handle times. This loop will try to find

// the current top's ancestor "size_of_handle" levels up.

***loop from** 1 **to** size_of_handle*

  // let pt_0 points to its own parent by tracing back in the direction

  // of its incoming path arrow.

  pt_0 = parent_of(pt_0);

end loop

// Create a new MPS node and make it hold the left hand side symbol.

```
        MN_new = new_mps_node(lhs_symbol);
        // Let TC_0 be its previous active top.
        set_previous_active_top(MN_new, TC_0);

        // Push MN_new into the MPS.
        push(MN_new);
      end if
      if act indicates the accept action.
        // indicate parsing is successfully finished.
        set_flag(parsing_is_done);
      end if
    end loop
  end
```

7.7 An Example

As an example, a map grammar $\mathcal{MG}$ that deals with a limited set of map symbols is given in Section 7.7.1. Note that the identifiers that occur in the same production rule, such as a in M[a:map], refer to the same instances. The same instance identifiers occurring in different rules are not related. Also note that symbols with prefix "T_" stand for terminals that represent m-symbols. Non-terminals are written in upper-case letters, p-symbols and c-symbols written in lower-case. Brief descriptions of the

meanings of the map symbols can be found in Appendix C.

7.7.1 A simple map grammar $\mathcal{MG}$

---

0. M[a:map] ::= RNS[b:rns] is_fullrns[b]

1. $START ::= M[a:map] EOF[eof]

2. RNS[b:rns] ::= RNS[c:rns] RN[d:rn]

3. RNS[c:rns] ::= RN[c:rn]

4. RN[c:RN] ::= PRN[c:prn] is_fullprn[e]

5. PRN[b:prn] ::= PRN[b:prn] RS[r:rs] attaching[(b, r):rnh]

6. PRN[b:prn] ::= RS[r:rs]

7. RS[r:rs] ::= PRS[p:prs] full_prs[p]

8. PRS[p:prs] ::= T_SRS[s:srs]

9. PRS[p:prs] ::= PRS[u:prs] BRIG[b:brig] f_barline_of[b:x] p_brig_srs_connecting[u, x] f_barline_of[b:y] p_other_barline[b, x, y] T_SRS[v:srs] p_not_in_prs[v] p_brig_srs_connecting[v,y] p_form_two_sides[u, x, v, y]

10. PRS[p:prs] ::= PRS[u:prs] T_CITY[c:city] p_touch[u, c] T_SRS[v:srs] p_not_in_prs[u] p_touch[v, c]

11. PRS[p:prs] ::= PRS[u:prs] T_SRS[v:srs] p_not_in_prs[u] p_form_gap[u,v]

12. BRIG[b:brig] ::= T_LNS[l:lns] p_is_barline[l] T_LNS[m:lns] p_is_barline[m] p_form_bridge[l,m]

$\mathcal{MG}$ is a map grammar for modeling a map that contains road networks. It is derived from the relevant terminological axioms given in Section 5.2 of Chapter 5. These axioms are 1, 4, 5, and 6. Standard LR(k) parsers deal with grammars without ambiguities. However, this is not the case with the proposed map grammars. Normally, ambiguities are common in many language sentences. Because map grammars are syntactically more flexible than text grammars, ambiguities are almost inevitable. For example, consider production rules 10 and 11 in $\mathcal{MG}$. This results in a conflict between the shift action of T_CITY[t:city] and that of T_SRS[v:srs].

There are two types of grammar ambiguities: shift-shift and shift-reduce conflicts. In standard parsers, ambiguities are usually resolved by specifying priorities or associations. In map parsers, those terminals involved in a shift-shift conflict indicate all the possible "next" tokens under the "current" parse state. They are actually organized as the candidate list associated with the MPS node representing the current parse state.

One point to note is the reusability of tokens. For standard parsers, once a token produced by the input scanner is consumed, that is, shifted in, it will be removed from the input token list and cannot be reused. However, in the map parsing process, an

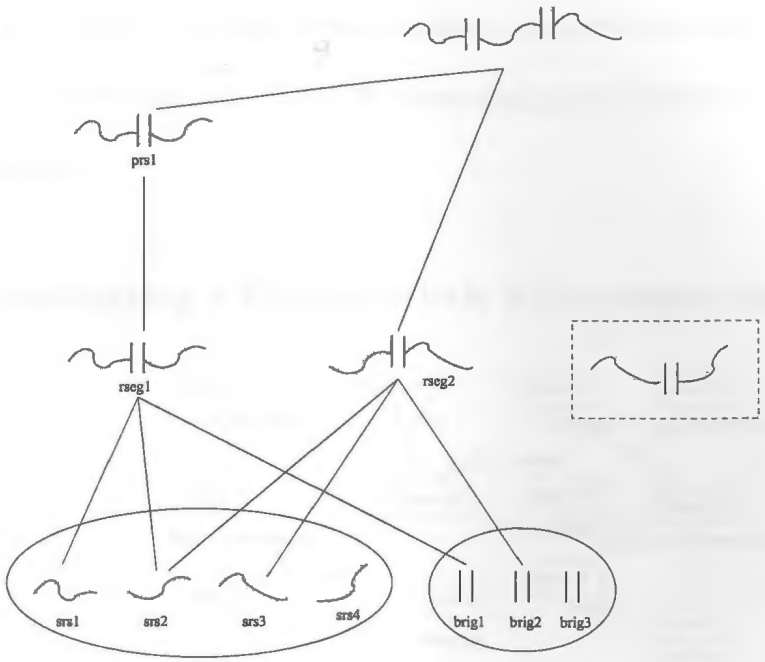


Figure 7.5: Reusable tokens.

input token can be reused. That is, after a token is selected from the input collection and shifted in, it may not be removed from the input collection and prevented from participating in the further parsing process. Such a token can be shifted in multiple times under different parsing states. For example, as illustrated in Figure 7.5, through a reduce action based on production 7 in $\mathcal{MG}$, a PRS symbol prs_1 is built with two RSEG symbols $rseg_1$ and $rseg_2$. In turn, $rseg_1$ is built with symbols srs_1, $brig_1$, and srs_2, and $rseg_2$ built with srs_2, $brig_2$, and srs_3. It is obvious that after being shifted to build $rseg_1$, srs_2 can be shifted in again to build $rseg_2$ through a reduce action with respect to production rule 9. Also note that after participating in the formation of symbols $rseg_1$ and $rseg_2$, srs_2 can no longer be shifted in when the parser is working towards reducing to a RSEG symbol. That is why a constraint symbol p_not_in_prs is

introduced. It indicates that a predicate has to be evaluated to make sure a T_SRS symbol is not a middle part of a PRS. Therefore, srs_2 is such a symbol that does not fulfill this predicate.

7.7.2 Constructing a Characteristic Finite State Machine

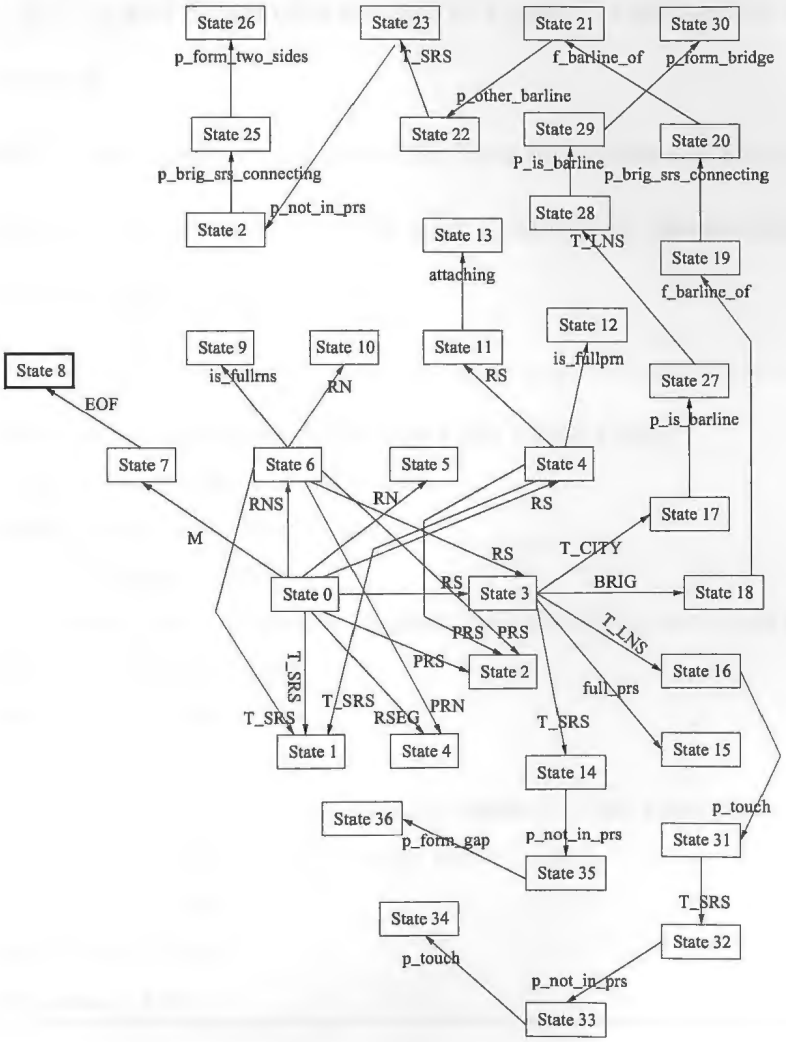


Figure 7.6: CFSM for $\mathcal{MG}$.

It is well known that a finite state machine called Characteristic Finite State Machine (CFSM) can be built based on a grammar. Each CFSM state holds a set of configurations. A configuration is an item of the form

$$A \rightarrow X_1 \cdots X_i \bullet X_{i+1} \cdots X_j,\ l$$

where A, X_1, $\cdots$, and X_j are map symbols in a grammar production, l is a list of lookahead terminals.

The CFSM of $\mathcal{MG}$ is shown in Figure 7.6. Note that State 8 is the accept state. The parse states are given below. For the sake of simplicity, the brackets following each symbol are ignored.

State [0]:

PRS ::= • PRS T_SRS p_not_in_prs p_form_gap , { full_prs, T_SRS, T_CITY, T_LNS }

PRS ::= • T_SRS , { full_prs, T_SRS, T_CITY, T_LNS }

PRN ::= • PRN RS attaching , { is_fullprn, T_SRS }

RNS ::= • RNS RN , { is_fullrns, T_SRS }

PRS ::= • PRS T_CITY p_touch T_SRS p_not_in_prs p_touch , { full_prs, T_SRS, T_CITY, T_LNS }

RS ::= • PRS full_prs , { is_fullprn, T_SRS }

RN ::= • PRN is_fullprn , { is_fullrns, T_SRS }

S ::= • M EOF, { λ }

PRS ::= • PRS BRIG f_barline_of p_brig_srs_connecting f_barline_of p_other_barline T_SRS p_not_in_prs p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

PRN ::= • RS , { is_fullprn, T_SRS }

RNS ::= • RN , { is_fullrns, T_SRS }

M ::= • RNS is_fullrns , { EOF }

State [1]:

PRS ::= T_SRS • , { full_prs, T_SRS, T_CITY, T_LNS }

State [2]:

PRS ::= PRS • T_CITY p_touch T_SRS p_not_in_prs p_touch , { full_prs, T_SRS, T_CITY, T_LNS }

RS ::= PRS • full_prs , { is_fullprn, attaching, T_SRS }

BRIG ::= • T_LNS p_is_barline T_LNS p_is_barline p_form_bridge , { f_barline_of }

PRS ::= PRS • BRIG f_barline_of p_brig_srs_connecting f_barline_of p_other_barline T_SRS p_not_in_prs p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

PRS ::= PRS • T_SRS p_not_in_prs p_form_gap , { full_prs, T_SRS, T_CITY, T_LNS }

State [3]:

PRN ::= RS • , { is_fullprn, T_SRS }

State [4]:

PRS ::= • T_SRS , { full_prs, T_SRS, T_CITY, T_LNS }

PRN ::= PRN • RS attaching , { is_fullprn, T_SRS }

PRS ::= • PRS T_CITY p_touch T_SRS p_not_in_prs p_touch , { full_prs, T_SRS, T_CITY, T_LNS }

RS ::= • PRS full_prs , { attaching }

RN ::= PRN • is_fullprn , { is_fullrns, T_SRS }

PRS ::= • PRS BRIG f_barline_of p_brig_srs_connecting f_barline_of p_other_barline T_SRS p_not_in_prs p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

PRS ::= • PRS T_SRS p_not_in_prs p_form_gap , { full_prs, T_SRS, T_CITY, T_LNS }

State [5]:

RNS ::= RN • , { is_fullrns, T_SRS }

State [6]:

PRS ::= • PRS T_SRS p_not_in_prs p_form_gap , { full_prs, T_SRS, T_CITY, T_LNS }

PRS ::= • T_SRS , { full_prs, T_SRS, T_CITY, T_LNS }

RNS ::= RNS • RN , { is_fullrns, T_SRS }

PRN ::= • PRN RS attaching , { is_fullprn, T_SRS }

PRS ::= • PRS T_CITY p_touch T_SRS p_not_in_prs p_touch , { full_prs, T_SRS, T_CITY, T_LNS }

RS ::= • PRS full_prs , { is_fullprn, T_SRS }

RN ::= • PRN is_fullprn , { is_fullrns, T_SRS }

PRS ::= • PRS BRIG f_barline_of p_brig_srs_connecting f_barline_of p_other_barline T_SRS p_not_in_prs p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

PRN ::= • RS , { is_fullprn, T_SRS }

M ::= RNS • is_fullrns , { EOF }

State [7]:

S ::= M • EOF, { λ }

State [8]:

S ::= M EOF • , { λ }

State [9]:

M ::= RNS is_fullrns • , { EOF }

State [10]:

RNS ::= RNS RN • , { is_fullrns, T_SRS }

State [11]:

PRN ::= PRN RS • attaching , { is_fullprn, T_SRS }

State [12]:

RN ::= PRN is_fullprn • , { is_fullrns, T_SRS }

State [13]:

PRN ::= PRN RS attaching • , { is_fullprn, T_SRS }

State [14]:

PRS ::= PRS T_SRS • p_not_in_prs p_form_gap , { full_prs, T_SRS, T_CITY, T_LNS }

State [15]:

RS ::= PRS full_prs • , { is_fullprn, attaching, T_SRS }

State [16]:

PRS ::= PRS T_CITY • p_touch T_SRS p_not_in_prs p_touch , { full_prs, T_SRS, T_CITY, T_LNS }

State [17]:

BRIG ::= T_LNS • p_is_barline T_LNS p_is_barline p_form_bridge , { f_barline_of }

State [18]:

PRS ::= PRS BRIG • f_barline_of p_brig_srs_connecting f_barline_of p_other_barline T_SRS p_not_in_prs p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

State [19]:

PRS ::= PRS BRIG f_barline_of • p_brig_srs_connecting f_barline_of p_other_barline T_SRS p_not_in_prs p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

State [20]:

PRS ::= PRS BRIG f_barline_of p_brig_srs_connecting • f_barline_of p_other_barline T_SRS p_not_in_prs p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

State [21]:

PRS ::= PRS BRIG f_barline_of p_brig_srs_connecting f_barline_of • p_other_barline T_SRS p_not_in_prs p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

State [22]:

PRS ::= PRS BRIG f_barline_of p_brig_srs_connecting f_barline_of p_other_barline • T_SRS p_not_in_prs p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

State [23]:

PRS ::= PRS BRIG f_barline_of p_brig_srs_connecting f_barline_of p_other_barline T_SRS • p_not_in_prs p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

State [24]:

PRS ::= PRS BRIG f_barline_of p_brig_srs_connecting f_barline_of p_other_barline T_SRS p_not_in_prs • p_brig_srs_connecting p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

State [25]:

PRS ::= PRS BRIG f_barline_of p_brig_srs_connecting f_barline_of p_other_barline T_SRS p_not_in_prs p_brig_srs_connecting • p_form_two_sides , { full_prs, T_SRS, T_CITY, T_LNS }

State [26]:

PRS ::= PRS BRIG f_barline_of p_brig_srs_connecting f_barline_of p_other_barline T_SRS p_not_in_prs p_brig_srs_connecting p_form_two_sides • , { full_prs, T_SRS, T_CITY, T_LNS }

State [27]:

BRIG ::= T_LNS p_is_barline • T_LNS p_is_barline p_form_bridge , { f_barline_of }

State [28]:

BRIG ::= T_LNS p_is_barline T_LNS • p_is_barline p_form_bridge , { f_barline_of }

State [29]:

BRIG ::= T_LNS p_is_barline T_LNS p_is_barline • p_form_bridge , { f_barline_of }

State [30]:

BRIG ::= T_LNS p_is_barline T_LNS p_is_barline p_form_bridge • , { f_barline_of }

State [31]:

PRS ::= PRS T_CITY p_touch • T_SRS p_not_in_prs p_touch , { full_prs, T_SRS, T_CITY, T_LNS }

State [32]:

PRS ::= PRS T_CITY p_touch T_SRS • p_not_in_prs p_touch , { full_prs, T_SRS, T_CITY, T_LNS }

State [33]:

PRS ::= PRS T_CITY p_touch T_SRS p_not_in_prs • p_touch , { full_prs, T_SRS, T_CITY, T_LNS }

State [34]:

PRS ::= PRS T_CITY p_touch T_SRS p_not_in_prs p_touch • , { full_prs, T_SRS, T_CITY, T_LNS }

State [35]:

PRS ::= PRS T_SRS p_not_in_prs • p_form_gap , { full_prs, T_SRS, T_CITY, T_LNS }

State [36]:

PRS ::= PRS T_SRS p_not_in_prs p_form_gap • , { full_prs, T_SRS, T_CITY, T_LNS }

State	Lookahead Terminals																
	T0	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17
0						S											
1					R8	R8						R8		R8			
2					S	S						S		S			
3			R6			R6											
4			S			S											
5		R3				R3											
6		S				S											
7	A																
8																	
9	R0																
10		R2				R2											
11				S													
12		R4				R4											
13			R5			R5											
14							S										
15			R7	R7		R7											
16													S				
17															S		
18								S									
19									S								
20								S									
21										S							
22						S											
23							S										
24									S								
25											S						
26					R9	R9						R9		R9			
27														S			
28															S		
29																S	
30								R12									
31						S											
32							S										
33													S				
34					R10	R10						R10		R10			
35																	S
36					R11	R11						R11		R11			

Table 7.1: The action table for $\mathcal{MG}$.

