P.4193/00

STUDIA INFORMATICA

Formerly: *Zeszyty Naukowe Politechniki Śląskiej, seria INFORMATYKA*

Volume 21, Number 4 (42)

Katarzyna STĄPOR

INTEGRATION OF STRUCTURAL PATTERN RECOGNITION METHODS INTO KNOWLEDGE-BASED FRAMEWORK

Application to geographic map image analysis

Silesian University of Technology Press
Gliwice 2000

P. 4193/00

STUDIA INFORMATICA

Formerly: *Zeszyty Naukowe Politechniki Śląskiej, seria INFORMATYKA*

Volume 21, Number 4 (42)

Katarzyna STĄPOR

INTEGRATION OF STRUCTURAL PATTERN RECOGNITION METHODS INTO KNOWLEDGE-BASED FRAMEWORK

Application to geographic map image analysis

Silesian University of Technology Press
Gliwice 2000

STUDIA INFORMATICA **Volume 21, Number 4 (42)**
Formerly: ***Zeszyty Naukowe Politechniki Śląskiej, seria INFORMATYKA***

Editor in Chief

Dr. Marcin SKOWRONEK
Silesian University of Technology
Gliwice, Poland

Editorial Board

Dr. Mauro CISLAGHI
Project Automation
Monza, Italy

Prof. Bernard COURTOIS
Lab. TIMA
Grenoble, France

Prof. Tadeusz CZACHÓRSKI
Silesian University of Technology
Gliwice, Poland

Prof. Jean-Michel FOURNEAU
Université de Versailles - St. Quentin
Versailles, France

Prof. Jurij KOROSTIL
IPME NAN Ukraina
Kiev, Ukraine

Dr. George P. KOWALCZYK
Networks Integrators Associates, President
Parkland, USA

Prof. Peter NEUMANN
Otto-von-Guericke Universität
Barleben, Germany

Prof. Olgierd A. PALUSINSKI
University of Arizona
Tucson, USA

Prof. Svetlana V. PROKOPCHINA
Scientific Research Institute BITIS
Sankt-Petersburg, Russia

Prof. Karl REISS
Universität Karlsruhe
Karlsruhe, Germany

Prof. Jean-Marc TOULOTTE
Université des Sciences et Technologies de Lille
Villeneuve d'Ascq, France

Prof. Sarma B. K. VRUDHULA
University of Arizona
Tucson, USA

Prof. Hamid VAKILZADIAN
University of Nebraska-Lincoln
Lincoln, USA

Prof. Adam WOLISZ
Technical University of Berlin
Berlin, Germany

Dr. Lech ZNAMIROWSKI
Silesian University of Technology
Gliwice, Poland

© Copyright by Silesian University of Technology Press, Gliwice 2000
PL ISSN 0208-7286
Printed in Poland

ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ

OPINIODAWCY:
Prof. dr hab. inż. Marek Kurzyński
Doc. dr hab. inż. Wojciech Mokrzycki

KOLEGIUM REDAKCYJNE
REDAKTOR NACZELNY - Prof. dr hab. Zygmunt Kleszczewski
REDAKTOR DZIAŁU - Dr inż. Marcin Skowronek
SEKRETARZ REDAKCJI - Mgr Elżbieta Leśko

In memory of my Parents

Contents

1 Introduction **11**
1. The motivation of the thesis . . . 11
   1.1. Geographic map image analysis . . . 11
   1.2. Structural pattern recognition in geographic map image analysis . 15
2. The aims of the thesis . . . 17
3. The overall structure of the thesis . . . 18

2 Structural pattern recognition **19**
1. Introduction . . . 19
2. Prototype matching methods . . . 21
   2.1. Strings . . . 21
   2.2. Graphs . . . 22
   2.3. Relational structures . . . 26
3. Syntactic methods . . . 28
   3.1. String grammar . . . 29
   3.2. Generalizations of string grammar . . . 33
   3.3. Graph grammar . . . 35
4. Pattern recognition using structural distances . . . 37

3 Knowledge representation and inference **41**
1. Introduction . . . 41
2. Classical first-order logic . . . 41
   2.1. Syntax . . . 42
   2.2. Semantics . . . 43
   2.3. First-order theory as a deduction system . . . 44
   2.4. Clausal form . . . 44
   2.5. Resolution rule . . . 46
3. Rules . . . 49
4. Semantic networks . . . 50
5. Frames . . . 50
6. Object-oriented approach . . . 51
7. Non-monotonic logic . . . 52
   7.1. Non-monotonic reasoning . . . 52

7.2. Poole's approach . . . 53

4 Geographic map image analysis **55**
1. Low level . . . 55
1.1. Binarization . . . 55
1.2. Text/Graphics segmentation . . . 56
1.3. Thinning . . . 56
1.4. Vectorization . . . 58
2. Intermediate level . . . 62
3. High level . . . 64

5 The proposed framework **67**
1. General characteristics of the framework . . . 69
1.1. The components of the framework . . . 69
1.2. The level structure . . . 71
1.3. The modular structure . . . 72
1.4. Model-based object recognition . . . 75

6 Vector Graph construction module **79**
1. Introduction . . . 79
2. Big Graph vectorization procedure . . . 81
2.1. The algorithm . . . 81
3. Small Graph vectorization procedure . . . 84
3.1. The algorithm . . . 85
3.2. Experimental results . . . 91
4. Vector Graph representation . . . 92

7 The map model **99**
1. Syntax . . . 99
2. Semantics . . . 101

8 The image analysis flow scheme **107**
1. The control mechanism . . . 107
1.1. Hypotheses generation . . . 109
1.2. Compatibility examination . . . 109
1.3. Scenario selection . . . 110
1.4. Scenario verification . . . 110
2. Complementary reasoning . . . 110
2.1. Construction of GE relations . . . 111
2.2. Construction of PO relations . . . 112
2.3. Formalization of complementary reasoning . . . 113
3. Examples of constructing interpretations . . . 117

9 Classifier based on relational matching **125**
1. The overview of the method . . . 125
2. Relational representation of a general 2D-curve . . . 128
2.1. Quasi-topological structure of a curve . . . 128
2.2. Geometric structure of a multiple point . . . 131
3. The hybrid description of a general 2D-curve . . . 132
3.1. The relational description . . . 132
3.2. Parameterization of a shape . . . 135
4. Hybrid matching . . . 136
4.1. Relational matching . . . 136
5. Experimental results . . . 138