State	Symbol																								
	T0	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	NT0	NT1	NT2	NT3	NT4	NT5	NT6	NT7
0						1													7	6	5	4	3	2	
1																									
2					15	14						16		17											18
3																									
4			12			1																	11	2	
5																									
6		9				1															10	4	3	2	
7	8																								
8																									
9																									
10																									
11				13																					
12																									
13																									
14							35																		
15																									
16													31												
17															27										
18								19																	
19									20																
20								21																	
21										22															
22						23																			
23							24																		
24									25																
25											26														
26																									
27														28											
28															29										
29																30									
30																									
31						32																			
32							33																		
33													34												
34																									
35																	36								
36																									

Table 7.2: The goto table for $\mathcal{MG}$.

The ACTION and GOTO tables obtained from the CFSM are shown in Tables 7.1 and 7.2 respectively. Column headings starting with T stand for terminals, and those starting with NT stand for non-terminals. Their corresponding map grammar symbols are given below:

T0: EOF, T1: error, T2: is_fullrns, T3: is_fullprn, T4: attaching, T5: full_prs, T6: T_SRS, T7: p_not_in_prs, T8: f_barline_of, T9: p_brig_srs_connecting, T10: p_other_barline, T11: p_form_two_sides, T12: T_CITY, T13: p_touch, T14: T_LNS, T15:p_is_barline, T16: p_form_bridge, T17: p_form_gap

NT0: S, NT1: M, NT2: RNS, NT3: RN, NT4: PRN, NT5: RS, NT6: PRS, NT7: BRIG

Note that T1 is the symbol indicating an error state. In Table 7.1, "S" denotes a shift action, "R" denotes a reduce action, and the integer following the "R" specifies the production number used for the reduce action. Note that T1 is ignored from the tables, because an empty cell in the table represents an error state. In the system, these tables are stored as a list of table rows. However, the length of a row is not equal to the number of cells in each row. Each element in a row is a pair (terminalID, terminalValue). An empty cell value (error state) does not appear in the row. The last element in a row is a special value indicating the end of the row.

7.7.3 Illustration of the Parsing Process

Now that the basic data structures needed by a map parser have become available, we are ready to follow the trace of the map parser on the test map shown in Figure 7.7. The first few steps of the parsing process are discussed in detail, followed by a table that lists the changes made to the MPS by the remaining parsing steps.

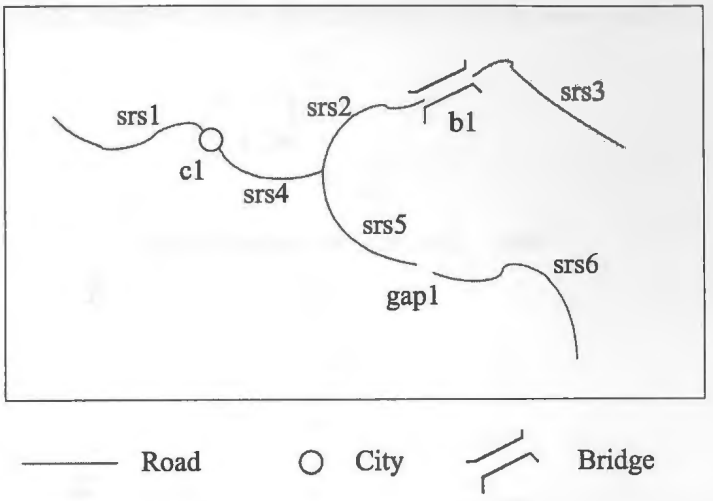


Figure 7.7: A test map with only road networks.

During the parsing process, the MPS experiences a series of parse states. Its structure at certain important points of the trace will be shown. In the beginning, the MPS for the parser has only one p-node with parse state 0, shown in Figure 7.9. The initial **OIR** graph associated with this parse state is also shown. All of its instance holders, except primitive instance holders, are empty. With the progress of the parsing, more than one p-node with the same parse state may emerge, so each p-node is given a node name, appearing at the upper left corner. Its candidate list has only one c-node, which expects a terminal of type T_SRS. At this point, p-node

N0 is the active top. Note that p-nodes are drawn as square boxes, and c-nodes are drawn as rounded boxes. The current active top item is represented by means of marking it with a black dot. Note that different styles of arrowed lines are used to indicate different types of relations between the MPS nodes (see Figure 7.8).

Figure 7.8: Arrowed lines that represent MPS node relationships.

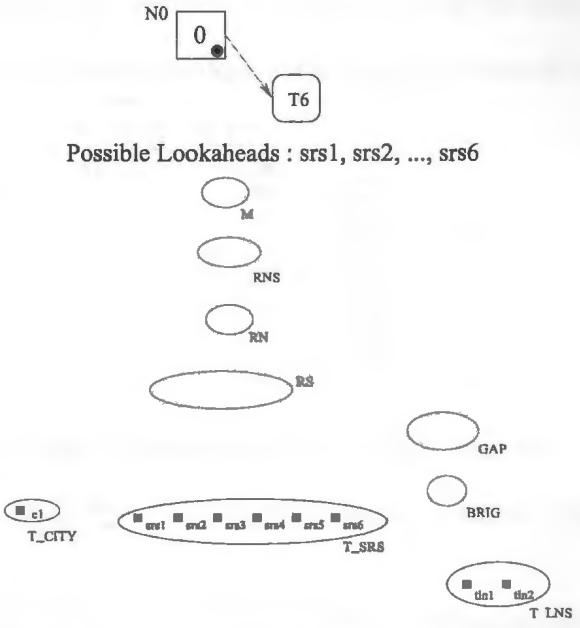
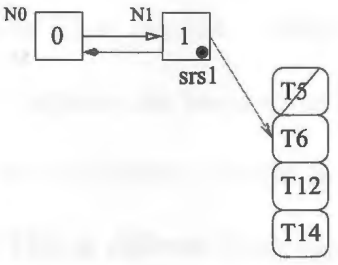


Figure 7.9: Illustration of the parsing process.

In each of the illustrations, the possible lookahead symbols of the current active top item are also listed. These lookahead symbols are grabbed from the input collection based on the types given in the associated candidate list. Usually multiple input symbols may be appropriate for lookaheads. We may randomly choose one from such a list as the lookahead, or we may use other selection strategies. For instance, a rule

based system may be exploited to deal with a large list of possible lookaheads, since the order of examining is very important to the system's performance. We may avoid excessive "tracking back" operations if the "right" lookaheads are examined earlier.

In this example, srs_1 is picked as the next lookahead. According to the action table, srs_1 suggests a shift operation so that a new p-node N1 with parse state 1 is created. The new p-node N1 takes control and becomes the active top, which is shown in Figure 7.10. The currently shifted-in terminal srs_1 is put under its corresponding active top N1. The shift operation does not change the state of the **OIR**.



Possible Lookaheads: srs1, srs2, ..., srs6, c1, tln1, tln2

Figure 7.10: Illustration of the parsing process. (cont'd)

There may or may not be a "tracking back" operation at this stage. If later on a "tracking back" operation is triggered and p-node N0 becomes the active top again, it means srs_1 has been determined not to be a fit for the lookahead. Another symbol should then be picked from the possible candidate list as the current lookahead, and the parsing process continues. Afterwards, srs_1 will be removed from the candidate list and marked "visited".

The candidate list of p-node N1 (Figure 7.10) indicates that four types of terminals

are expected: full_prs (T5), T_SRS (T6), T_CITY (T12), and T_LNS (T14). There may exist zero or more instances for each listed type. Since a road section is viewed as a partial road section that cannot be stretched any more, T5 (full_prs) means a fact specifying that the parts currently under consideration compose a road section. Such a fact is established by invoking a function, or procedure, that draws conclusions based on the overall situation of the current parse stage, which is the current state of the **OIR** and the primitive collection. In this example, since there are some instances attached to srs_1, we can draw the conclusion that no terminal of type T5 can be found. As a result, c-node T5 is marked "visited" with a diagonal line. We then examine whether any T6 instance can become the next lookahead symbol.

Note that although srs_1 is already shifted in, it is still listed as a lookahead candidate for p-node 1. This is different from the standard parser, which does not use a shifted-in token more than once. The reason is that a single object can take part in the construction of more than one higher level map object. For example, a "bridge" instance may be shifted in as a building block of a river section, and later shifted in again as a building block of a road section.

To proceed from p-node N1, what was performed on p-node N0 is repeated on p-node N1. Assume that srs_1 is picked as the current lookahead, which indicates a reduce action with respect to production 8. According to the parsing algorithm, a new p-node N2 is created to represent the left hand side of the production 8, which is prs_1. Again, its candidate list is also shown. Since p-node N0 is the immediate

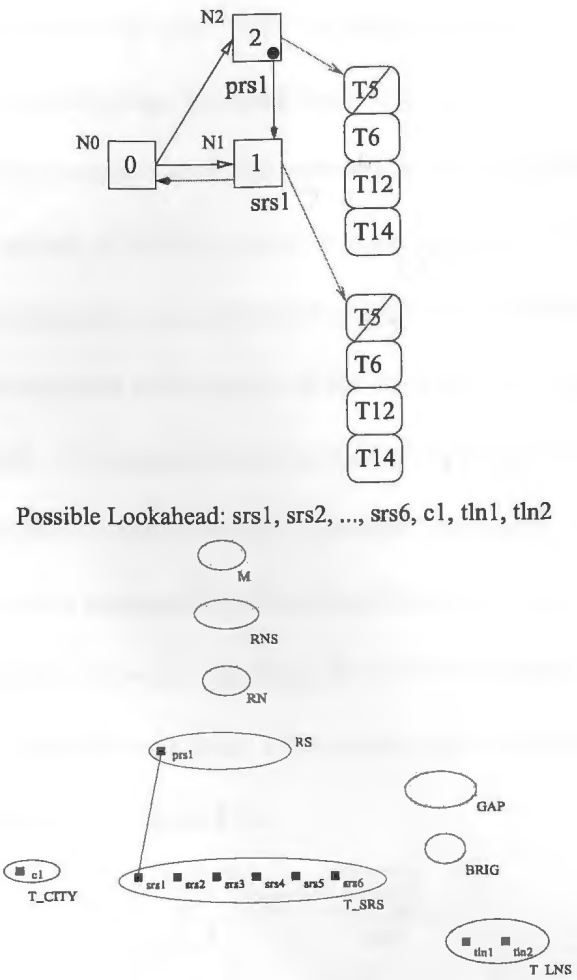


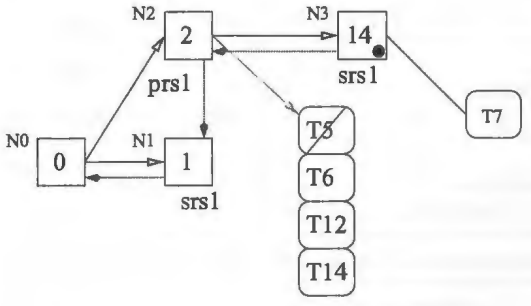
Figure 7.11: Illustration of the parsing process. (cont'd)

ancestor of the handle, p-node N2 becomes one of its children. Note that p-node N2 turns into the active top, and p-node N1 is its previous top. The resulting MPS is shown in Figure 7.11.

The **OIR** graph is modified due to the reduce action. Production 8 specifies that the existence of a type "T_SRS" (simple road segment) instance implies the existence of an instance of type "RS" (road section). A new instance $prs1$, which stands for a

partially recognized instance of type "RS", is placed in the RS instance holder. At this point, only one assembly edge has been explicitly shown. Later on more assembly edges are added to this instance with the proceeding of the parsing. Each time new assembly edges are added, a different name is used to label the instance, until it is fully recognized. For example, the parser will change its label from $prs1$ to $prs2$ if $c1$ and $srs4$ are also determined to be parts of the road section (see step 38 and 39 in Table 7.3 on page 196). This means that the road section instance has changed from one "partially-recognized" state to another "partially-recognized" state.

The p-node N2 is also associated with a candidate list consisting of T5, T6, T12 and T14. Once again the function checking for predicate symbols indicates that no instance of type T5 is available. A type T6 terminal srs_1 is shifted in and parse state 14 is reached, as depicted in Figure 7.12.



Possible Lookahead: p_not_in_prs

Figure 7.12: Illustration of the parsing process. (cont'd)

The first few steps of the parsing process are sufficient to illustrate how the parse actions (shift and reduce) affect the MPS. To save space, we will list the rest of the

parsing steps in Table 7.3, where each row lists the following information: (1) step number, (2) current top node, its parse state, and the shifted-in symbol, (3) the previous top item, (4) the parent node, (5) the candidate list of the current top node, and (6) the lookahead selected for the current top and the action based on the current top and the lookahead. Actions can be shift (S), reduce (R) and tracking back (TK). Visited terminal types in the "Candidate List" column are underlined. If the action is reduce or tracking back, the lookahead symbol is not shown, since any instance of the current candidate types is enough for the parser to decide the next action.

Table 7.3: Remaining parse steps of the example.

Step	MPS				LK/ACT
	C.T/P.S/SYM	P.Top	P	Candidate List	
(4)	N3/14/srs1	N2	N2	T7	-/TK
(5)	N2/2/prs1	N1	N0	T5,T6,T12,T14	srs2/S
(6)	N4/14/srs2	N2	N2	T7	p_not_in_prs1/S
(7)	N5/35/p_not_in_prs1	N4	N4	T17	-/TK
(8)	N4/14/srs2	N2	N2	T7	-/TK
(9)	N2/2/prs1	N1	N0	T5,T6,T12,T14	srs3/S
a series of steps similar to (5)(6)(7)(8) repeated on srs3,srs4,srs5,srs6					
(25)	N2/2/prs1	N1	N0	T5,T6,T12,T14	c1/S
(25)	N6/16/c1	N2	N2	T13	p_touch1/S
shift-ins of srs1,srs2,srs3 result in track backs					
(35)	N7/31/p_touch1	N6	N6	T6	srs4/S
(36)	N8/32/srs4	N7	N7	T7	p_not_in_prs2/S
(37)	N9/33/p_not_in_prs2	N8	N8	T13	p_touch2/S
(38)	N10/34/p_touch2	N9	N9	T5,T6,T12,T14	-/R10
(39)	N11/2/prs2	N10	N0	T5,T6,T12,T14	full_prs1/S

(40)	N12/15/full_prs1	N2	N2	T3,T4,T6	-/R7
(41)	N13/3/rs1	N12	N0	T3,T6	-/R6
(42)	N14/4/prn1	N13	N0	T2,T6	srs2/S
(43)	N15/1/srs2	N14	N14	T5,T6	-/R8
(44)	N16/2/prs2	N15	N14	T5,T6,T12,T14	tln1/S
(45)	N17/17/tln1	N2	N2	T15	p_is_barline1/S
(46)	N18/27/p_is_barline1	N17	N17	T14	tln2/S
(47)	N19/28/tln2	N18	N18	T15	p_is_barline2/S
(48)	N20/29/p_is_barline2	N19	N19	T16	p_form_b1/S
(49)	N21/30/p_form_bridge1	N20	N20	T8	-/R12
(50)	N22/18/brig1	N21	N16	T8	f_barline_of/S
(51)	N23/19/tln1	N22	N22	T9	p_bs_ct1/S
(52)	N24/20/p_bs_ct1	N23	N23	T8	f_barline_of/S
(53)	N25/21/p_bs_ct1	N24	N24	T10	p_other_b/S
(54)	N26/22/p_other_b	N25	N25	T6	srs3/S
(55)	N27/23/srs3	N26	N26	T7	p_not_in_prs3/S
(56)	N28/24/p_not_in_prs3	N27	N27	T9	p_bs_ct2/S
(57)	N29/25/p_bs_ct2	N28	N28	T11	p_fts1/S
(58)	N30/26/p_fts1	N29	N29	T5,T6,T12,T14	-/R9
(59)	N31/2/prs3	N30	N11	T5,T6,T12,T14	full_prs2/S
(60)	N32/15/full_prs2	N31	N31	T3,T4,T6	-/R7
(61)	N33/11/rs2	N32	N11	T4	attaching1/S
(62)	N34/13/attaching1	N33	N33	T3,T6	-R5
(63)	N35/4/prn2	N34	N0	T3,T6	srs5/S
(64)	N36/1/srs5	N35	N35	T5,T6	-/R8
(65)	N37/2/prs4	N36	N35	T5,T6,T12,T14	srs6§
(66)	N38/14/srs6	N37	N37	T7	p_not_in_prs4/S
(67)	N39/35/p_not_in_prs5	N38	N38	T17	p_form_gap1/S
(68)	N40/36/p_form_gap1	N39	N39	T5,T6,T12,T14	-/R11
(69)	N41/2/prs5	N40	N35	T5,T6,T12,T14	full_prs3/S
(70)	N42/15/full_prs3	N41	N41	T3,T4,T6	-/R7

(71)	N43/11/rs3	N42	N35	T4	attaching2/S
(72)	N44/13/attaching2	N43	N43	T3,T6	-/R5
(73)	N45/4/prn3	N44	N0	T3,T6	is_fullprn1/S
(74)	N46/12/is_fullprn1	N45	N45	T2,T6	-/R4
(75)	N47/5/rn1	N46	N0	T2,T6	-/R3
(76)	N48/6/rns1	N47	N0	T2,T6	is_fullrns1/S
(77)	N49/9/is_fullrns1	N48	N48	T0	-/R0
(78)	N50/7/m1	N49	N0	T0	Accept

Figure 7.13 shows the output of the parsing process, which is a complete **OIR** graph.

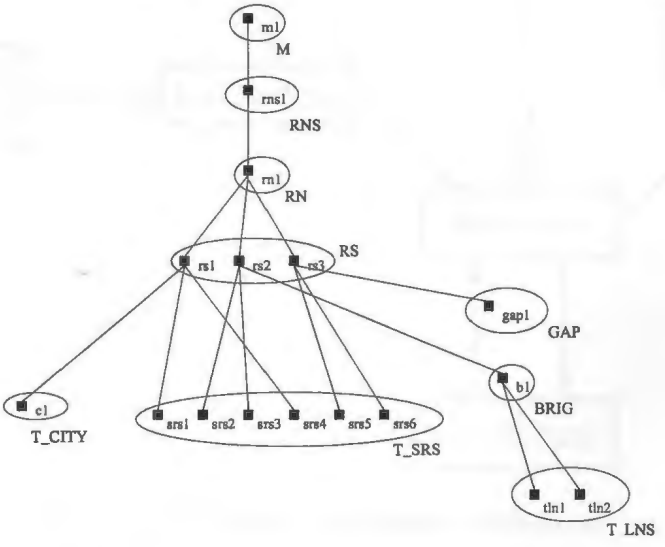


Figure 7.13: The complete **OIR** produced by the parser.

7.8 Prototype System Architecture

In this section an illustration of the architecture of the prototype system is given in Figure 7.14. The steps and tools used in the implementation of the prototype system are also described. Subsystem modules are represented by rectangles. The filled arrows indicate the invocation relations between modules. The empty-headed arrows specify the external data input.

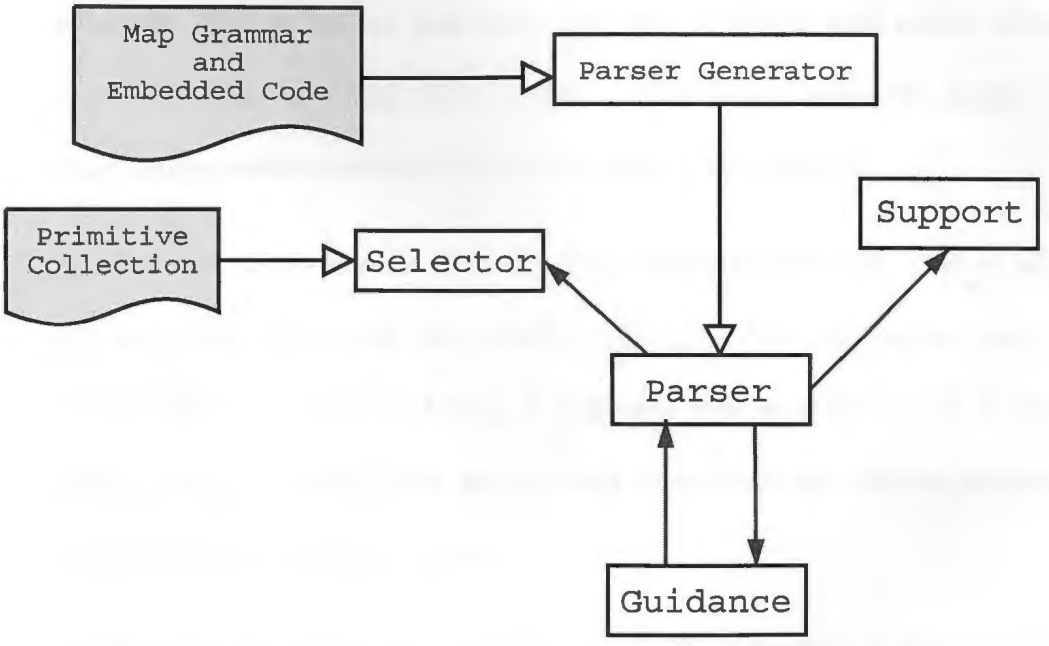


Figure 7.14: Prototype system architecture.

The system was implemented in Java (JDK 1.3) and C programming languages. The following steps are involved in the prototype development process:

1. Development of the parser generator for the map grammar. The parser generator module takes in the grammar representation and produces the Java source

code for the map parser. The parser is then compiled and becomes a part of the interpretation system. Note that this parser generator is different from the traditional ones in that it has to implement the map parsing algorithm described in Section 7.6.

2. Specification of the map grammar and embedded Java code. Users have to specify the map grammar to be used in the format required by the parser generator. The embedded Java code is invoked to handle each reduce action performed. One advantage of this system is that we can adapt the system to other interpretation problems by simply changing the grammar.

3. Development of the selector module. The selector performs the task to pick the next input token from the primitive collection. Since the parser input is not organized as a string of tokens, a number of data structures, such as hash tables, heaps and linked lists, are provided to facilitate the efficient indexing and retrieving of primitive objects.

4. Development of the guidance module. The guidance module is a key part in the parsing process. Its function is to guide the selection of primitives and constraint checking functions using domain knowledge. For example, domain expertise is required to define the thresholds and parameters of the gap filling function. Sometimes the parameters have to be adaptive according to different situations. In the current implementation, if the distance between the two closer

end points of two line segments is smaller than 5 pixels, they are considered to belong to one line. If the distance is bigger than 5 pixels and smaller than 15 pixels, they belong to the same line only if they fit in a flat curve. We chose the Levenberg-Marquardt nonlinear regression algorithm as the curve fitting algorithm. The parser and guidance modules work in coordination with each other. In the current system, the logic of the guidance modules is implemented directly in the selection module or the constraint checking functions. Rule engines are a powerful mechanism to store and apply domain knowledge. The possibility of using JESS (v5.0), an expert system shell, as a rule engine has been studied. A mechanism to map Java objects to JESS facts can be adopted. In future research, a rule engine based on JESS may be integrated into the proposed system. In such a system, a constraint checking function does not have to implement the knowledge inference logic. It only has to prepare the Java objects that indicate the current states of the parser and the primitive collection. At the JESS side, these Java objects can appear in the form of JESS facts on which the JESS rules can perform the constraint checking reasoning. The result is then sent back to the constraint checking function at the Java side. In this way the procedural knowledge and the declarative can be further separated.

5. Development of the support module. This module is designed to hold and manage a set of low-level image process procedures written in C programming

language. These procedures are executed when it is necessary to verify a spatial object or relation in case such information is not readily included in the primitive collection. Java native interface is used to enable Java code to invoke these C programs.

Just like traditional parser development, each map grammar production is attached with a procedure implemented in Java. Such a procedure implements the operations to be performed after a reduce action is triggered. The map grammar, together with the associated procedure in the form of embedded codes, is supplied to the parser generator as input. The parser generator produces the map parser in Java code. The map understanding system is built around the generated map parser, whose implementation is discussed in detail in Section 7.5. The map parser works together with three other modules: selector, guidance and support.

7.9 Dealing with Uncertainty in Map Understanding Process

Map grammar parsing can be considered a heuristic search process. Each map grammar production expresses a heuristic that can be applied in the **OIR** formation, which tells what kind of "new" instances and relations can be discovered given a certain context (existence of relevant instances and relations). On many occasions, complexities of maps prevent us from adding correct new instances or relations to an **OIR**.

Two types of uncertainties are considered in the map understanding process: (1) distorted or missing features resulted from the low-level map processing; and (2) map objects with fuzzy features. A lot of low-level processing algorithms can satisfactorily extract and vectorize line features and other map features. However, each such algorithm also has its own limitations, which generates various distorted results. Some map features are difficult to be described with precise terms. Therefore, Uncertainty measures are needed in many map understanding process steps.

One advantage of the proposed semantic representation is that it provides a natural and intuitive approach to deal with uncertainty reasoning. One of the common problems is the broken line phenomenon; that is, line features are broken into a number of segments caused by poor quality results of previous image processing options. Various forms of gaps have to be filled. Therefore, gaps should be introduced as a concept category in the proposed semantic representation. Its relationship with other concepts has to be explicitly represented as shown in Figure 7.15.

Let a be a "partial road section" instance and b be a "road segment" instance. We say a and b are two consecutive sections of another "partial road section" instance if they form a "gap" in between. To decide whether a and b form a gap, we check whether they are close to each other and can fit in a line or a smooth curve. It is obvious the conclusion can only be drawn with a degree of confidence. The results produced by all kinds of image processing algorithms have varied error rates, especially when applied in different contexts.

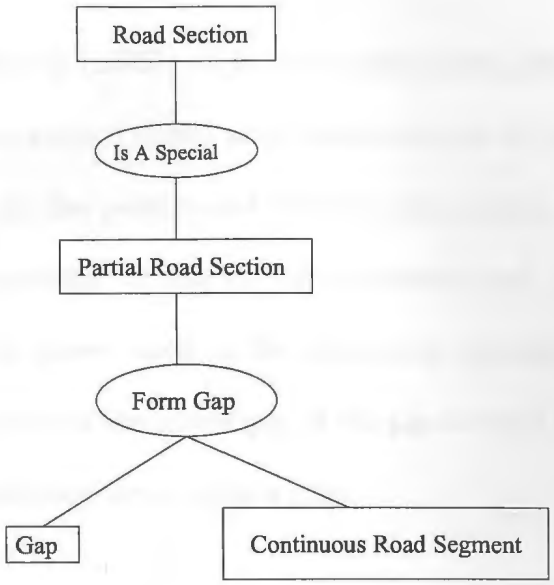


Figure 7.15: Modeling broken line features.

A certainty factor can be attached to some object and relation instances, and later be used in the parsing process. The confidence levels for the existence of certain objects and relations can be determined by their checking algorithms or by special evaluation algorithms based on the current parsing context or commonsense knowledge. Take a look at production rule 9 of $\mathcal{CMG}$ (see Appendix C). If a bridge instance b (type BRIG) is shifted in with a high confidence level, the road segment instance v (type T_SRS) shifted in later will also be assigned a high confidence level. This stems from the fact that if a bridge is present, there must exist connected road segments at both ends of the bridge.

The uncertainty issue can also be addressed through the syntax of map grammars. For example, a special confidence symbol is introduced into the grammar production

shown below:

PRS[p:prs] ::= PRS[u:prs] T_SRS[v:srs] p_not_in_prs[u] p_form_gap[u,v] p_conf_gap[u,v]

The symbol p_form_gap[u,v] is used to determine whether two road segments, u and v, form a gap based on the position and direction information. p_conf_gap[u,v] uses the commonsense knowledge to evaluate the confidence level. Since the confidence factors are set by the parser based on the parameters generated by the constraint symbol checking algorithms like p_form_gap. If the gap between two road segments is bigger, a smaller confidence factor value is given.

7.10 Discussion

The disadvantages of the existing rule based approaches are as follows:

- Many rules intend to represent commonsense knowledge and are formed through intuition. Sometimes they can be either too general or too specific. It is not easy to verify whether they are incorrect.

- They may fail to handle unusual situations. Since rules are experimental and heuristic knowledge directly obtained from real world experience, we may not be aware of certain exceptions.

The disadvantages of the existing semantic based map interpretation approaches are as follows:

- So far they are usually applied on utility maps or cadastral maps that are characterized by less complex features than topographic maps.

- They concentrate on the structural objects and their *part/part-of* relationships, but contribute less to reasoning processes. Therefore, in terms of the utilization of the explicitly represented knowledge, these approaches usually only present very general reasoning strategies, such as goal-driven (top-down) and data-driven (bottom-up) search strategies.

A syntax based approach for the interpretation of maps is presented. The advantages of the proposed syntactic method are as follows:

- It aims at representing structural knowledge, as well as knowledges that support reasoning processes.

- Knowledge of the recognition of object instances and their relationship instances, as well as knowledge of reasoning processes, is formalized as a grammar like representation, which is easier for analysis and verification.

- It provides a system developer the ability to easily enhance and modify the grammar. As a result, many other supporting analysis and interpretation utilities need not to be changed. Since knowledge engineering is an evolving process, the ability to modify is very important.

- It is possible to represent a large set of complex map phenomena with a finite number of concepts, since recursive grammar rules enable us to achieve this.

Chapter 8

Experiments and Results

This chapter presents the results obtained from the experiments that are carried through on test maps. Details of the methods utilized to acquire and preprocess the experimental data are given, followed by the description of the map parsing procedures performed on the data, and the analysis of the results.

8.1 Data Acquisition and Preprocessing

To demonstrate the feasibility of the methodology proposed in this work, experiments were carried out by applying the prototype map understanding system on a number of city or regional maps. These maps comprise the following map objects: road networks, river networks, route number symbols, bridges, location symbols and characters. An example of such a map used in the experiments is shown in Figure 8.1. These city or regional maps are stored in TIFF format.

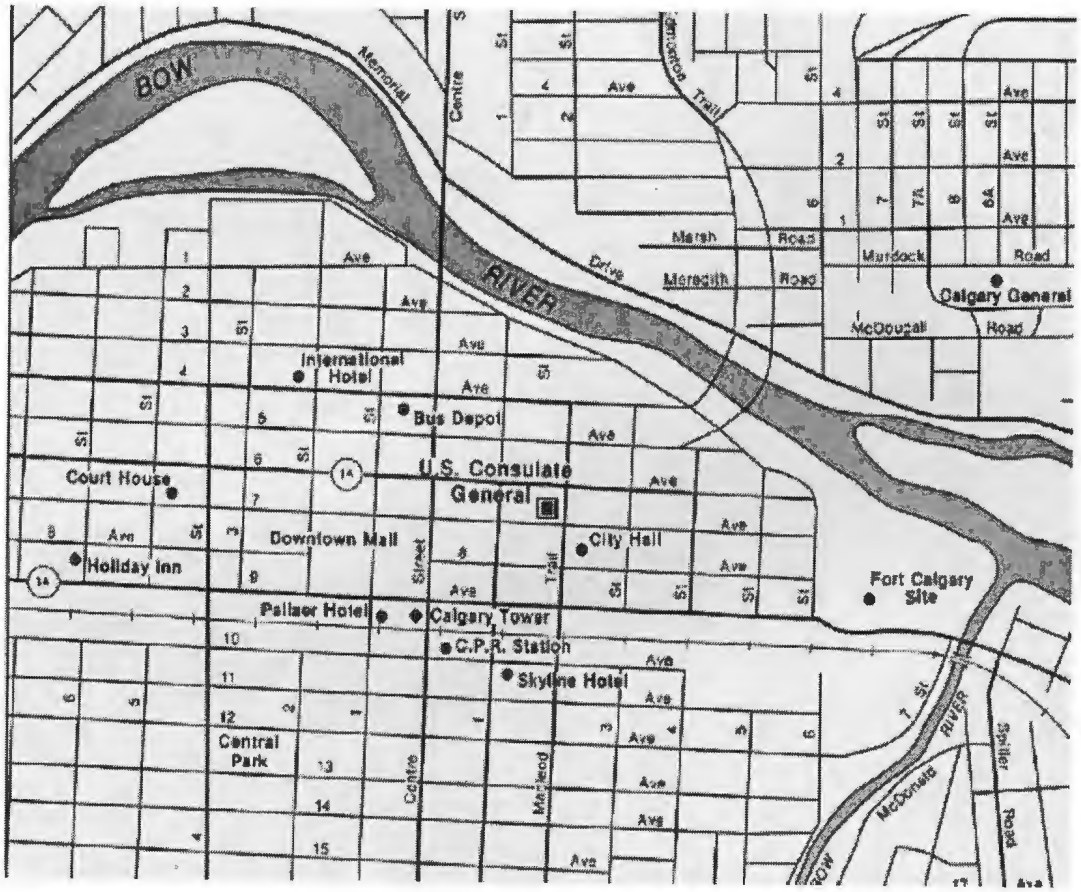


Figure 8.1: City map of Calgary, Canada. U.S. Department of State 1988.