10 Classifier based on inexact graph matching **143**
1. Introduction . . . 143
2. Geographic symbol representation . . . 144
3. Homomorphism between graphs . . . 147
4. Graph transformations and their costs . . . 150
5. Graph distance measure . . . 152
6. Inexact graph matching algorithm . . . 153
6.1. Matching component graphs . . . 154
6.2. Matching graphs of components . . . 154
6.3. The complexity . . . 159
7. Experimental results . . . 159
7.1. The example of 1-component symbol matching . . . 160
7.2. The example of multi-component symbol matching . . . 161

11 Syntactic classifier **163**
1. Introduction . . . 163
2. Recognition of overlapped objects . . . 164
3. The grammar . . . 165
3.1. Primitives . . . 165
3.2. Relational symbols . . . 166
3.3. Production rules . . . 167
4. The recognition algorithm . . . 168
4.1. The parsing algorithm . . . 168
4.2. Error-correction . . . 170
4.3. Computational complexity . . . 170

12 Discussion and conclusions **175**
1. Image analysis flow scheme . . . 176
2. Map model . . . 178
3. Object detection methods . . . 178
4. Model-based vectorization . . . 180

A Layers of Polish Fundamental Land Map 181

Bibliography 185

Abstract 209

Spis treści

1 Wstęp 11
1. Motywacja pracy . . . 11
1.1. Automatyczna analiza map geograficznych . . . 11
1.2. Strukturalne rozpoznawanie obrazów w automatycznej analizie map geograficznych . . . 15
2. Cele pracy . . . 17
3. Układ pracy . . . 18

2 Strukturalne rozpoznawanie obrazów 19
1. Wprowadzenie . . . 19
2. Metody uzgadniania wzorców . . . 21
2.1. Łańcuchy . . . 21
2.2. Grafy . . . 22
2.3. Struktury relacyjne . . . 26
3. Metody syntaktyczne . . . 28
3.1. Gramatyka łańcuchowa . . . 29
3.2. Uogólnione gramatyki łańcuchowe . . . 33
3.3. Gramatyka grafowa . . . 35
4. Rozpoznawanie obrazów na podstawie strukturalnych odległości . . . 37

3 Reprezentacja wiedzy i wnioskowanie 41
1. Wprowadzenie . . . 41
2. Logika predykatów pierwszego rzędu . . . 41
2.1. Syntaktyka . . . 42
2.2. Semantyka . . . 43
2.3. Teoria pierwszego rzędu jako system dedukcyjny . . . 44
2.4. Postać klauzulowa . . . 44
2.5. Reguła rezolucji . . . 46
3. Reguły . . . 49
4. Sieci semantyczne . . . 50
5. Ramy . . . 50
6. Podejście obiektowe . . . 51
7. Logika niemonotoniczna . . . 52

7.1. Wnioskowanie niemonotoniczne . . . 52
7.2. Podejście Poole'a . . . 53

4 Automatyczna analiza map geograficznych 55
1. Poziom niski . . . 55
1.1. Binaryzacja . . . 55
1.2. Segmentacja tekstu/grafiki . . . 56
1.3. Ścienianie . . . 57
1.4. Wektoryzacja . . . 58
2. Poziom średni . . . 62
3. Poziom wysoki . . . 64

5 Zaproponowana struktura wnioskująca 67
1. Ogólna charakterystyka struktury . . . 69
1.1. Składniki struktury . . . 69
1.2. Schemat poziomów . . . 71
1.3. Schemat modułów . . . 72
1.4. Rozpoznawanie obiektów oparte na modelu . . . 75

6 Moduł konstrukcji grafu wektorowego 79
1. Wprowadzenie . . . 79
2. Procedura wektoryzacji dużego grafu . . . 81
2.1. Algorytm . . . 81
3. Procedura wektoryzacji małego grafu . . . 84
3.1. Algorytm . . . 85
3.2. Wyniki eksperymentu . . . 91
4. Reprezentacja grafu wektorowego . . . 92

7 Model mapy 99
1. Syntaktyka . . . 99
2. Semantyka . . . 101

8 Schemat przebiegu analizy obrazu 107
1. Mechanizm kontroli . . . 107
1.1. Generacja hipotez . . . 109
1.2. Test kompatybilności . . . 109
1.3. Wybór scenariusza . . . 110
1.4. Weryfikacja scenariusza . . . 110
2. Wnioskowanie uzupełniające . . . 110
2.1. Konstrukcja związków GE . . . 111
2.2. Konstrukcja związków PO . . . 112
2.3. Formalizacja wnioskowania uzupełniającego . . . 113
3. Przykłady tworzenia interpretacji . . . 117

9 Klasyfikator oparty na uzgadnianiu relacyjnym 125
1. Ogólny opis metody . . . 125
2. Reprezentacja relacyjna krzywej 2D . . . 128
2.1. Quasi-topologiczna struktura krzywej . . . 128
2.2. Struktura geometryczna punktu wielokrotnego . . . 131
3. Opis hybrydowy krzywej 2D . . . 132
3.1. Opis relacyjny . . . 132
3.2. Parametryzacja kształtu . . . 135
4. Uzgadnianie hybrydowe . . . 136
4.1. Uzgadnianie relacyjne . . . 136
5. Wyniki eksperymentu . . . 138

10 Klasyfikator oparty na niedokładnym dopasowaniu grafów 143
1. Wprowadzenie . . . 143
2. Reprezentacja symbolu geograficznego . . . 144
3. Homomorfizm grafów . . . 147
4. Transformacje grafów i ich koszty . . . 150
5. Miara odległości grafów . . . 152
6. Algorytm niedokładnego dopasowania grafów . . . 153
6.1. Uzgadnianie grafów struktury . . . 154
6.2. Uzgadnianie grafów komponentów . . . 154
6.3. Złożoność obliczeniowa . . . 159
7. Wyniki eksperymentu . . . 159
7.1. Przykład dla symbolu 1-składnikowego . . . 160
7.2. Przykład dla symbolu wieloskładnikowego . . . 161