Preprocessing operators or low-level image processing algorithms are then applied to each scanned image. The primary purpose of preprocessing is to remove noises and non-linear features such as characters. Another important goal is to vectorize the map image. The map understanding system is not able to work directly on raster format images, because skeletonized image layers present important topologic and geometric information which is not readily available in images. In addition, vectorization is of great significance in terms of reducing the amount of data to be stored and processed. Since it is considerably easier for the inference modules to manipulate and reason

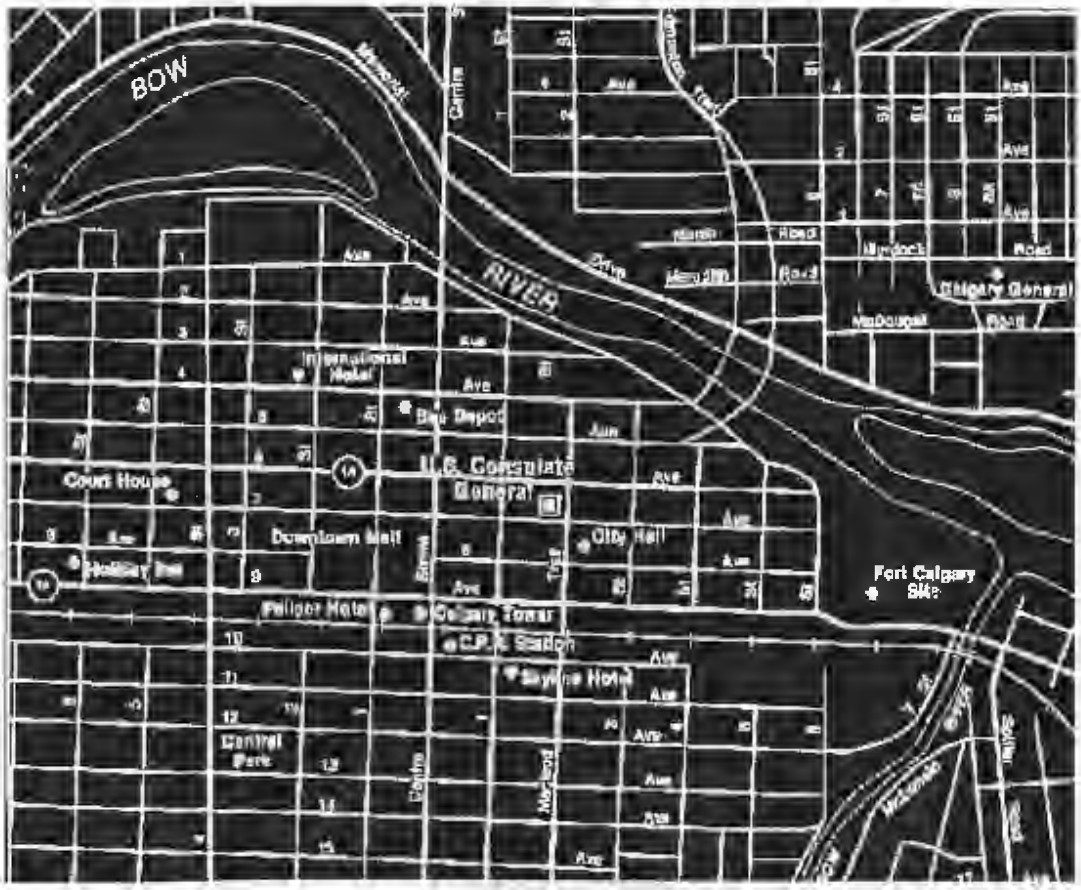


Figure 8.2: Image layer of linear features.

on vectorized data, vectorization can greatly improve the performance of the map understanding system. Software packages Khoros 2.2.0 and Grass 5.0.1 are used for preprocessing.

Image processing algorithms such as color segmentation, adaptive threshold, edge detection and mathematical morphology operations are adopted first to decompose the original color or graylevel map into a number of image layers. For example, the image layer with road and river bank features shown in Figure 8.2 is obtained by applying the k-means clustering algorithm (for line feature extraction) and median

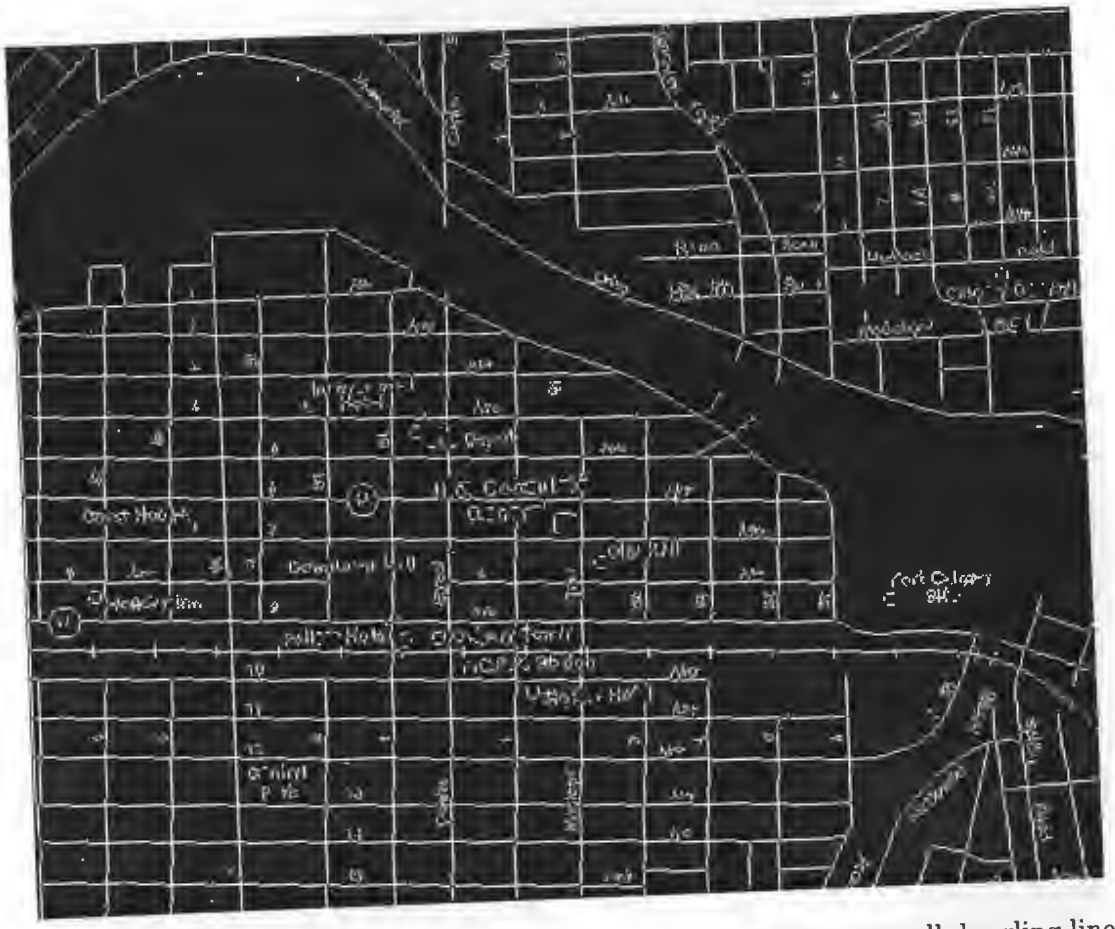


Figure 8.3: Skeletonized image layer of linear features with many small dangling line segments.

filtering operation (for noise removal). Sometimes dilation and erosion operations are applied alternatively and repeated a number of times in order to separate line features from other point features.

The following steps are taken to obtain the vectorized image layers with road features.

1. Use histogram analysis and dynamic thresholding to separate white color from other graylevels. As a result, an image layer with only road and river features

is obtained.

2. Separate river features from roads by applying mathematical morphology algorithms to obtain an image layer with only road features and characters.

3. Perform the thinning algorithm to reduce features to one pixel wide, and vectorize the thinned image layer.

4. Remove short and dangling line segments. Noises caused by isolated areas are almost eliminated. Although noises resulted from characters are already short fragmented line segments at this stage, this step cannot remove them because characters usually touch road lines.

5. Use mathematical morphology operators to remove those thinned lines with a width of one pixel.

6. Subtract the result of step 5 from that of step 4.

7. Go through the thinning and trimming operations again.

After step 3, an intermediate result (see Figure 8.3 for an example) with a lot of small dangling lines is obtained. After all seven steps are carried out, the image layer, depicted in Figure 8.4, shows much less dangling line segments.

Each image layer can be viewed as a collection of one or more types of map objects. Further image processing or interpretation algorithms can easily access and manipulate objects in image layers without complex analysis. In the map parsing

Figure 8.4: Skeletonized image layer of road features where small spurs are trimmed.

process, the information that forms the primitive collection does not come from just one image layer, but from all image layers obtained. Both vector and raster format image layers will be utilized. Since each image layer focuses on revealing a map image from a certain aspect, there will be times in the reasoning process when information from more than one layer needs to be referenced to provide complementary details to the image layer currently under processing. Take the vector format image layer shown in Figure 8.4 as an example. To make up for the gaps we meet in the line tracking process, we need to decide whether two line segments are actually part of

one road. One method is to use image layer in Figure 8.2 as a condition layer, which means that any filled gap has to happen in areas where pixel values are 1s in the condition layer. Therefore, one important task of the map grammar is to express the control knowledge when referring to a condition layer.

8.2 Map Parsing and Results

A map grammar $\mathcal{CMG}$ (see Appendix C), which is obtained by making a few extensions to $\mathcal{MG}$ in Chapter 7, is used for the experiments. The main focus is to see how the map grammar directs the interpretation system to handle various situations in the process of extracting the road network. River networks are not considered in the current experiments.

The primitive collections are obtained using the preprocessing method described in the previous section. An assumption is made that existing algorithms and methods (see [33, 34, 35, 36, 49, 51, 52, 57, 82, 83, 91]) are available to extract and recognize geometric shapes of point symbols such as route numbers and bridge symbols. Due to the difficulties and the tremendous workloads involved in obtaining and customizing those existing algorithms, these point symbols are treated as already correctly recognized in the experiments, and are manually indexed and placed in the primitive collection.

To evaluate and analyze the experimental results, a mechanism that contains certain evaluation criteria is employed. In our experiments, a standard **OIR** graph,

called ***target*** **OIR**, is assigned to the map image to be processed. Such a standard **OIR** is created based on human judgment. It serves as a structured description of certain map objects and their relations, which is considered to be the desired output of the map parsing system. The result produced by the system, which can be represented by an **OIR**, is compared against its target **OIR**. If the obtained result matches the target **OIR** by a certain percentage, it can be considered satisfactory. Therefore, the key is to find out those wrongly classified map objects and relations.

In this section, experimental results of some typical test maps will be presented. These maps include the following: the regional map of Denver (see figure E.2), the city map of Ottawa (see figure E.4), the city map of Halifax (see figure E.6), and the city map of Calgary (see figure 8.1). The numbers of the primitive objects in the maps are reported in Table 8.1.

map	Denver	Ottawa	Halifax	Calgary
size	712×480	493×589	1002×478	761×604
nodes	707	476	302	1060
line fragments	770	544	439	1151
route number symbol	3	0	0	2

Table 8.1: Primitives in the map used in experiments.

In the context of map parsing, some low-level object instances may be the building blocks of many high-level concepts, either directly or indirectly. For example, the correct recognition of road segments and how they relate to other point symbols are essential to the extraction of a whole road network. To measure errors in each of the

experiments, we keep track of the following objects:

- the number of individuals wrongly classified,
- the number of individuals missed,
- the number of noisy individuals removed, and
- the number of individuals wrongly added.

In the process of recognizing road networks, different scenarios of broken lines and dangling lines have to be handled. The grammar can guide the parser to deal with various situations based on the context. Assume, at a certain parse state, p_form_gap is chosen as the type of the possible next token. The selector is instructed to call a function to check whether the immediately previous two line segments shifted in belong to one road section. Usually the criteria to examine whether two line segments s_1 and s_2 form a gap are defined as follows: (1) one end point of s_1 is close to an end point of s_2, and (2) s_1 and s_2 approximately fit into a curve. The threshold we specify to check the closeness is 15 pixels. A curve-fitting algorithm based on Levenberg-Marquardt nonlinear regression [24] is employed to analyze the direction of a line segment. A number of discrete points are taken from the line segment and passed to the curve-fitting algorithm. As a result, a third degree polynomial is generated and a correlation operation is performed to see whether the other line segment fits in the curve defined by the polynomial. In the experiments, we also consider the scenario when road lines are fragmented by gaps larger than the predefined threshold. In this

case, we will try to extend a line segment following its curve direction as much as possible until it meets the other line segment.

During the parsing process, a number of small dangling line segments may be encountered. They should be discarded because commonsense knowledge suggests that road sections are usually represented by relatively long lines and connected to road networks. In order to handle such a situation, some special semantics are assigned to symbol **is_fullprn** to treat a partial road network composed of very short road sections:

IF *the partial network does not have any more unrecognized road section, and the sum of all its road sections does not exceed a threshold*

THEN *discard the short line segments.*

From the above discussion, it can be seen that the map grammar is especially appropriate for fitting the abstract representation of relations among constructing primitives into the interpretation process. It is concluded that the control mechanism is very flexible. When some new situations need to be addressed, we just need to designate c-symbols and m-symbols that represent the constraints and the participating concepts and put the symbols in the appropriate places in the grammar. In general, the system separates declarative and procedural knowledge clearly and makes reuse and evolution of knowledge much easier.

Table 8.2 shows the results produced by the understanding system. The recognition of the features listed in the table is critical to showing the correct topologic structure of the road network.

map	Denver	Ottawa	Halifax	Calgary
no. road network	1	1	1	1
no. road section	407	208	317	1093
no. road intersections	327	152	242	406
no. route symbol	3	0	0	2
no. dangling line segments removed	11	27	41	26
no. broken road section repaired	5	8	3	31

Table 8.2: Recognized instances.

Due to the complexities associated with map phenomena, some errors occur in the process of map parsing. In one of the experiments, a portion of the color map of Denver was used. The color segmentation method described in Chapter 3 was utilized to extract the red color layer, followed by application of the preprocessing operations presented in the previous section. Unfortunately, some of the text features in this map are also drawn in red color. These texts happen to be very close to each other and to the road lines. In addition, the size of the texts is relatively larger. This means a number of dilation operations are needed to fill the space in the texts to make them solid areas. However, too much dilation will cause other features that are close to one another become connected. We chose less aggressive dilation operations to avoid this problem. As a result, some of the texts were not effectively removed by the mathematical morphology operations and were misclassified as small road segments. The two parallel red lines that depict the portion of the highway in this map are too close that at some locations they touch each other. These touching points will be taken as road intersections. Therefore the dilation and erosion operations are

performed alternatively for two times to fill the gap between the two highway lines and to make it appear like one single line. During this process, some fragmented short lines in other parts of the map are lost. Because of the above mentioned factors, the error rate of misclassified line segments was higher, especially in the downtown area, where different line and text features overlap with one another. The misclassified line segments caused by the texts are especially short in size (mostly < 10 pixels). Each text symbol creates a number of such segments in the small area it covered. This is why the misclassified line segments for the map of Denver make a high percentage of the total number. The result may be improved by applying rules that specially treat such line segments. For example, short line segments that display sharp curves or appear twisted can be seen as the remnants of the texts and be removed. Map grammar symbols for special constraints can be designed to capture the geometric features of these line fragments. It can be observed that the error rates for the other test maps were much lower. This may be attributed to the fact that the low level primitive extraction algorithms are more effective on these maps. Shown in Table 8.3 are the line features that are wrongly recognized.

map	Denver		Ottawa		Halifax		Calgary	
		%		%		%		%
missed road section	14	3.44	3	1.44	12	3.79	37	3.39
missed road intersection	19	5.81	5	3.29	3	1.24	49	4.48
unrepaired broken line segment	12	2.95	4	1.92	7	2.21	12	1.14
wrongly added line segment	73	17.94	1	0.48	6	1.89	14	1.28

Table 8.3: Wrongly recognized individuals.