11 Klasyfikator syntaktyczny 163
1. Wprowadzenie . . . 163
2. Rozpoznawanie obiektów nałożonych . . . 164
3. Gramatyka . . . 165
3.1. Prymitywy . . . 165
3.2. Symbole relacyjne . . . 166
3.3. Produkcje . . . 167
4. Algorytm rozpoznawania . . . 168
4.1. Algorytm analizy syntaktycznej . . . 168
4.2. Korekcja błędów . . . 170
4.3. Złożoność obliczeniowa . . . 170

12 Dyskusja i wnioski 175
1. Schemat przebiegu analizy obrazu . . . 176
2. Model mapy . . . 178
3. Metody detekcji obiektów . . . 178
4. Wektoryzacja oparta na modelu . . . 180

A Warstwy Polskiej Zasadniczej Mapy Kraju 181
Literatura 185
Streszczenie 211

Chapter 1

Introduction

1. The motivation of the thesis

1.1. Geographic map image analysis

The main objectives of *document image analysis* ([28, 135]) are to obtain a succint description of the information content in *paper documents*, and to create a *digital* representation that would facilitate efficient storage, retrieval, transmission and updating. The dominant component of many documents such as engineering drawings, electrical schematics, flow charts, utility company drawings, and different *maps*, is *graphics*, that means different lines, symbols, regions. *Textual* elements are used only to augment the information given by graphical elements in these documents.

Document image analysis as any other image analysis process ([133]), consists of three main levels of abstraction. In the *low level*, processing is performed on the input, raw document image acquired by scanning, and includes: noise reduction, signal enhancement, binarization, and segmentation which usually includes separation of text from graphics component to enable easier subsequent detection of pertinent features, and object of interest. In the *intermediate level*, text and the primitive graphics objects like lines with different styles, polygons, circles, etc. are recognized. In the *high level*, known as *image understanding*, by interrelating the recognized graphics objects with textual ones, using *a domain dependent knowledge*, a description (semantics) is extracted from the document. Techniques used in this level are strongly application-dependent.

In the industrialized world there is an urgent need for solutions in conversion of different documents from paper to a digital format. Efficient algorithms for document image analysis have many immediate applications. For example, they provide a mechanism for archiving documents such as journals and news magazines, permit a conversion of paper-based engineering drawings into a CAD (Computer Aided Design) compatible form, or enable a conversion of paper-based maps of different types into a database of Geographic/Land Information System (GIS/LIS) (see for example [99] for an overview of GIS technology).

Among many branches of document image analysis, the most important we find geo-

graphic map image analysis, and engineering drawing image analysis. The conversion of paper maps into GIS, which is the subject of research of geographic map image analysis, due to special nature of maps, has not been achieved a matured level yet. There are still lots of problems waiting for solutions ([320, 342, 24]). This is the main reason for choosing conversion of paper maps as a subject of our research.

The rapid growth in the volume and complexity of the geographic maps, naturally led to the development of computerized geographic information systems to facilitate the information management. The main application of GIS is a *cartographic modeling*, or *spatial analysis*, for solving compound spatial problems ([99]). For efficient management of many companies such as public utility, the information in different paper maps has to be available in GIS, even at the high expense in investments in the conversion process. The motivation of this process is threefold. First, there is a need for efficient retrieval of symbolic, numeric, and spatial information. Manual retrieval of information from maps is at least time consuming. A second reason for conversion is the efficient maintenance of the information in maps. The infrastructure drawn on the maps is not static, it is therefore necessary to update the paper maps on a regular basis. Modeling and simulation of different events, optimization of spatial processes (for example routes), is the third important advantage of a digital description.

Considering the enormous amount of paper information and its importance to a variety of industrial branches, efficient conversion of maps is a topic of great interest.

The acquisition of the relevant information to fill GIS is more cost-intensive than the investment into GIS system. Compared to photogrammetry, or surveying, existing paper maps are rather low-cost source of the input data. Acquiring the relevant geographic information from existing paper maps into GIS by manual digitization, using overlay of scanned map in an interactive editor, is a very tedious, expensive and long-time process. Moreover, as it requires the use of the expertise of the human operator, the time, cost and error rate of the acquistion rate depends on the skill and care of the operator, and the quality inspection. Thus, the availability of systems for automatic conversion of paper maps to adequate geographical data standards would be very welcome.

We have identified the following problems in map image analysis, which have not yet been treated successfully: 1) reasoning with incomplete information, 2) control strategy, 3) contextual layer separation, 4) reliable analysis. We will discuss them briefly.

Problem 1. *Reasoning with incomplete information*

The problem of map image understanding is *underconstrained*, that means the observed information is incomplete, as the new image features and object instances may be inserted at any time during the map analysis process, which affects the interpretation. Since map image analysis systems in principle cannot recover this intrinsic insufficiency, the best possible solution is to construct plausible descriptions of the map based on the available information. Since the amount of input information is limited, multiple descriptions should be constructed. In *optimization* approach to cope with incomplete information (see for example [154]), the information obtained from an input image is considered as constraints, and the evaluation function which measures the goodness of match between image features and object models

is optimized. Beside the problem of correct formulation of evaluation functions, optimization methods often perform uniform processing, while the image structure, and complexity of the map vary depending on locations. Among various methods in probabilistic reasoning, the *evidential reasoning* ([130, 255, 128, 70, 289]) is useful to cope with incomplete information. Each piece of information is associated with a probability representing its reliability. When new information is obtained, the probability is modified depending on whether or not the new information is consistent with it. The kernel computation in evidential reasoning is how to modify the probability to accomodate the new information. Although probabilistic reasoning is useful to cope with incomplete information, it is *passive*. No active reasoning mechanism to generate hypotheses for missing information is included. In image analysis systems we need active reasoning mechanisms to complement the intrinsic insufficiency of the input information. Map analysis process should be constructive, that means it should *actively generate* new information based on the model to cope with the insufficiency of the input information. Reasoning methods in classic AI systems are based on ordinary deductive reasoning. Ordinary *deductive reasoning* based on classical formal logic does not work well in image analysis, as it proves the validity of theorems in the light of the given (fixed) information, while map image analysis systems need to construct the description of the map by dynamically generating new information. In AI, several methods have been proposed to formulate reasoning with incomplete knowledge, so-called *non-monotonic reasoning* ([281, 225, 226, 223, 224, 266, 267, 301]), and the map image analysis should be based on such non-monotonic reasoning scheme to be able to perform deductions of new facts from sets of axioms.