Many reasons may lie behind the erroneous recognition. It is not possible to eliminate all such errors. The following are the factors that directly contribute to the problem:

1. Information loss caused by preprocessing operations. The mathematical morphology operators used to separate point symbols of different sizes may leave gaps on line segments. In most cases, these gaps can be repaired by some special error correcting algorithms. However, sometimes the information about line segments is lost in the process of vectorizing and trimming. Many of the road sections and their associated intersections are lost along the map boundaries.

2. The error correcting algorithms are not sophisticated enough to deal with severely distorted line segments. These algorithms are devised to deal with various scenarios involving distorted line segments resulting from the vectorization process. Even if well-tuned thresholds and adaptive measures are used, it is not easy to capture all the unpredictable behaviors of distorted features. For example, the gap filling algorithms are not able to deal with those broken road sections with big gaps.

3. Interference among different types of map objects. Digits and characters that touch or are close to road sections may cause unwanted small line segments, which are often incorrectly taken as road sections. This is because the mathematical morphology methods cannot remove all such digits and characters. If

aggressive dilation and erosion operations are used to eliminate the characters, the nearby road sections will be badly fragmented.

It is obvious that the places on maps where errors take place involve very small line segments. Although on many occasions they do not affect the overall parsing result very much, sometimes they do cause the incorrect extraction of certain significant road sections. The following measures may be taken to improve the recognition results: (1) applying more advanced low-level image processing methods to obtain better preprocessing results, (2) using refined commonsense knowledge in the parsing process, (3) using post-parsing editing tools to correct the parsing results. The reconstructed road networks of the test maps based on the parsing results are displayed in Appendix E.

The limitation of the current implementation is its dependency on the availability of high quality low-level image processing algorithms. The map parser attempts to make use of detailed attributes of primitives and their relations. Many existing algorithms are not readily available for the map understanding task. For example, it is hard to obtain an algorithm that decides whether two line segments are parallel in the context of geographical maps. The highway route lines on maps that are considered "parallel" are actually not parallel in terms of geometry definition. At some points, they may be very close to each other, but at other points, they are far apart from each other. Under many such situations, the domain knowledge has to be used. However, this mean that map objects have to be examined in the context of a much larger area

and in a more accurate manner. This in turn means the performance may become poor. The chance of accumulated error may become bigger as well because of the large number of objects involved when confirming a single constraint.

The main focus of the current implementation is the recognition of point and line features on maps, because the point and line features have already possessed those typical abstractions and constraints to be dealt with in a map understanding system. The proposed method can also be directly applied to recognize region features such as lakes, recreation parks, and woods areas. In general, region features can be viewed as composite features that are comprised of a number of point, line, or other simpler region features. For example, a recreation park can be represented by a polygon with a number of road and point symbols inside. As long as its properties and relations with other features are identified and specified, a region feature can be represented using DL concepts and roles and thus be processed by the map understanding system.

Chapter 9

Conclusions

Representations such as geographic maps have played an indispensable part in our daily life and will continue to be the primary tool for obtaining knowledge of a region or an area. Meanwhile, computers and Internet have become extremely popular during the past decade. With efforts continuing to be made to improve their performance and to lower the cost, it can be expected that in the near future their popularity will increase dramatically. A large collection of topographical maps, just like other text based contents, constitutes a large part of the information that is expected to be processed or stored by computers, to be transfered across the Internet, and to be accessed by various users for numerous applications. As an analog format of expression, geographic maps encode a vast range of spatial information, which is very difficult to decipher automatically by a computer system. It will be immensely valuable to develop a map understanding system that is capable of discovering heretofore implicit map features and relationships. High-level map understanding, which attempts to

describe map phenomena declaratively, is in particular a research area that needs a great amount of effort.

9.1 Review of The Research

In this dissertation, a novel approach for color map segmentation was presented. Color segmentation is a very important low-level image analysis method. Due to the fact that colors are one of the most prominent attributes to distinguish map features, successful color segmentation is crucial for further map image interpretation. Since the pixel values of areas drawn with the same color tend to cluster around a center, any statistical clustering method can correctly categorize most of the pixels. However, one key issue is how to capture the pixel variances. The accuracy of color segmentation depends on whether those "gone astray" pixels are also correctly classified. Unfortunately the more detailed an area is, the more varied the pixels are, and the more fine-tuned classification is needed. The pixels of an area having relatively high density will be much more difficult to classify because different map features may interweave with one another and the variation of pixel values becomes more complicated. Usually color segmentation is regarded as a clustering problem without taking into consideration the physical process that causes the pixel variations. To catch the pixel variation over the boundary or overlapping areas of colors, a physical reflection model was proposed. It characterizes the relationship of factors that may affect the resulting color image, such as a paper surface, transparency, and illumination envi-

ronment. Two heuristics, H.1 and H.2, were discovered based on the reflection model. H.1 suggests that pixels in an area where two colors overlap distribute within a narrow wedge shaped space. H.2 indicates that pixel values along the boundary areas of two colors approximately obey a linear distribution. A fuzzy neural architecture with self-adjustment components was proposed for color map segmentation. Based on the two heuristics, two types of self-adjustment components were introduced to capture pixel variations in overlapping and boundary areas. They are capable of adjusting sample pixels dynamically among different clusters in neural network training. A color map segmentation system based on the proposed method was designed and implemented.

In this dissertation, a methodology for high-level map understanding was also presented. Map understanding is considered to be a process that converts a map from its low-level format (raw map) into an accurate and meaningful representation of map objects and their relationships. The terms that are used to describe map objects and relations should be similar or close to those we use in our daily life and conform to human beings' intuitive way of thinking. The proposed methodology is based on a formal language called Description Logics, which is particularly suitable for representing knowledge about individuals, classes of individuals (concepts), and relationships (roles). Furthermore, the semantics and the reasoning mechanism were also provided. This method is superior to traditional methods in that it is possible to represent knowledge based on formal theories, thus it can prove consistency and establish precise mapping rules that automatically verify a representation. In

Chapter 4, we presented the essence of conceptual modeling for domain knowledge of maps and knowledge representation using Description Logics. Later we introduced $\mathcal{GALC}(\mathcal{D})$, a DL system that extends $\mathcal{ALC}(\mathcal{D})$ to have roles representing n-ary relations. In Chapter 6, the theory of Description Logics was applied to the task of map understanding. Having established the method to formalize knowledge representation, we turned to discuss what kind of reasoning mechanism can effectively and efficiently make use of the formalism. We defined that the map understanding process aims to build an object-instance-relation graph, which is a representation for map object instances and their relations. A special grammar, called map grammar, is derived from the DL based knowledge representation. In the map grammar, we represent not only the structural information, but also knowledges about interpretation processes. A map grammar parsing algorithm, which works on a multiple path stack (MPS), was presented. Therefore, we have offered a viable approach to build a map understanding system.

9.2 Future Research

This dissertation studies a number of aspects of the establishment of a methodology for map understanding. There are a lot of potential future research directions for this research. Some interesting problems that need further investigation are given.

- A knowledge engineering environment that enables knowledge developers to create, edit, and verify Description Logics representations (concepts, roles, axioms)

using visual design aid tools.

- Extensibility. Very often in knowledge engineering, due to the lack of understanding of the domain, an iterative refining process may be necessary. For instance, rewriting some of the axioms may cause the semantics of other axioms to be changed, or sometimes even cause conflicts. The study of semantics evolution therefore is very important.

- Reuse of knowledge. Different types of geographical maps may look very different in detail. There exist, however, intrinsic similarities among the ways in which map features are represented. It would be very useful to study and develop a methodology that borrows pieces from the conceptual model of one type of map and uses them in the design process of another map type.

- Taking into consideration more complex map objects and relationships would help us deal with a wider range of map phenomena and build better map understanding systems.

9.3 Other Applications

The methodology introduced in this dissertation is not unique to understanding of geographical maps. In this section, several other application areas that can adopt the methodology presented in this work are listed.

1. Automatic Image Annotation.

To achieve highly efficient storage and retrieval of image format data, the technique known as automatic image annotation is used to label certain features on images with names. Thus queries in terms of keywords, texture, or some structural descriptions can be formed to retrieve information from a very large image collection.

2. Video Encoding

   The MPEG-4 video compression standard specifies the ability to code semantically separate objects independently to achieve high compression rate. This requires extraction of meaningful objects from raw natural video sequences.

3. Automatic Image Scene Interpretation.

   As one branch of computer vision, automatic image scene interpretation or picture interpretation is one of those research topics that interpret an image once primitive level objects are recognized [28].

Appendix A

Description Logics

A.1 Definitions of Concrete Domains, Concepts, and Roles

Definition A.1 *A* **concrete domain** $\mathcal{D}$ *is a pair (dom($\mathcal{D}$), pred($\mathcal{D}$)), where dom($\mathcal{D}$) is a set of elements called domain, and pred($\mathcal{D}$) is a set of predicate names. Each predicate name P is associated with an arity n, and an n-ary predicate $P^{\mathcal{D}} \subseteq dom(\mathcal{D})^n$. A concrete domain is called admissible iff (1) pred($\mathcal{D}$) is closed under negation and contains a name for dom($\mathcal{D}$); and (2) the satisfiability problem for finite conjunctions of predicates is decidable.*

Note that pred($\mathcal{D}$) is required to be *closed under negation*, i.e., if P is an n-ary predicate in pred($\mathcal{D}$), then there has to exist a predicate $\overline{P}$ in pred($\mathcal{D}$) such that $\overline{P}^{\mathcal{D}} = \text{dom}(\mathcal{D})^n - P^{\mathcal{D}}$. The next definition shows how to combine two disjoint domains.

Definition A.2 *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be admissible concrete domains with pred($\mathcal{D}_1$) = $\{P_{1,1}, ..., P_{1,n_1}\}$ and pred($(\mathcal{D}_2)$ = $\{P_{2,1}, ..., P_{2,n_1}\}$ such that dom($\mathcal{D}_1$) $\cap$ dom($\mathcal{D}_2$) = $\emptyset$. Then $\mathcal{D}_1 \oplus \mathcal{D}_2$ can be constructed as follows:*

- *Dom($\mathcal{D}_1 \oplus \mathcal{D}_2$) = $\mathcal{D}_1 \cup \mathcal{D}_2$*
- *the predicates of $\mathcal{D}_1 \oplus \mathcal{D}_2$ are*

$$Q_{1,1}, ..., Q_{1,n_1}, \widehat{Q_{1,1}}, ..., \widehat{Q_{1,n_1}}$$

$$Q_{2,1}, ..., Q_{2,n_2}, \widehat{Q_{2,1}}, ..., \widehat{Q_{2,n_2}}$$

where the predicates are defined by

$(x_1, ..., x_n) \in Q_{i,j}$ iff $(x_1, ..., x_n) \in P_{i,j}$, and

$(x_1, ..., x_n) \in \widehat{Q_{i,j}}$ iff $(x_1, ..., x_n) \in \overline{P_{i,j}}$ or there is a k such that $x_k \in dom(\mathcal{D}_{\mu(i)})$,

where $\mu(i) = \begin{cases} 0 & \text{if } i = 1, \\ 1 & \text{if } i = 0. \end{cases}$

Baader and Hanschke [38] show that the combination of admissible domains is still admissible.

Now we are ready to define the conceptual language $\mathcal{ALC}(\mathcal{D})$.

Definition A.3 *(concept terms and terminologies of $\mathcal{ALC}(\mathcal{D})$)*

Let **C**, **R** *and* **F** *be disjoint sets of* ***concept***, ***role***, *and* ***feature*** *names. The set of concept terms are inductively defined. As a starting point of the induction, any element of C is a concept term (atomic terms). Now let C and D be concept terms, let*

R be a role name or feature name, $P \in pred(\mathcal{D})$ be an n-ary predicate name, and u_1, u_2, ..., u_n be feature chains. Then the following expressions are also concept terms:

1. *$C \sqcup D$ (disjunction), $C \sqcap D$ (conjunction), and $\neg C$ (negation),*

2. *$\exists R.C$ (exists-in restriction) and $\forall R.C$(value restriction),*

3. *$P(u_1, u_2, ..., u_n)$ (predicate restriction).*

Let A be a concept name and let D be a concept term. Then A=D is a terminological axiom. A terminology (T-box) is a finite set $\mathcal{T}$ of terminological axioms with the additional restrictions that (i) no concept name appears more than once as a left hand side of a definition, and (ii) $\mathcal{T}$ contains no cyclic definitions.

The above definition gives the syntax of $\mathcal{ALC}(\mathcal{D})$. Then we define its semantics.

A.2 Definitions of Models and Interpretations

Definition A.4 ***(interpretations and models)***

*An **interpretation** $\mathcal{I}$ for $\mathcal{ALC}(\mathcal{D})$ consists of a set $dom(\mathcal{I})$, the abstract domain of the interpretation, and an interpretation function. The abstract domain and the given concrete domain have to be disjoint, i.e., $dom(\mathcal{D}) \cap dom(\mathcal{I}) = \emptyset$. The interpretation function associates with each concept name A a subset $A^{\mathcal{I}}$ of $dom(\mathcal{I})$, with each role name R a binary relation $\mathcal{R}^{\mathcal{I}}$ on $dom(\mathcal{I})$, i.e., a subset of $dom(\mathcal{I}) \times dom(\mathcal{I})$, and with each feature name f a partial function $f^{\mathcal{I}}$ from $dom(\mathcal{I})$ into $dom(\mathcal{I}) \cup dom(\mathcal{D})$.*

For such a partial function $f^{\mathcal{I}}$ *the expression* $f^{\mathcal{I}}(x) = y$ *is sometimes written as* $(x, y) \in f^{\mathcal{I}}$. *If* $u = f_1, \cdots, f_n$ *is a feature chain, then* $u^{\mathcal{I}}$ *denotes the composition* $f_1^{\mathcal{I}} \circ \cdots \circ f_n^{\mathcal{I}}$ *of the partial functions* $f_1^{\mathcal{I}}$, $\cdots$, $f_n^{\mathcal{I}}$. *The interpretation function, which gives an interpretation for atomic terms, can be extended to arbitrary concept terms as follows:*

Let C and D be concept terms, let R be a role name or feature name, $P \in pred(D)$ *be an n-ary predicate name, and* u_1, ..., u_n *be feature chains. Assume that* $C^{\mathcal{I}}$ *and* $D^{\mathcal{I}}$ *are already defined. Then*

1. $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, *and* $(\neg C)^{\mathcal{I}} = dom(\mathcal{I}) \setminus C^{\mathcal{I}}$,
2. $(\forall R : C)^{\mathcal{I}} = \{x \in dom(\mathcal{I})$; *for all y such that* $(x, y) \in R^{\mathcal{I}}$ *we have* $y \in C^{\mathcal{I}}\}$ *and*

   $(\exists R : C)^{\mathcal{I}} = \{x \in dom(\mathcal{I})$; *there exists y such that* $(x, y) \in R^{\mathcal{I}}$ *and* $y \in C^{\mathcal{I}}\}$,
3. $P(u_1, ..., u_n)^{\mathcal{I}} = \{x \in dom(\mathcal{I})$; *there exist* r_1, ..., $r_n \in dom(\mathcal{D})$ *such that*

   $u_1^{\mathcal{I}}(x) = r_1$, ..., $u_n^{\mathcal{I}}(x) = r_n$ *and* $(r_1, ..., r_n) \in P^{\mathcal{D}}\}$.

An interpretation $\mathcal{I}$ *is a model of the T-box* $\mathcal{T}$ *iff it satisfies* $A^{\mathcal{I}} = D^{\mathcal{I}}$ *for all terminological axioms* $A = D$ *in* $\mathcal{T}$.

Lemma A.1 shows how to obtain equivalent expressions.

Lemma A.1 *Let* $\mathcal{D}$ *be a concrete domain such that* $pred(\mathcal{D})$ *is closed under negation and contains a name for* $dom(\mathcal{D})$. *Assume that this name is* $Top_{\mathcal{D}}$, *let Top be an abbreviation for the concept term* $A \sqcup \neg A$ *where A is an arbitrary concept name.*

Let C, D be concept terms of $\mathcal{ALC}(\mathcal{D})$, R be a role name, f be a feature name, P be an n-ary predicate in pred($\mathcal{D}$), and $u_1, \cdots, u_n$ be feature chains. Then the following transformations preserve the equivalence of concept terms:

1. $\neg(C \sqcup D) \longrightarrow ((\neg C) \sqcap (\neg D))$, $\neg(C \sqcap D) \longrightarrow ((\neg C) \sqcup (\neg D))$, $\neg(\neg C) \longrightarrow C$, $\neg(\forall R.C) \longrightarrow (\exists R.C)$, *and* $\neg(\exists R.C) \longrightarrow (\forall R.\neg C)$.

2. $\neg(\forall f.C) \longrightarrow ((\exists f.\neg C) \sqcup Top_{\mathcal{D}}(f))$ *and* $\neg(\exists f.C) \longrightarrow ((\forall f.\neg C) \sqcup Top_{\mathcal{D}}(f))$.

3. $\neg P(u_1, \cdots, u_n) \longrightarrow (\overline{P}(u_1, \cdots, u_n) \sqcup (\forall u_n.Top_{\mathcal{D}}(f)) \sqcup \cdots \sqcup (\forall u_n.Top_{\mathcal{D}}(f))$.

A.3 Terminological Reasoning

The syntax and the semantics of $\mathcal{ALC}(\mathcal{D})$ provide a basis for a formal representation. An important feature is to deduce new facts from a given T-box. In this section, we will discuss the subsumption problem (also called classification problem), which computes the sub-super relationship between two concepts in a T-box.

Subsumption can be formally defined as follows.

Definition A.5 *Let $\mathcal{T}$ be a T-box and $\boldsymbol{A}$, $\boldsymbol{B}$ be concept names, $\boldsymbol{B}$ is said to subsume $\boldsymbol{A}$ with respect to $\mathcal{T}$ iff $\boldsymbol{A}^{\mathcal{I}} \subseteq \boldsymbol{B}^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$.*

The subsumption problem can be reduced to the satisfiability problem, which can be defined as follows.

Definition A.6 *Let C be a concept name, then C is said to be satisfiable iff there exists an interpretation $\mathcal{I}$ such that $C_{\mathcal{I}} \neq \emptyset$.*

A.4 The Assertional Language

In the previous section, concepts and their relationships are formally defined. The symbols, terms and notation used to denote $\mathcal{ALC}(\mathcal{D})$ do not deal with the properties of individual instances. Instead, they concentrate on the common characteristics of groups of instances (concepts) and what kind of relationship may occur among them. However, as we have discussed earlier, a concept is actually a group of instances each of which has its own properties, while a role is a set of individual relation instances. Therefore, it is necessary to have a modeling language to describe what kind of instances exist and how they are related. In this section, we will show that a DL system with assertional capabilities is such a language. This assertional part of the system uses the concept terms for making statements about parts of a given world. We now show how to integrate a concrete domain into an assertional language.

Let $\mathcal{D}$ be an arbitrary concrete domain. It is known that we have to deal with two different kinds of objects: the individuals of the concrete domain and the individuals in the abstract domain (see **Definition A.4**). The names for objects of the concrete domain will come from a set **OC** of object names, and the names for objects of the abstract domain from a set **OA**.

Definition A.7 ***(assertional axioms and A-boxes for $\mathcal{ALC}(\mathcal{D})$)***

Let **OC** *and* **OA** *be two disjoint sets of object names. The set of all assertional axioms is defined as follows. Let C be a concept term of $\mathcal{ALC}(\mathcal{D})$, R be a role name, f be a feature name, and P be an n-ary predicate name of $\mathcal{D}$, and let a, b be elements of* ***OA*** *and $y_1, y_2, \cdots, y_n$ be elements of OC. Then the following are assertional axioms:*

$a : C, (a, b) : R, (a, b) : f, (a, y) : f, (y_1, ..., y_n) : P.$

An A-box is a finite set of such assertional axioms.

Definition A.8 ***(interpretations and models)***

An interpretation for the assertional language is simply an interpretation for $\mathcal{ALC}(\mathcal{D})$ which, in addition, assigns an object $a^{\mathcal{I}} \in dom(\mathcal{I})$ to each object name $a \in$ **OA**, *and an object $x^{\mathcal{I}} \in dom(\mathcal{D})$ to each object name $x \in$* **OC**. *Such an interpretation satisfies an assertional axiom*

$a : C$ *iff* $a^{\mathcal{I}} \in C^{\mathcal{I}}$, (a, b):R *iff* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$, (a, b):f *iff* $f^{\mathcal{I}}(a^{\mathcal{I}}) = b^{\mathcal{I}}$, $(a, y) : f$ *iff* $f^{\mathcal{I}}(a^{\mathcal{I}}) = y^{\mathcal{I}}$, $(y_1, ..., y_n) : P$ *iff* $(y_1^{\mathcal{I}}, ..., y_n^{\mathcal{I}}) \in P^{\mathcal{D}}$.

An interpretation is a model of an A-box $\mathcal{A}$ iff it satisfies all the assertional axioms of $\mathcal{A}$, and it is a model of an A-box $\mathcal{A}$ together with a T-box $\mathcal{T}$ iff it is a model of $\mathcal{T}$ and a model of $\mathcal{A}$.

The definition shows that we do not require unique names for the objects.

Considering an A-box without a corresponding T-box means that all the concepts names occurring in concept terms are assumed to be primitive.

A.5 Assertional Reasoning

In the following, $\mathcal{A}$ will always denote an A-box, $\mathcal{T}$ a T-box, $\mathbf{C}$, $\mathbf{D}$ concept terms, a, $b \in \mathbf{OA}$ names of abstract objects, and x; $y \in \mathbf{OC}$ names of concrete objects.

An obvious requirement on the represented knowledge is that it should not be contradictory. Otherwise, it would be useless to deduce other facts from this knowledge since logically, everything follows from an inconsistent set of assumptions. However, for a given A-box (or an A-box together with a T-box) it is not necessary to have a model. For example, an A-box containing the axioms $a : C$ and $a : \neg C$, or the axioms $(a, b) : f$, $(a, y) : f$ for a feature name f is contradictory, and thus cannot have a model.

We say that an A-box (an A-box together with a T-box) is consistent *iff* it has a model. Otherwise, it is called *inconsistent.*

For the above mentioned reason it is important to have an algorithm which decides consistency of a given A-box. In addition, it will turn out that such an algorithm can also be used to solve all the other important inference problems, namely subsumption between concepts, satisfiability of concepts, consistency of an A-box together with a T-box, and the so-called instantiation problem.

This last problem is defined as follows. The abstract object a is an instance of C with respect to $\mathcal{A}$ (with respect to $\mathcal{A}$ together with $\mathcal{T}$) *iff* $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models of $\mathcal{A}$ (for all models of $\mathcal{A}$ together with $\mathcal{T}$).

Consistency of —— as well as instantiation with respect to —— an A-box together

with a T-box can easily be reduced to the corresponding problems for A-boxes alone. In fact, one must simply unfold the corresponding T-box, and then replace all defined concept names occurring in concept terms of the A-box by their definitions, in the unfolded T-box.

In addition, the instantiation problem can be reduced to the consistency problem as follows: a is an instance of C with respect to $\mathcal{A}$ *iff* the A-box $\mathcal{A} \cup a : \neg C$ is inconsistent.

Finally, the satisfiability problem for concept terms (and thus also the subsumption problem) can also be reduced to the consistency problem for A-boxes. In fact, C is satisfiable *iff* the A-box $\{a : C\}$ is consistent.

Appendix B

List of Maps Used for Testing

1. Name: Portion of Ontario, Canada. National Atlas of Canada Series, 1981. Source: Queen Elizabeth Library, Memorial University of Newfoundland. Publisher: Natural Resource of Canada. Size: 154 × 136 pixels. Resolution: 300dpi.

2. Name: Portion of Denver, Colorado. Source: University of Texas at Austin Library Online Map Collection. Publisher: US Geological Survey (USGS). Size: 945 × 1024 pixels. Resolution: 200dpi.

3. Name: City Map of Calgary, Canada. Source: University of Texas at Austin Library Online Map Collection. Publisher: U.S. Department of State, 1988. Size: 571 × 453 pixels. Resolution: 150dpi.

4. Name: Portion of Cincinnati, Source: University of Texas at Austin Library Online Map Collection. Publisher: US Geological Survey (USGS). Size: 363 × 274. Resolution: 200dpi.

5. Name: Portion of Boston, Massachusetts. Source: University of Texas at Austin Library Online Map Collection. Publisher: US Geological Survey (USGS). Resolution: 200dpi.

6. Name: Portion of Minneapolis, Minnesota. Source: University of Texas at Austin Library Online Map Collection. Publisher: US Geological Survey (USGS). Resolution: 200dpi.

7. Name: City Map of Ottawa, Canada. Source: University of Texas at Austin Library Online Map Collection. Publisher: U.S. Department of State, 1988. Size: 493 $\times$ 589 pixels. Resolution: 150dpi.

8. Name: City Map of Halifax, Canada. Source: University of Texas at Austin Library Online Map Collection. Publisher: U.S. Department of State, 1988. Size: 1002 $\times$ 478 pixels. Resolution: 150dpi.

Appendix C

Map Grammar $\mathcal{CMG}$

1. M[a:map] ::= RWS[c:rws] is_fullrws[c] RNS[b:rns] is_fullrns[b]

2. RNS[b:rns] ::= RNS[c:rns] RN[d:rn]

3. RNS[c:rns] ::= RN[c:rn]

4. RN[c:RN] ::= PRN[c:prn] is_fullprn[e]

5. PRN[b:prn] ::= PRN[b:prn] RS[r:rs] p_attaching[(b, r):rnh]

6. PRN[b:prn] ::= RS[r:rs]

7. RS[r:rs] ::= PRS[p:prs] full_prs[p]

8. PRS[p:prs] ::= T_SRS[s:srs]

9. PRS[p:prs] ::= PRS[u:prs] BRIG[b:brig] f_barline_of[b:x] p_brig_srs_connecting[u, x] f_barline_of[b:y] p_other_barline[b, x, y] T_SRS[v:srs] p_not_in_prs[v]

p_brig_srs_connecting[v,y] p_form_two_sides[u, x, v, y]

10. PRS[p:prs] ::= PRS[u:prs] T_CITY[c:city] p_touch[u, c] T_SRS[v:srs] p_not_in_prs[u] p_touch[v, c]

11. PRS[p:prs] ::= PRS[u:prs] T_SRS[v:srs] p_not_in_prs[u] p_form_gap[u,v]

12. BRIG[b:brig] ::= T_LNS[l:lns] p_is_barline[l] T_LNS[m:lns] p_is_barline[m] p_form_bridge[l,m]

13. RWS[c:rws] ::= RWS[d:rws] RW[e:rw]

14. RWS[c:rws] ::= RW[c:rw]

15. RW[e:rw] ::= PRW[p:prw] is_fullprw[p]

16. RW[e:rw] ::= PRW[p:prw] RWSEC[r:rwsec] p_attaching[(p,r):rnh]

17. PRW[p:prw] ::= RWSEC[r:rwsec]

18. RWSEC[r:rwsec] ::= T_SRS[t:srs] p_form_railway[t]

The meanings of the grammar symbols are given below.

M —— an instance representing a map scene.

RWS —— an instance representing a set of railroad networks.

RNS —— an instance representing a set of road networks.

is_fullrws[c] —— a predicate representing the fact that no more instances can be added to the set of railroad networks represented by c.

is_fullrns[b] —— a predicate representing the fact that no more instances can be added to the set of road networks represented by b.

RN —— an instance representing an individual road network.

PRN —— an instance representing a partially recognized road network; that is, a road network where some parts are not yet explicitly shown.

is_fullprn[e] —— a predicate representing the fact that the PRN instance e does not have any parts that are not explicitly shown.

RS —— an instance representing a road section.

p_attaching[(b,r):rnh] —— a predicate representing the fact that a road section r is attached to a partially recognized road network b.

PRS —— an instance representing a partially recognized road section.

full_prs[p] —— a predicate representing the fact that no more parts can be attached to the partially recognized road section p.

T_SRS —— a terminal instance representing a simple road section.

BRIG —— an instance representing a bridge.

f_barline_of[b:x] —— a function symbol that designates the barline attribute of BRIG b, which is represented by x.

p_brig_srs_connecting[u,x] —— a predicate representing the fact that a bridge u is connected with a barline x of a simple road section.

p_other_barline[b,x,y] —— a predicate representing the fact that x and y are two different barlines of bridge b.

p_not_in_prs[v] —— a predicate representing the fact that a simple road section v is not part of a partial road section.

p_form_two_sides —— a predicate representing the fact that two simple road sections u and v and two bridge barlines x and y form a road section with a bridge in the middle.

T_CITY —— a terminal instance representing a city symbol.

p_touch[u,c] —— a predicate representing the fact that a partial road section u touches a city symbol c.

p_form_gap[u,v] —— a predicate representing the fact that there is a gap between two partial road sections u and v that in fact belong to one whole road section.

T_LNS —— a terminal instance representing a polyline.

p_is_barline[l] —— a predicate representing the fact that a T_LNS instance can be recognized by a barline, a part of a bridge.

p_form_bridge[l,m] —— a predicate representing the fact that two barlines (l and m) can form a bridge.

RW —— an instance representing a railroad.

PRW —— an instance representing a partially recognized railroad.

is_fullprw[p] —— a predicate representing the fact that no other parts can be added to the partial railroad p.

RWSEC —— an instance representing a railroad section.

p_form_railway[t] —— a predicate representing the fact that a T_SRS instance can

be recognized as a railroad.

Appendix D

Color Segmentation Results

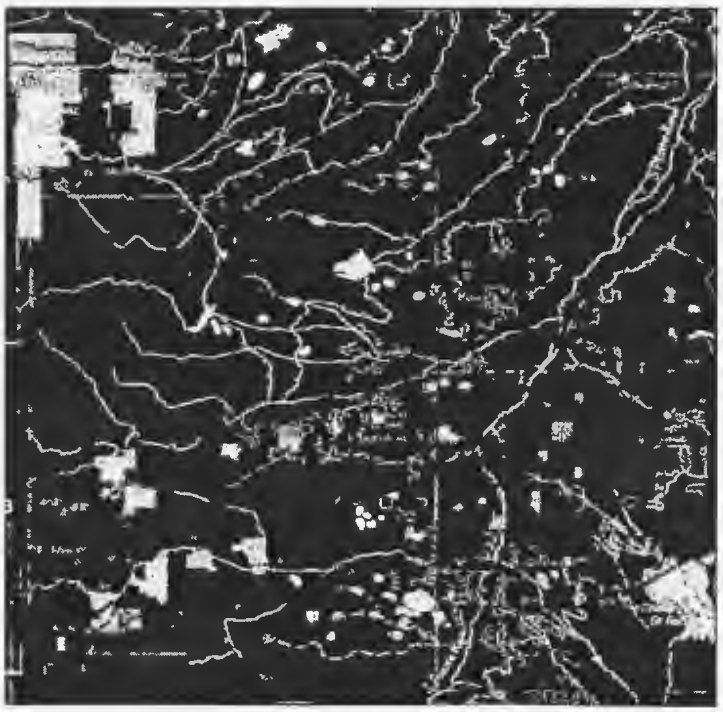


Figure D.1: The blue color layer extracted from the map of Denver using the proposed fuzzy neural network.