Problem 2. *Control strategy*

For optimal interpretation results, an effective map image analysis *strategy* is required to control the analysis. Neither of the control strategies: top-down or bottom-up, yields a complete solution to the map image analysis. In *bottom-up strategy* algorithms are performed in a fixed sequence: low-level basic primitives (lines, polygons, symbols, characters, textures, circles, dashed-lines, etc.) are extracted by specialized operators. Higher level graphics entities (houses, parcels, etc.) are constructed from low-level ones, guided by some a priori knowledge. If the image data is noisy or ambigous, the results of a processing step may contain errors. Due to the static nature of the bottom up approach, subsequent steps may not be able to detect and handle these errors, thus causing the errors to propagate upwards along the processing path to the final understanding step. For more complex applications, a single algorithm within each processing step may be insufficient, but in a bottom-up approach, no model is available to determine when a specific algorithm should be used. The *top-down* strategy is based on hypothesize-and-test concept. The internal model guides the analysis process in promising directions through the generation of the proper hypotheses. A complex hypothesis may be split into sub-hypotheses, which in turn can be divided further until a hypothesis

can be either accepted, or rejected using only the image processing for its verification. The analysis, only yields good results if the image contains what the model describes. Top-down control alone is never sufficient for image interpretation, and some bottom-up processing will always be required. Therefore, it is a need for developing a strategy which integrates both to get better results.

Problem 3. *Contextual layer separation*
Complex maps contain several layers of information that overlap substantially, making it impossible to geometrically segment the map data into distinct regions containing a single class of graphics object. Typical situations which cause problems in map image segmentation are: overlapping elements (graphics-graphics, text-graphics), deformation of symbols, errorneous shapes: open polygons, missing elements of objects (text description), not brought up to date maps with false objects (for example somebody has forgotten to remove the scarp along a river after the road has been built). An approach that applies the same processing uniformly to features in all layers, by isolation each of the graphics entity types, application of entity-specific recognition, and merging the results into an integrated interpretation by applying some contextual knowledge may fail, because an error in the isolation will preclude success in the subsequent processing steps. For example, the parameter values that are apropriate for controling a vectorization and linking process for one type of linear feature (e.g. roads), may not works well on another type of linear feature (e.g. streams). The imperfection of segmentation leads to the following two problems:

1. Introduction of errorneous information (errorneous image features.
2. Lack of meaningful information (i.e. missing image features).

Since these problems are inevitable, image analysis systems should incorporate versatile mechanisms to cope with them: identification and elimination of errorneous information, and investigation and extraction of missing information.

Problem 4. *Reliable analysis*
The automatic conversion of maps should be very reliable as errors in the analysis may seriously affect later processing. To obtain a reliable interpretation it is necesary to utilize all the available knowledge about the application. There is rich a priori knowledge associated with most maps: information from legends, drawing rules, map specifications. A priori knowledge can be used at various levels to drive the analysis process. A strong verification of the results with a model, as well as partial results is needed. As input information is incomplete, construction of multiple description can increase the reliability of the analysis.

Despite the research efforts in map image analysis, most of the research in this area still concentrates on the design of specific, application-dependent algorithms rather than on the overall process of map image analysis. Only 4 described in the literature systems, namely [222, 306, 252, 151] can be regarded as map image analysis systems. None of these approaches, however, propose reasoning method with incomplete information. Only in

[151], the domain model is used, but no multiple object descriptions are generated to construct plausible interpretation. The required mixed control strategy in map analysis systems has different degree of flexibility. In the simplest one, a top-down analysis is used only to obtain additional results when bottom-up processing is incorrect. The mixed control strategy in the Dutch utility maps ([151]) is more matured, but it is based on fixed alternation of a top-down process with a bottom-up one and inconsistency detection. There is still a long way to go before having a good contextual layer separation process for technical drawings ([342]). A common disadvantage of all the mentioned systems is the lack of sufficient reliability and accuracy ([321]).

From this overview it may be concluded that most of the mentioned problems in geographic map image analysis still need solutions to obtain a level of reliability allowing for practical application.

Some solutions to problem 1 and 3 were proposed in aerial image interpretation. A proposition to cope with insufficiency of input information was proposed in [221] for interpretation of buildings and roads in simple aerial images. Also, in [227, 362], being the systems for aerial image understanding, some resegmentation technique to cope with imperfect segmentation have been used.

1.2. Structural pattern recognition in geographic map image analysis

Document image analysis is a domain where the *structural pattern recognition* methods ([120, 51, 331, 285, 52, 132, 233, 235, 107, 253, 193]) are well suited. Examples are *character* recognition ([91, 302, 365, 314, 317]) and *symbol* recognition ([141, 103, 201, 48] in different documents and drawings (electrical, mechanical, maps), *line texture* extraction by means of syntactic rules in engineering drawings ([2, 318]), *mathematical notation* recognition ([14, 33]) which is typically organized according to strict syntactic rules, *printed documents* recognition ([140, 246, 356]) and *music scores* recognition ([32, 65]) in which syntactic methods have been used to recognize their logical structure. In mechanical engineering drawings it has been proved that the dimensioning rules can be described by a grammar, thus enabling syntactic recognition ([77, 78, 79, 80, 64]). Structural prototypes have been used to represent the syntax of engineering drawings, and to to perform various interpretation tasks based on this representation ([156, 163, 164, 111]).

Geographic map images have special properties which would make structural pattern recognition techniques particularly attractive for recognition. A map is composed of several layers of information. Each map layer is composed of several groups of geographic objects, and can be described by means of a prototype structure, or a grammar. These formal pattern representations can be used to verify and organize the data. Classes of geographic objects also follow some kind of structure which can be represented by strings, trees or graphs - the basic kinds of representation methods used in structural pattern recognition. Moreover, in each layer, there exist many different spatial relations between geographic objects in a domain, which make structural pattern recognition particularly suitable. Some classes of geographic objects may be specializations of more general ones. Instances of some classes may be composed of parts being the instances of other classes.

As the "layout" of each map layer is essentially structured, and can be described by a tree, graph or a grammar, its analysis can be done through structural pattern recognition techniques.

But it has been ours, as well as other author's experience (see for example [282, 342, 337]), that when designing larger drawing analysis systems in general, and map image analysis ones in particular, we come to a number of what can be called *border problems*. The following two are connected with usage a grammar: limited descriptive power of a grammar and errorneous patterns. Also, when using structural prototypes, we encounter the following problem: high number of prototypes and high diversity of documents structures. Beside the above problems which are *intrinsic* to a method, the common problem is the high sensitivity to noise and errors of structural pattern recognition methods.