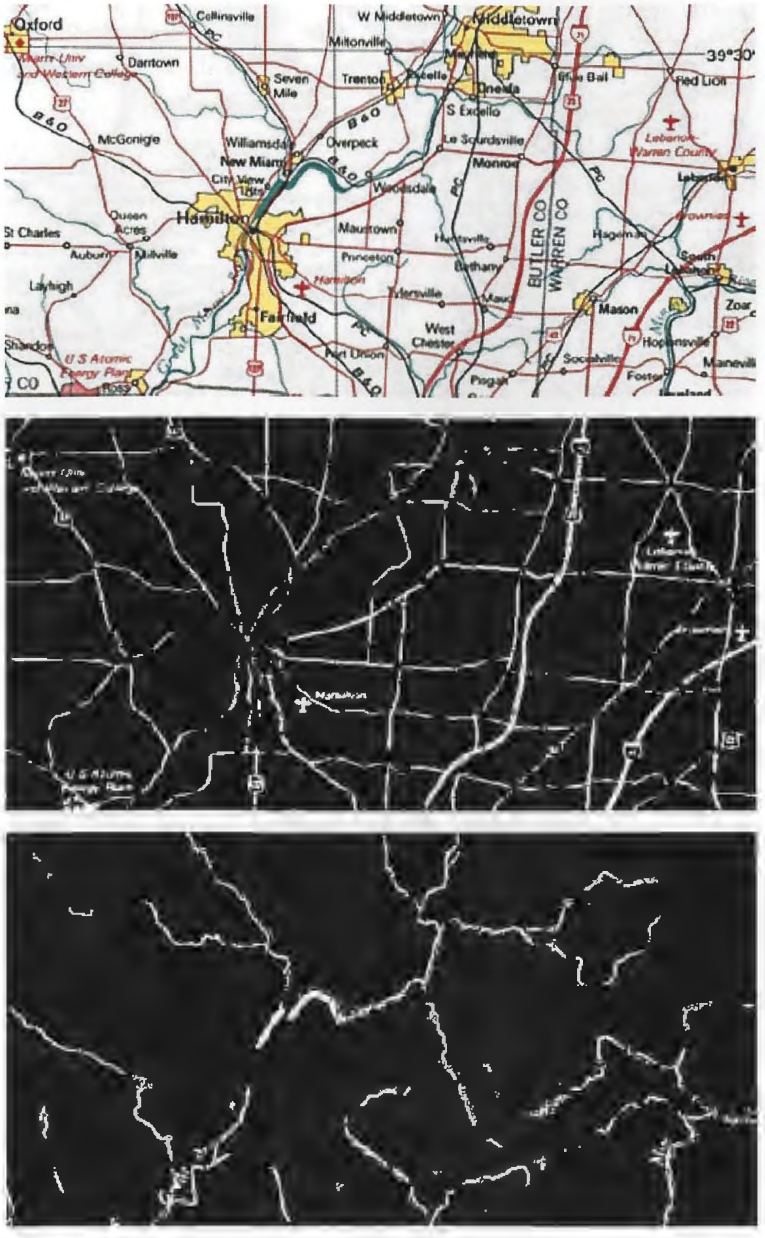


Figure D.2: The same fuzzy neural network used for the map of Denver is applied to the map of Cincinnati without retraining. (a) Original map. Original scale 1:500,000 U.S. National Atlas 1970. (b) Red color layer. (c) Blue color layer.

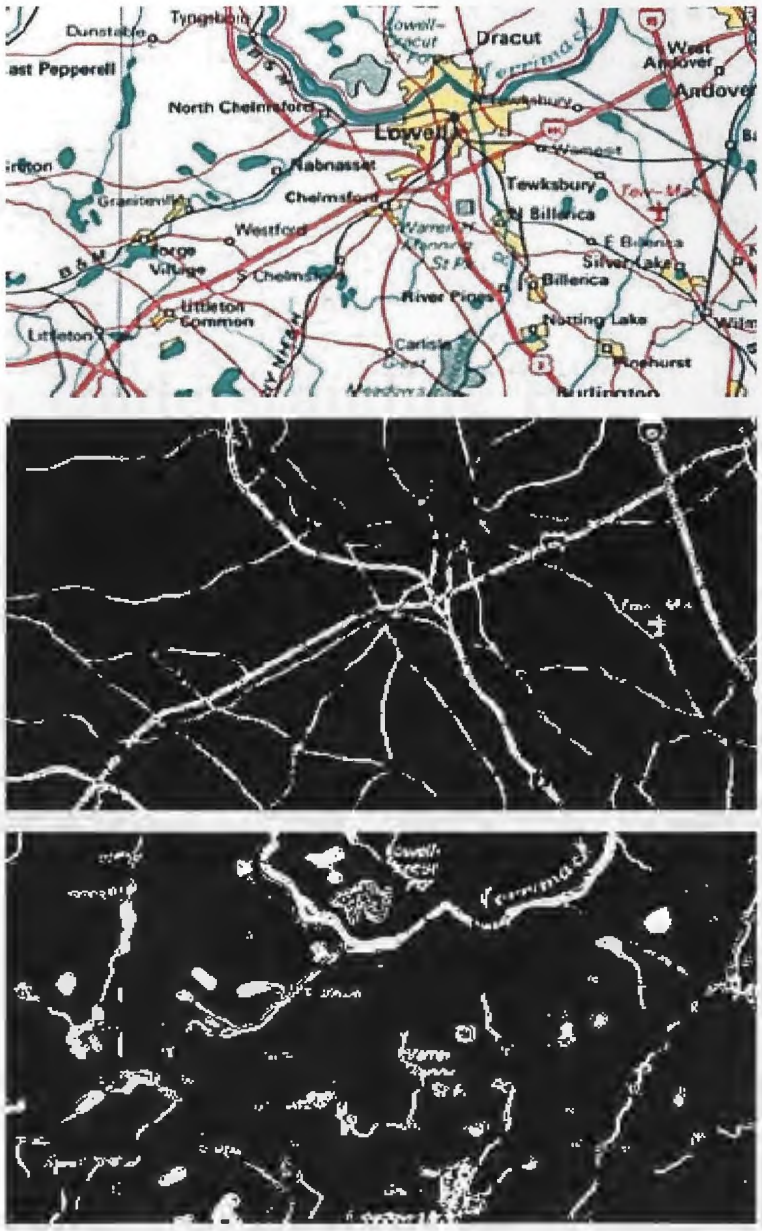


Figure D.3: The same fuzzy neural network used for the map of Denver is applied to the map of Boston without retraining. (a) Original map. Original scale 1:500,000 U.S. National Atlas 1970. (b) Red color layer. (c) Blue color layer.

Appendix E

Map Understanding Results

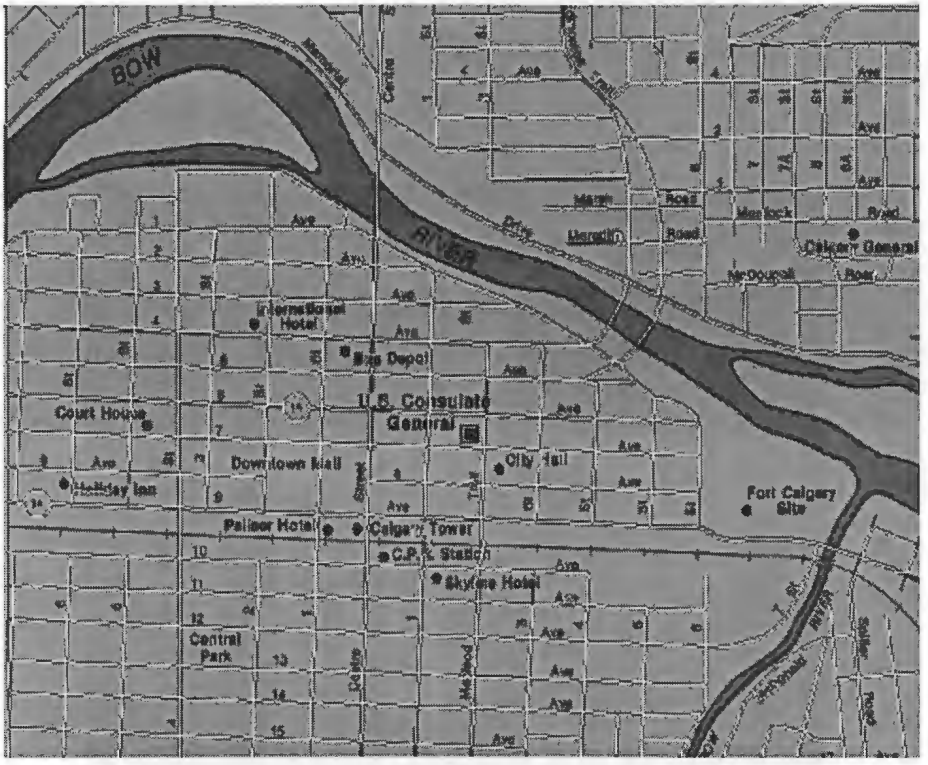


Figure E.1: Extracted road network from the map of Calgary.

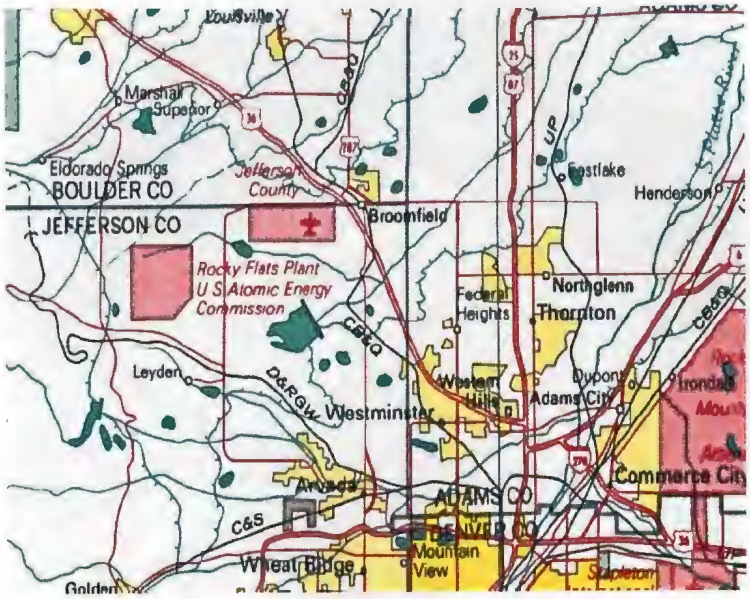


Figure E.2: A portion of the regional map of Denver.

Figure E.3: Extracted road network from the map of Denver.

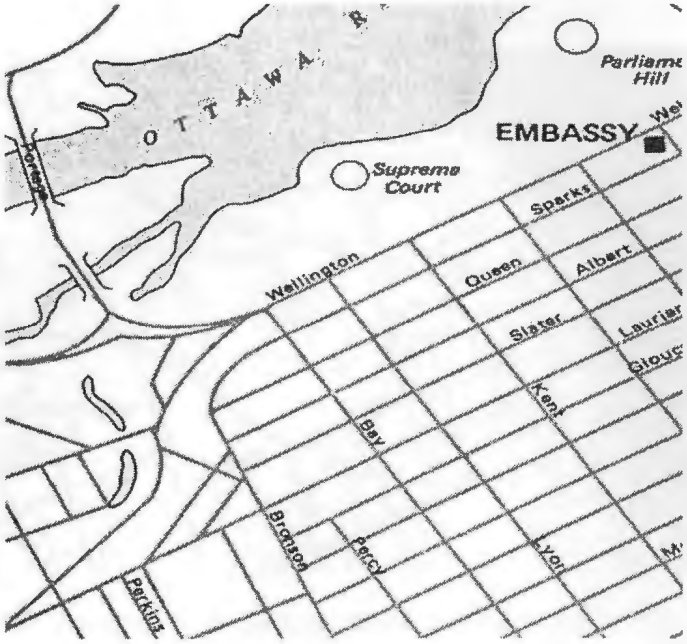


Figure E.4: A portion of the city map of Ottawa.

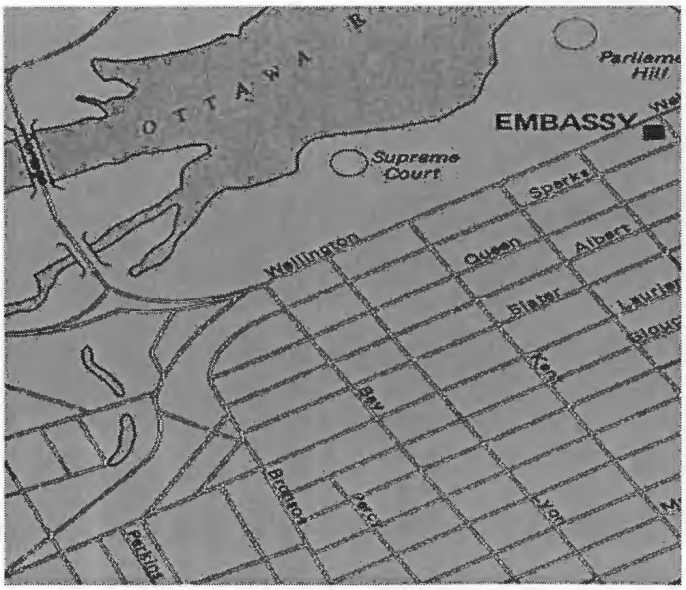


Figure E.5: Extracted road network from the map of Ottawa.

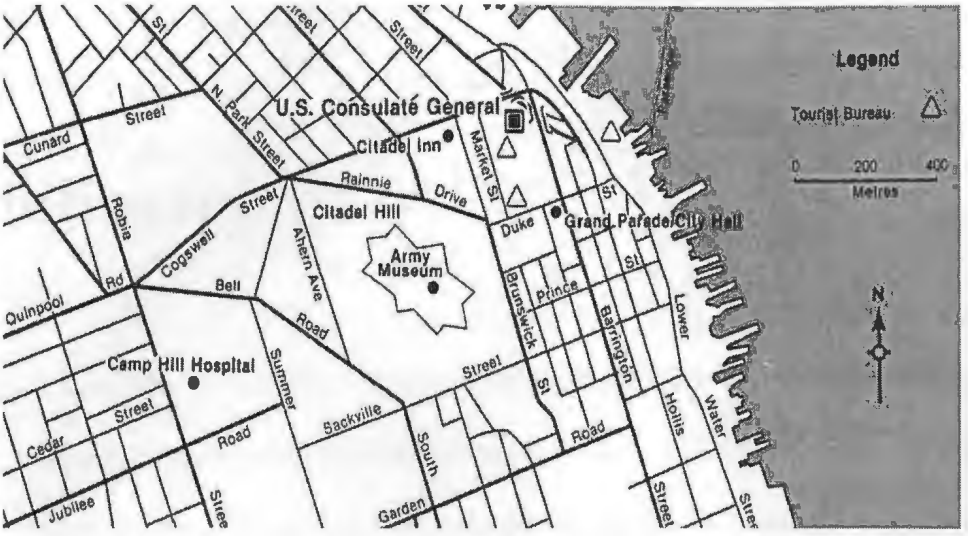


Figure E.6: A portion of the city map of Halifax.

Figure E.7: Extracted road network from the map of Halifax.

Bibliography

[1] L. Lucchese, S.K. Mitra, "Unsupervised Segmentation of Color Images Based on k-means Clustering in the Chromaticity Plane", IEEE Workshop on Content-Based Access of Image and Video Libraries, pp. 74-78, June 22 - 22, 1999, Fort Collins, Colorado.

[2] Y. W. Lim and S. U. Lee, "On the Color Image Segmentation Algorithm Based on the Thresholding and the Fuzzy c-means Techniques", Pattern Recognition, Vol. 23, No.9, 935-952, 1990.

[3] G. Healey, S. Shafer, L. Wolff (eds.), *Physics-based Vision: Principles and Practice*, COLOR, Jones and Bartlett, Boston, 1992.

[4] G. Healey, "Modeling Color Images for Machine Vision," in Advances in Image Processing and Machine Vision, J. Sanz editor, Springer-Verlag, 1996.

[5] T. Uchiyama, M.A. Arbib, "Color Image Segmentation using Competitive Learning", IEEE transactions on Pattern Analysis and Machine Intelligence, Vol 16, No. 12, pp. 1197-1206, December 1994.

[6] Jan Puzicha, Thomas Hofmann and Joachim Buhmann, "Histogram Clustering for Unsupervised Image Segmentation", Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'99), Fort Collins, pp. 602-608, 1999.

[7] C. L. Huang, "Pattern image segmentation using modified Hopfield model", Pattern Recognition Letters, 13 (1999), 345-353.

[8] A. P. Petrov and L. L. Kontsevich, "Properties of Color Images of Surfaces under Multiple Illuminants", Optical Society of America, Vol. 11, No. 10, 2745-2749, Oct. 1994.

[9] B. A. Maxwell and S. A. Shafer, "Physics-based Segmentation: Moving Beyond Color", IEEE International Conference on Computer Vision and Pattern Recognition, 742-749, 1996.

[10] D. Nardi, R. J. Brachman, "An Introduction to Description Logics", Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 5-44.

[11] T. L. Huntsberger, C. L. Jacobs and R. L. Cannon, "Iterative Fuzzy Image Segmentation", Pattern Recognition, Vol. 18, No. 2, 131-138, 1985.

[12] A. Moghaddamzadeh and N. Bourbakis, "A Fuzzy Technique for Image Segmentation of Color Images", IEEE Word Congress on Computational Intelligence: FUZZY-IEEE, Orlando, Florida, June 1994.

[13] Zhong, D. X., Yan, H., "Color image segmentation using color space analysis and fuzzy clustering", Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop Volume: 2, 2000, pp. 624-633 vol.2.