We will discuss all the mentioned problems briefly.

Problem 5. *Limited descriptive power of a grammar*
The descriptive power of the string and tree grammars are limited. This implies that some complex patterns can not be generated, and in others it is not trivial to find the grammar. In some cases it is also difficult to modify a grammar to include new patterns which structure differs slightly from the basic, typical structure ([282]).

Problem 6. *Errorneous patterns*
The use of the grammar which generates an infinite language can become a problem, since it can derive structures which do not belong to the pattern class ([337].

Problem 7. *High number of prototypes per class and diversity of document's structures*
The design of pattern analysis system implies generation of all the prototypes patterns which must be used to analyze the pattern class. Document images have not a fixed structure, but contain substructures which differ slightly in information, position and orientation. If the number of patterns is high, which is not rare in document analysis, it is not possible to generate all possible patterns ([282]).

Parsing, as well as structural prototype matching methods tend to be very *sensitive to noise and errors* in the input data. Many methods have been proposed to deal with this: inexact graph matching using some kind of distance ([291]), error-correcting parsing ([334]) or probabilistic relaxation ([47]). Thus, in principle, error-correcting methods are available if we consider the pattern recognition problem as a standalone module. However, real life applications are more complex than that, and that's where we come to another limits of the existing methods.

Problem 8. *Interaction between segmentation and recognition*
In document images in general, and map drawings in particular, despite a noise associated with scanning, discretization, and image processing, we have to deal with information which is represented clearly enough, but which consists of several information layers which are superimposed in document (for example text overlaps graphics). As bottom-up methods usually perform some string segmentation based on the analysis of connected components, touching characters tend to be consider as mere noise. In this case it is not possible to only rely on the robustness to noise of the pattern recognition method. We need to actually retrieve the lost information through a some kind of interaction between segmentation and recognition.

Problem 9. *Selection a region of interest*
When analyzing a complex document, we need an efficient method for *selection of region of interest*, that means to extract potential candidate structures (graphs, trees), or primitive pattern constituents to be matched with the models, or used in construction of a derivation by a parser. Without such a method the system would waste its time trying to match what is actually intersections between lines, too large areas of the diagram, or even subgraphs containing only a part of a candidate.

The graphics objects, as well as text elements in most existing map analysis systems are recognized by statistical classifiers ([264, 372, 237, 16, 98, 261, 242, 219, 380, 280]) or by simple template matching technique ([151, 222]). No structural detectors have been used in map image analysis, a few in other documents analysis systems (for example [35]).

The main drawback of all the existing algorithms is that they rely on perfect entity isolation prior recognition which may not be successful because due to objects overlapping, an error in isolation will preclude success in the subsequent processing steps. To cope with problems 1 and 2 of structural pattern recognition, an approach based on attributed programmed graph grammar applied to schematic diagrams interpretation has been proposed in [48]. Attributed grammar has also been used in character recognition ([348]). An approach augmenting limitive desciptive power of a string grammar, by using special operators describing two-dimensionality has been proposed in [232], for sketches recognition, but a parser has non-polynomial complexity. To decrease a high number of structural prototypes, an interesting approach based on inexact graph matching has been proposed in [270] for character recognition, but as it relies on local conditions during matching, it fails in more complicated situations.

From this overview it may be concluded that relatively little research on the problems of structural pattern recognition in map image analysis has been reported.

2. The aims of the thesis

The goal of the research described in this thesis is the design, and development of efficient techniques which are capable of (semi-) automatic map drawing conversion. Because of the multitude of map applications, the flexibility of these techniques is a very important aspect of our research. Moreover, as structural pattern recognition is particularly suitable in map image analysis, methods to overcome its border problems mentioned above, should also be developed, in order to enable its successful application in map image analysis.

In summary, the following are the two main aims of the presented thesis:

- The first aim is the development of a generally applicable *reasoning framework* for geographic map image analysis which allows to solve all the mentioned problems of map image analysis. This means that we should provide in this work a knowledge representation mechanism, together with suitable inference procedures, and control strategies which allow for the integration of a variety of pieces of knowledge, each represented by means of a different formalism into one consistent framework.
- The second aim is to augment the proposed framework with properly devised *structural* detectors of graphics and text components of map image, so as to overcome the mentioned border problems of structural pattern recognition. This means that we should devise in this work structural classifiers in which the intrinsic limitations of structural pattern recognition are solved by proper augmentation of a grammar, or prototype structure. Also, the devised classifiers should be properly inserted in the whole analysis process, so as to enable to solve the remaining border problems of structural pattern recognition.

3. The overall structure of the thesis

As one of the aims of the thesis is to develop suitable structural detectors for geographic object recognition, Chapter 2 describes basic notions related to the structural pattern recognition. It shortly presents the representation and recognition methods used in two basic approaches in structural pattern recognition: syntactic and prototype structure one.

As it has been discussed in the previous section, there is much domain knowledge connected with each type of a map which should be represent and used to guide the analysis process. The knowledge representation format should, therefore, provide for simple but effective means to model and manipulate the knowledge. Basic knowledge representation formalisms, as well as the reasoning methods are shortly characterized in Chapter 3.

Chapter 4 outlines the state-of-the-art in geographic map image analysis, that means it presents the existing algorihms used in the three levels of image analysis.

The remaining chapters (5-12) describe the proposed, general reasoning framework for geographic map image analysis. Chapter 5 describes the overall structure of the proposed framework. Chapters 6 describes the construction of the graph representation of a scanned map image. Chapters 7 and 8 are descriptions of the proposed map model and image analysis flow scheme. Chapters 9-11 characterize the three structural detectors (classifiers) for geographic objects developed within the proposed framework. These detectors perform the model-based object recognition based on appropriate shape models. Experimental results of performing map image analysis according to the proposed methodology are given in appropriate chapters describing particular aspect of the analysis. Chapter 12 discusses the concepts developed during this research, the merits and limitations of these concepts, and gives directions for future research. Finally, Appendix A shows examples of fragments of maps from the Polish Fundamental Land Map.

Chapter 2

Structural pattern recognition

1. Introduction

The aim of the pattern recognition science ([60, 331, 285]) is to provide methods for *classification* (*statistical* pattern recognition) or *description* and *classification* (*structural* pattern recognition) of entities/events called *patterns*.

Statistical, or decision-theoretic methods ([195, 123, 90, 74, 189, 67, 169, 196, 197, 198]) attempt to classify patterns based on a set of extracted features and an underlying statistical (perhaps ad hoc) model for the generation of these patterns. In a statistical classifier, samples are represented by a coefficients of a decision function, parameters of a probability distribution, or points in a feature space, and the concept of similarity is based on the distance, or decision functions in the n-dimensional space of real numbers.

Many patterns contain structural, or more generally relational information, that is difficult, or impossible to quantify in feature vector form. Statistical approach works well when the problem involves models that do not include constraints relating different parts of a model.

Structural pattern recognition ([331, 285, 120, 51, 52, 132, 233, 235, 107, 253, 191, 192, 193]) **(spr)** methods, on the other hand, are based on explicit or implicit representations of a structure of a class, where *structure* conceptually means the characteristic way in which the subpatterns (components) of a pattern are related, or configured together. Structural information is used here for modelling the hierarchical composition of a complex pattern from simpler subpatterns, and the various relations which may exist between different subpatterns and/or characteristic features. The two major subcategories of structural pattern recognition are:

- **Syntactic** methods
- **Prototype matching** methods.

In syntactic approach, the structure of classes of patterns is represented by *formal grammars*, while in the prototype matching approach, by *structural prototypes*, that means different symbolic data structures like strings, trees, graphs, etc., instead of vector

of numbers that are used in the statistical approach. Patterns are usually divided into two sets: the sample set of patterns that are used for system design (so called *model*, or *prototype patterns*), and the unknown patterns to be recognized in the actual application phase of a pattern recognition system. Typically, the same kind of data structures is used for the unknown patterns and the samples.

In spr, the *recognition* of an unknown pattern is usually accomplished by comparing its symbolic representation with a number of predefined model patterns. The process of comparison is based on *syntactic parsing* (in syntactic methods) or *symbolic matching* (in prototype matching).

The use of *formal grammars* is based on a very well established formal language and automata theory ([11, 156]). Every pattern class C is defined by a *language* L generated by a grammar G defining this class, i.e. $C = L(G)$. Structural relations are defined *implicitly*, that is the relations satisfied by every element in the language are implied by the syntactic and semantic rules of the grammar. Grammars can be used for generating the document images and for accepting them by means of parsing (i.e. to know if a document structure belongs to a grammar). Grammars are classified according to the type of structure they generate: *string* grammars ([11, 156]) (generating strings over an alphabet), *matrix* and *array* grammars ([274]) (generating two dimensional arrays, *tree* ([274, 30, 76, 31]) and *graph* ([245, 95, 96, 97, 108]) grammars (generating trees and graphs). There are other special type of grammar which are actually graph grammars, but that are generally known as *higher-dimensional extensions* of string grammars, for example PDL ([297]) or Plex grammar ([105]). If the large number of pattern samples is involved, pattern class representation is more efficient if a grammar is used. In the case where too few sample patterns are available for the *inference* ([117, 118]) of an abstract class description in terms of a grammar, structural prototype matching is the method to be preferred.

Prototype structures are based on discrete mathematical models ([308, 218, 240]) like *strings*, *trees*, *graphs*, *arrays* or *relational structures* as a detailed description of a structure. Words of symbols, or *strings* are the most fundamental data structure for pattern representation. The individual symbols in a string usually represent atomic pattern components. More powerful structure for 2- and 3-dimensional pattern representation is a *graph* consisting of a set of nodes which usually represent simpler subpatterns, and the edges which indicate relations of any kind between those subpatterns. *Trees* are special subclass of graphs having three types of nodes: exactly one root having only outgoing edges, several interior nodes having exactly one incoming and at least one outcoming edge, and several leave nodes having only incoming edges. An *array* is a special type of graph in which nodes and edges are arranged in a regular form ([361]), for example rectangular arrays with 4- or 8-neighborhood. Prototype structures allow the *explicit* description of relations between elementary pattern components. An example is a perfect prototype in which all details of the relations among subpatterns are fully listed, say, in the explicit form of a directed graph with nodes labeled as subpatterns and arcs labeled as relations.

2. Prototype matching methods

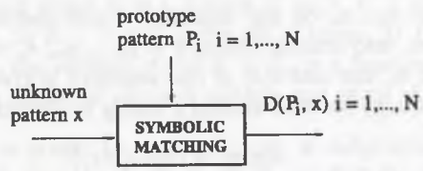


Fig. 2.1. Block diagram of structural matching

Rys. 2.1. Schemat blokowy uzgadniania strukturalnego

In prototype matching approach ([57]), the task of pattern classification can be solved by the recognition procedure shown schematically in Fig. 2.1. Given an unknown input pattern x, and N prototype structures P_i, each of which being a structural specification of a class C_i, the recognition task involves the symbolic *match* (depicted as rectangle in Fig. 2.1), that computes structural distance between an unknown input pattern x, and a number of prototype models P_i, in order to find the prototype P_j, which is the *most similar* to an unknown input pattern x. In real world environments, in which a candidate x may be deformed by noise, or may have natural variations not completely defined in a representation of a class C_i, *inexact*, or *error-correcting* comparison based on *similarity* or *distance* measures is inevitable.

An approach used successfuly in many applications is to define a finite set of structure-modifying *edit operations* as "errors", or "mutations"that can occur in candidate patterns, and to compute quantitative measures that take the costs of these operations into account.

2.1. Strings

2.1.1. String similarity measures

An *alphabet* is a finite set of symbols. A *word*, or *string* x over an alphabet T, is a sequence of symbols $x = x_1x_2...x_n$, where $x_i \in T; i = 1, ..., n$. The empty word ϵ is the sequence with no symbols. The *length* of a word, denoted $|x|$ is equal to the number of symbols contained in it. The set of all words over an alphabet is denoted by T^*.

The *distance* between two strings $x = x_1, ..., x_n$ and $y = y_1, ..., y_m \in T^*, n, m \geq 0$, is defined in terms of elementary *edit operations* which are required in order to transform x into y ([205, 9]), namely:

- substitution of a symbol $a \in T$ in x by a symbol $b \in T$ in y, $a \neq b$ $(a \rightarrow b)$.
- insertion of a symbol $a \in T$ in y $(\epsilon \rightarrow a)$.
- deletion of a symbol $a \in T$ in x $(a \rightarrow \epsilon)$

The edit operations are used for modelling variations which may change an ideal prototype string into its actual, noisy version. Depending on the particular application, certain

distortions, i.e. edit operations, may be more likely than others. In order to take account of this observation, each elementary operation e_i is assigned a cost $c(e_i)$.