[14] Noordam, J. C., van den Broek, W.H.A.M., Buydens,L.M.C., "Geometrically guided fuzzy C-means clustering for multivariate image segmentation", Pattern Recognition, 2000. Proceedings. 15th International Conference on Volume: 1, 2000, pp. 462-465.

[15] A. Borgida, M. Lenzerini, R. Rosati. "Description Logics for Databases", Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pp. 472-494.

[16] A. Borgida, "Description Logics in Data Management", IEEE Transactions on Knowledge and Data Engineering vol.7, No. 5, October 1995, pp. 671-682.

[17] Ragnhild Van Der Straeten, Miro Casanova, "The Use of DL in Component Libraries - First Experiences", KI-2002 Workshop on Applications of Description Logics ADL'02, Aachen, Germany September 16th, 2002.

[18] Daniela Berardi, Diego Calvanese, Guiseppe De Giacomo, "Reasoning on UML Class Diagrams using Description Logic Based Systems", KI-2001 Workshop on Applications of Description Logics Vienna, Austria September 18, 2001.

[19] Michael Eisfeld, "Model Construction for Configuration Design", KI-2002 Workshop on Applications of Description Logics ADL'02 Aachen, Germany September 16th, 2002.

[20] A. K. Jain and R. C. Bubes, "Algorithms for Clustering Data", Englewood Cliff, NJ: Prentice Hall, 1988.

[21] F. Baader, "A Formal Definition for the Expressive Power of Terminological Knowledge Representation Languages", Journal of Logic and Computation, 6(1): 33-54, 1996.

[22] H. J. Levesque and R. J. Brachman, "Expressiveness and tractability in knowledge representation and reasoning", Computational Intelligence Journal 3, 78-93 (1987).

[23] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi, "Reasoning in Expressive Description Logics", in Handbook of Automated Reasoning, A. Robinson and A. Voronkov(eds), Elsevier Science Publishers (North-Holland), Amsterdam, 2001, pp. 1581-1634.

[24] G. A. F. Seber and C. J. Wild, *Nonlinear Regression*, Wiley, New York, p. 624, 1989.

[25] D. Comaniciu and P. Meer, "Robust Analysis of Features Spaces: Color Image Segmentation", Proc. of IEEE Conf. on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, pp. 750-755, June 1997.

[26] Mark S. Drew, Jie Wei, and Ze-Nian Li, "On Illumination Invariance in Color Object Recognition", Pattern Recognition, Vol.31, No. 8, pp. 1077-1081, 1998.

[27] Lippman R. P., "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, Vol. 4, pp. 14, 1987.

[28] Sandy Dance and Terry Caelli, "On the Symbolic Interpretation of Traffic Scenes", ACCV93 Proceedings of the Asian Conference on Computer Vision, pages 798–801, Osaka Japan, November 1993.

[29] Helmet Mayer, "Automatic Knowledge Based Extraction of objects of the real world from scanned maps", International Archives of Photogrammetry and Remote Sensing(30) 3/2, pp. 547-554, 1994.

[30] A. I. Abdelmoty, M. H. Williams and J M P Quinn, "A rule-based approach to computerized map reading", Information and Software Technology, Vol. 35, No. 10, October 1993.

[31] J. E. Den Hartog, T. K. Ten Kate, and J. J. Gerbrands, "Knowledge-based Interpretation of Utility Maps", Computer Vision and Image Processing, vol. 63, No. 1, pp. 105-117, 1996.

[32] Rik D. T. Janssen, The Application of Model-based Image Processing to the Interpretation of Maps, Ph.D thesis, Delft University of Technology, Delft, the Netherland, 1995.

[33] Fletcher L.A and Kasturi R., "A robust algorithm for text string separation from mixed text/graphics images", IEEE trans. PAMI, 10, pp. 910-918, 1988.

[34] John Shunen Shieh, "Recursive Morphological Sieve Method for Searching Pictorial Point Symbols on Maps", Proc. the 3rd Int. Conf. on Documentation Analysis and Recognition, Montreal, Canada, pp. 931-935, Aug. 14-16, 1995.

[35] Masanori Anegawa, Osamu Shiku, Akira Nakamura, Terumitsu Ohyama, and Hideo Kuroda, "A System for Recognition Numeric Strings from Topographical Maps", Proc. the 3rd Int. Conf. on Documentation Analysis and Recognition, Montreal, Canada, pp. 940-943, Aug. 14-16, 1995.

[36] Hiromitsu Yamada, Kazuhiko Yamamoto, and Masanobu Nakamura, "Increasing the Performance of MAP(Multi-Angled Parallelism) Erosion-Dilation for Feature Extraction through Unary/Binary Operations", IEEE trans. of Information Processing Society of Japan, Vol. 31, No. 6, June 1990.

[37] Karen A. Lemone, *Design of Compilers: techniques of programming language translation*, CRC Press, Inc., 1992.

[38] F. Baader and P. Hanschke. "A Scheme for Integrating Concrete Domains into Concept Languages", in Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI-91, pp. 452-457, Sydney (Australia), 1991.

[39] R. J. Brachman and J. G. Schmolze. *An overview of the KL-ONE knowledge representation system.*, Cognitive Science, 9(2): 171:216, 1985.

[40] Carlo Meghini, Fabrizio Sebastiani, Umberto Straccia, "The Terminological Image Retrieval Model", ICIAP (2), pp. 156-163, 1997.

[41] Masakazu Ejiri, *et al.*, "Automatic Recognition of Design Drawings and Maps", Seventh International Conference on Pattern Recognition, pp. 1296-1305, Montreal, Canada, 1984.

[42] Rangachar Kasturi, "Image-Analysis Techniques for Information Systems", Image Analysis Applications, pp. 127-163, Kasturi R. and M. Trivedi(editors), Marcel Dekker, Vol. 24, 1990.

[43] J. M. P. Quinn, *et al.*, "Knowledge-based Systems for Geographical Information Systems", The Yearbook of the Association for Geographic Information, J. Cadoux-Hudson and D. I. Heyword(eds), pp. 423-430, 1992.

[44] Terry Caelli and David Reye, "On the Classification of Image Regions by Color, Texture and Shape", Pattern Recognition , Vol. 26, No. 4, pp. 461-470, 1993.

[45] H. Yamada, *et al.*, "Directional Mathematical Morphology and Reformalized Hough Transformation for the Analysis of Topographic Map", IEEE Trans on Pattern Analysis and Machine Intelligence, Vol. 15, No. 4, pp. 380-387, 1993.

[46] S. Shimotsuji, *et al.*, "A Robust Recognition System for a Drawing Superimposed on a Map", IEEE Computer, Vol. 25, No. 7, pp. 56-59, July 1992.

[47] Oivind Due Trier and Torfinn Taxt, "Data Capture from Maps Based on Gray Scale Topographic Analysis", Proc. of 3rd Int Conf. on Document Analysis and Recognition, Vol. 2, pp. 923-926, Montreal, Canada, 1995.

[48] Desachy Jachy, "A Knowledge-based System for Satellite Image Interpretation", 11Th International Conf. of Pattern Recognition, Vol. 1, pp. 198-201, 1992.

[49] Yu Zhong, *et al.*, "Locating Text in Complex Color Images", Proc. of 3rd Int Conf. on Document Analysis and Recognition, Vol. 2, pp. 146-149, Montreal, Canada, 1995.

[50] Heutte L., *et al.*,"Two Aspects of Automatic Map Treatment: Road and Texture Extraction", 11th Int. Conf. Of Pattern Recognition, Vol. 3, pp. 109-112, 1992.

[51] Mohamed Kamel, *et al.*, "Binary Character/Graphics Image Extraction: a New Technique and Six Evaluation Aspects", 11th Int. Conf. Of Pattern Recognition, Vol. 3, pp. 113-116, 1992.

[52] David S. Doermann, *et al.*, "Logo Recognition Using Geometric Invariants", IEEE International Conference on Document Analysis and Recognition, pp. 894-901, Japan, Oct. 1993.

[53] H. Yamada, *et al.*, "MAP: Multi-Angled Parallelism for Feature Extraction from Topographical Maps", Pattern Recognition, Vol. 24, No. 6 , pp. 479-488, 1991.

[54] M. T. Musavi, *et al.*,"A Vision Based Method to Automatic Map Processing", Pattern Recognition , Vol. 21, No. 4, pp. 319-326, 1988.

[55] R. Kasturi, *et al.*, "A System for Interpretation of Line Drawings", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 12, No. 10, pp. 978-991, Oct. 1990.

[56] V. Nagasamy and N. A. Langrana, "Engineering Drawing Processing and Vectorization System", Computer Vision, Graphics and Image Processing, Vol. 49, pp. 379-397, 1990.

[57] Huizhu Luo, *et al.*, "Directional Mathematical Morphology Approach for Line Thinning and Extraction of Character Strings from Maps and Line Drawings", Proc. of 3rd Int Conf. on Document Analysis and Recognition, Vol. 2, pp. 257-260, Montreal, Canada, 1995.

[58] S. V. Ablameiko, "A System for Automatic Vectorization and Interpretation of Graphic Images", Pattern Recognition and Image Analysis, Vol. 3, No. 1, pp. 39-52, 1993.

[59] P. Vaxiviere and K. Tombre, "Knowledge Organization and Interpretation Process in Engineering Drawing Interpretation" , Document Analysis Systems, A. Lawrence and Andreas Dengel, Eds, pp. 307-317, World Scientific Publishings, 1995.

[60] Boris Pasternak, "Processing Imprecise and Structural Distorted Line Drawings by an Adaptable Drawing Interpretation Kernel", Document Analysis Systems, A. Lawrence and Andreas Dengel, Eds, pp. 318-337, World Scientific Publishings, 1995.

[61] F. Meyer, "Color Image Segmentation", IEEE Int Conf. Image Processing and Its Applications, pp. 303-306, Veniee, the Netherlands, 1992.

[62] P. Campadelli, "Color Image Segmentation Using Hopfield Networks", Image and Vision Computing, Vol. 15, pp. 161-166, 1997.

[63] Jander Moreira, *et al.*, "Neural-based Color Image Segmentation and Classification Using Self-Organization Maps", *Anais do* IX SIBG RAPI, pp. 47-54, 1996.

[64] Din-Chang Tseng, *et al.*, "Circular Histogram Thresholding for Color Image Segmentation, Proc. of 3rd Int Conf. on Document Analysis and Recognition, Vol. 2, pp. 673-684 , Montreal, Canada, 1995.

[65] Rafael Santos, *et al.*, "Supervised Image Classification with Khoros and the Classify Toolbox—Tutorial Outline", Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology-Iizuka, Fukuoka, Japan.

[66] Kazuhiko Yamamoto, "Recognition of Elevation Symbols and Reconstruction of 3D Surface from Contours by Parallel Method", IEICE Trans. Information and Systems, Vol. E77-D, No. 7, pp. 749-753, July, 1994.

[67] Wei Lu, T. Okuhashi and M. Sakauchi, "A Proposal of Efficient Interactive Recognition System for Understanding of Map Drawings", Proc. of 3rd Int. Conf. on Document Analysis and Recognition, Vol. 2, pp. 520-523, Montreal, Canada, 1995.

[68] T. Tjahjadi, *et al.*, "A Knowledge Based System for Image Understanding", Int. Conf. on Image Processing & Its Application, pp. 88-116, July, 1989.

[69] C. Kim, *et al.*, "Understanding Three-View Drawings Based on Heuristics", 11th Int. Conf. Pattern Recognition, Vol. 1 , pp. 514-663, The Hague, the Netherlands, 1992.

[70] L. Lam, *et al.*, "A Knowledge-based Boundary Convergence Algorithm for Line Detection", Pattern Recognition Letters, Vol. 15, pp. 383-392, 1994.

[71] Y. Zhu, "New Line-based Thinning Algorithm", IEE Proceedings-Vision and Image Signal Process, Vol. 142, No. 6, pp. 351-358, Dec. 1995.

[72] Palol Pulite, *et al.*, "Knowledge-Based Approach to Image Interpretation", Image and Vision Computing, Vol. 11, No. 3, pp. 122-128, Apr. 1993.

[73] V. M. Kiyko, "Recognition of Objects in Images of Paper Based Line Drawings", Proc. of 3rd Int. Conf. on Document Analysis and Recognition, Vol. 2, pp. 970-973, Montreal, Canada, 1995.

[74] G. Jiao, *et al.*, "On the Extraction of Various Regions in Vector Maps", IEICE Trans. Information and Systems Vol. E78-D, No. 12, pp. 1539-1545, Dec. 1995.

[75] Spatial Data Transfer Standard(SDTS), Federal Information Processing Standards Publication 173, US Geological Survey(USGS), 1998.

[76] Line Eikvil, K Aas, Hans Koren, "Tools for Interactive Map Conversion and Vectorization", Proc. of 3rd Int Conf. on Document Analysis and Recognition, Vol. 2, pp. 927-930, Montreal, Canada, 1995.

[77] B. Pasternak, "The Role of Taxonomy in Drawing Interpretation", Proc. of 3rd Int Conf. on Document Analysis and Recognition, Vol. 2, pp. 799-802, Montreal, Canada, 1995.

[78] Markus Roosli, *et al.*, "A High Quality Vectorization Combining Local Quality Measures and Global Constraints", Proc. of 3rd Int Conf. on Document Analysis and Recognition, Vol. 2, pp. 243-248, Montreal, Canada, 1995.

[79] Zbigniew M. Wojcik, "An Approach to the Recognition of Contours and Line-Shaped Objects", Computer Vision, Graphics, and Processing, Vol. 26, pp. 184-204, 1984.

[80] S. Ablameyko, *et al.*, "Fast Raster-to-vector Conversion of Large-size 2-D Line-drawings in a Restricted Computer Memory", IAPR Workshop on Machine Vision Applications, pp. 59-62, Tokyo, Dec. 1992.

[81] Rafael Santos, Takeshi Ohashi, Takaichi Yoshida, Toshiaki Ejima, Tutorial: Supervised Image Classification with Khoros and the Classify Toolbox Khoros Symposium '97 proceedings, pp.179-191, March 26, 1997.

[82] Thien M. Ha, *et al.*, "Design, Implementation, and Testing of Perturbation Method for Handwritten Numeral Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 535-539, Vol. 19, No. 5, May 1997.

[83] S. Liang, *et al.*, "Segmentation of Interference Marks Using Morphological Approach", Proc. of 3rd Int Conf. on Document Analysis and Recognition, Vol. 2, pp. 1042-1046, Montreal, Canada, 1995.

[84] Hong Yan, "Color Map Image Segmentation Using Optimized Nearest Neighbor Classifiers", Proceedings of the Second International Conference on Document Analysis and Recognition, pp. 111-118, Tsukuba Science City, Japan, Oct. 1993.

[85] Feng Yucai, *et al.*, "An Algorithm for Layering Map Image By Colour", Journal of Software, Vol. 6, No. 7, pp. 435-439, 1995.

[86] S. Ablameyko, "Line-drawing Description: From Skeleton to Hierarchical Vector Representation", Proc. of IEICE Workshop on Pattern Rec. and Understanding, Japan, Oct. 1991.

[87] S. Ablameyko, *et al.*, "Knowledge Based Technique for Map-drawing Interpretation", Proceedings of the International Conference on Image Processing and Its Application, pp. 550-554, Maastricht, the Netherlands, 1992.

[88] S. Ablameyko, *et al.*, "A System for Automatic Vectorization and Interpretation of Graphic Images", Pattern Recognition and Image Analysis Systems, Vol. 3, No. 1, pp. 39-52, 1993.

[89] Gudrun J. Klinker, "A Physical Approach to Color Image Understanding", A K Peters, Ltd. Wellesley, Massachusetts, 1993.

[90] Valavanis KP, "A Total Color-Difference Masure For Segmentation In Color Images", Journal Of Intelligent & Robotic Systems, Vol. 16, No. 3, pp. 269-313, July 1996.

[91] Yan H, Wu J, "Character And Line Extraction From Color Map Images Using A Multilayer Neural-Network", Pattern Recognition Letters, Vol. 15, No. 1, pp. 97-103, Jan. 1994.

[92] A. Okamoto, "Integration of Color and Range Data for Three-Dimensional Scene Description", IEICE Trans. Information and Systems, Vol. E76-D, No. 4, Apr. 1993.

[93] Dori D., "Orthogonal ZIGZAG —— An Algorithm for Vectorizing Engineering Drawings Compared with Hough Transform", Advances in Engineering Software, Vol. 28, No. 1, pp. 11-24, Jan. 1997.

[94] T. Pavidis, "A Vectorizer and Feature Extractor for Document Recognition", Computer Vision, Graphics and Image Processing, Vol. 35, pp. 111-127, 1986.

[95] Limin Fu, "Neural Networks in Computer Intelligence", McGraw-Hill, Inc. 1994.