Definition 2.1 Let $s = e_1, ...e_n$ be any sequence of edit operations for transforming a string $x = x_1, ..., x_n$ into another string $y = y_1, ..., y_m, n, m \geq 0$. Given the cost of any edit operation e_i, the cost $c(s)$ of this sequence is given by: $c(s) = \sum_{i=1}^{n} c(e_i)$, and the *distance* $d(x, y)$ between strings x and y is defined by ([205, 9]):

$$d(x, y) = min_s\{c(s)\}$$

2.1.2. *Error-correcting string matching*

The algorithm to compute the distance between two strings $x = x_1, ..., x_n$ and $y = y_1, ..., y_m \in T^*, n, m \geq 0$, ([358, 284, 143]), known as *weighted Levenshtein distance*, computes the elements of the $(n+1) \times (m+1)$ matrix $D(i, j)$, row by row from left to right. The basic idea of the algorithm is to find the minimum cost path from $D(0, 0)$ to $D(n, m)$. This path corresponds to the minimum cost sequence of edit operations for transforming x into y. In each element $D(i, j)$ of the path, the minimum accumulative costs are stored for transforming $x' = x...x_i$ into $y' = y...y_j$, i.e. $D(i, j) = d(x', y')$. For any path element $D(i, j)$, going from its predecessor $D(i, j-1)$, corresponds to the insertion of y_j, going from predecessor $D(i-1, j-1)$ represents the substitution of x_i by y_j, and finally, a transition from predecessor $D(i-1, j)$, corresponds to the deletion of x_i. A sequence of edit operations obtained as a result of this algorithm can be interpreted as a structural description of the input pattern y with respect to the most similar pattern x. The time complexity of the algorithm is $O(nm)$ where n and m are lengths of strings x and y.

Improving the time and space complexity of the basic algorithm resulted in many variations of the basic algorithm [358], for example, [155] ($O(min(m, n))$), [220] ($O(nm/min(m, logn))$), [160] ($O(nlogn)$)). Another examples of inexact string matching algorithms can be found in [351, 66, 1, 338, 190, 10]. Context-dependent costs have been studied in [338]. An open problem in string matching is the proper adjustment of the cost of the elementary edit operations. The distance of two strings depends critically of these two costs. A novel approach to the adjustments of edit costs has been proposed in [53]. The idea is to consider the edit costs as parameters, and to compute in the training phase string distances as a function of these parameters. Then, given some optimization criterion, certain intervals of the parameters, i.e. edit costs can be found where the optimization criterion does not change.

2.2. Graphs

Strings are limited in their representational power, because they are intrinsically 1-dimensional. To overcome this problem, *graphs* have been used as a more general method of pattern representation. In this section we introduce *similarity measures* on graphs, and procedures for their computation.

2.2.1. *Graph similarity measures*

2.2.1.1. *Subgraph isomorphism*

Definition 2.2 A *labeled graph* G is a 4-tuple $G = (V, E, f, g)$ ([148]), where: 1) V is the set of vertices, 2) $E \subseteq V \times V$ is the set of edges, 3) $f : V \rightarrow L_V$ is a function assigning labels to the vertices, 4) $g : E \rightarrow L_E$ is a function assigning labels to the edges (L_V, L_E are the set of labels of vertices and edges of the graph, respectively).

Definition 2.3 Given a graph $G = (V, E, f, g)$, a *subgraph* of G is a graph $S = (V_s, E_s, f_s, g_s)$ such that 1) $V_s \subseteq V$, 2) $E_s = E \cap (V_s \times V_s)$, 3) f_s, g_s are the restrictions of f and g to V_s and E_s, respectively.

Definition 2.4 A bijective function $F : V \rightarrow V_1$ is a *graph isomorphism* from a graph $G = (V, E, f, g)$ to a graph $G_1 = (V_1, E_1, f1, g1)$ if ([148]):
1) $f(v) = f1(f(v))$ for all $v \in V$,
2) for any edge $e = (v_1, v_2) \in E$ there exists an edge $e' = (f(v_1), f(v_2)) \in E_1$ such that $g(e) = g1(e')$, and for any $e' = (v_1', v_2') \in E_1$ there exists an edge $e = (f^{-1}(v_1'), f^{-1}(v_2')) \in E$ such that $g1(e') = g(e)$.

Definition 2.5 An injective function $f : V \rightarrow V_1$ is a *subgraph isomorphism* from G to G_1 if there exists a subgraph $S \subseteq G_1$ such that f is a graph isomorphism from G to S.

Subgraph isomorphism detection is an NP-complete problem ([127]), i.e. the number of computational steps required to detect all subgraph isomorphisms from one graph to another is exponential in the size of the underlying graphs. The most common technique to establish a subgraph isomorphism is based on a backtracking in a search tree. In order to prevent the search tree from growing unnecessarily large, different refinement procedures such as the one by Ullman ([352]), forward checking and looking-ahead ([147]), or discrete relaxation ([184]) have been proposed. These techniques are fairly stable and perform well in most cases. Another approaches are based on maximal clique detection [104, 241], or partition the graphs according to lattice theory [37]. All these methods provide an optimal solution to the graph matching problem, but may in the worst case become computationally intractable. Continuous optimization methods (simulated annealing ([152]), genetic algorithms ([178, 46]) and probailistic relaxation ([63])) aim at providing a solution within reasonable time, however, they may not always find the optimal solution, i.e. the mapping of model graph vertices to input graph vertices found, not necessarily represents a subgraph isomorphism.

2.2.1.2. *Measures allowing deformations*

In most practical problems, isomorphism is too rigorous, as its determination procedures allow only *exact matching*. Real world objects which are represented by graphs may be affected by noise, structural deformations and distortions in symbolic attributes. In order to compare the undistorted model graphs to such a distorted input graph, and decide which of the models is most similar to the input, numerous *distance measures* have been

defined ([283, 290]). In this section we describe a subgraph edit distance measure ([49]) which is based on the idea of compensating the distortion in the input graph by means of edit operations that are applied to the model graph, i.e. the graph edit operations are used to alter the model graphs until there exist subgraph isomorphism to the input graph.

Definition 2.6 Given a graph $G = (V, E, f, g)$, a *graph edit opration* δ on G is any of the following ([49]):

- $f(v) \rightarrow l, v \in V, l \in L_V$: substituting the label $f(v)$ of vertex v by l (*correction of vertex label distortions*)
- $g(e) \rightarrow l', e \in E, l' \in L_E$: substituting the label $g(e)$ of edge e by l' (*correction of edge label distortions*)
- $v \rightarrow \epsilon, v \in V$: deleting the vertex v from G (*correction of missing vertices*)
- $e \rightarrow \epsilon, e \in E$: deleting the edge e from G (*correction of missing edges*)
- $\epsilon \rightarrow e = (v_1, v_2), v_1, v_2 \in V$: inserting an edge between two existing vertices v_1, v_2 of G (*correction of extraneous edges*)

In order to model the fact that certain distortions are more likely than others, each graph edit operation δ is assigned a certain *cost* $C(\delta)$.

Definition 2.7 Given a graph $G = (V, E, f, g)$, and an edit operation δ, the *edited graph* ([49], $\delta(G)$, is a graph $\delta(G) = (V_\delta, E_\delta, f_\delta, g_\delta)$ with:

$$V_\delta = \begin{cases} V - \{v\} & if \;\; \delta = (v \rightarrow \epsilon) \\ V & otherwise \end{cases}$$

$$E_\delta = \begin{cases} E \cup \{e\} & if \; \delta = (\epsilon \rightarrow e) \\ E - \{e\} & if \; \delta = (e \rightarrow \epsilon) \\ E \cap (V_\delta \times V_\delta) & otherwise \end{cases}$$

$$f_\delta(v) = \begin{cases} l & if \;\; \delta = (f(v) \rightarrow l) \\ f(v) & otherwise \end{cases}$$

$$g_\delta(e) = \begin{cases} l & if \;\; \delta = (g(e) \rightarrow l) \\ g(e) & otherwise \end{cases}$$

Given a graph $G = (V, E, f, g)$, and a sequence of edit operations $\Delta = (\delta_1, \delta_2, ..., \delta_k)$, the *edited graph* $\Delta(G)$ is a graph $\Delta(G) = \delta_k(...\delta_2(\delta_1(G)))...)$. The total cost of the transformation of G into $\Delta(G)$ is given by $C(\Delta) = \sum_{i=1}^{k} C(\delta_i)$.

Definition 2.7 Given two graphs G and G', an *error-correcting (ec) subgraph isomorphism* f from G to G' ([49, 230]) is a 2-tuple $f = (\Delta, f_\Delta)$, where Δ is a sequence of edit operations such that there exists a subgraph isomorphism from $\Delta(G)$ to G', and f_Δ is a subgraph isomorphism from $\Delta(G)$ to G'. The cost of an *ec* subgraph isomorphism $f = (\Delta, f_\Delta)$ is the cost of the edit operations Δ, i.e. $C(f) = C(\Delta)$.

Definition 2.8 Let G and G' be two graphs. The *subgraph distance* from G to G' ([49, 230]), $d(G, G')$, is given by the minimum cost taken over all error-correcting subgraph isomorphisms f from G to G':

$$d(G, G') = MIN_\Delta\{C(\Delta) \mid \text{there is an } ec \text{ subgraph isomorphism } f = (\Delta, f_\Delta) \text{ from } G \text{ to } G'\}$$

Definition 2.9 The subgraph isomorphism f associated with $d(G, G')$ is called the *optimal error-correcting* subgraph isomorphism from G to G'.

2.2.2. *Error correcting subgraph isomorphism detection by A^**

Given a graph $G = (V, E, f, g)$, an input graph $G_I = (V_I, E_I, f_I, g_I)$ and a set of graph edit operations for which a cost function is defined, we want to find an optimal error-correcting subgraph isomorphism (Δ, f) such that there exists a subgraph isomorphism f from $\Delta(G)$ to G_I, and the costs of Δ, $C(\Delta)$ are minimal.

For convenience, we introduce the following notation. Let p be a mapping from the vertices of G to the vertices of G_I, and p consists of tuples $(v, w), v \in V, w \in (V_I \cup \{\epsilon\})$. The symbol ϵ is used to denote missing vertices in the input graph.

Let $V^p = \{v|(v, w) \in p\}$ and $V_I^p = \{w|(v, w) \in p\}$ denote subsets of vertices of G and G_I, respectively. A mapping p implies a set of edit operations Δ^p on the subgraph S^p of G induced by the vertices V^p, and also the subgraph isomorphism f that matches $\Delta^p(S^p)$ to G_I. For each tuple (v,w) in p:

1. if $w \neq \epsilon$ and $f(v) \neq f_I(w)$ then $(f(v) \rightarrow f_I(w)) \in \Delta^p$
2. if $w = \epsilon$ then $(v \rightarrow \epsilon \in \Delta^p)$.

Also, Δ^p may contain edit operations, i.e. for each pair of tuples $(v, w), (v', w') \in p$:

1. if there exists an edge $e = (v, v') \in E$ and an edge $e_I = (w, w') \in E_I$ then $(g(e) \rightarrow g_I(e_I)) \in \Delta^p$.
2. if there exists an edge $e = (v, v') \in E$ and there is no edge $(w, w') \in E_I$ then $(e \rightarrow \epsilon) \in \Delta^p$.
3. if there exists an edge $e = (v, v') \in E$ but there is an edge $e_I = (w, w') \in E_I$ then $(\epsilon \rightarrow (v, v')) \in \Delta^p$.

The subgraph isomorphism f^p from $\Delta^p(S^p)$ to G_I that is implied by p is defined as:

$$f^p(v) = \begin{cases} w & if \;\; (v, w) \in p, w \neq \epsilon \\ undefined & otherwise \end{cases}$$

for all v in $\Delta^p(G)$.

Thus, given the graphs G and G_I and a set of edit operations, we want to find a mapping p such that $V^p = V$ and $C(p)$ is minimal over all mappings. The search for such a mapping is performed by an algorithm shown in Fig. 2.2 that is similar to A^* algorithm [250], and has been previously described for graph matching by various authors (for example, [49, 291, 347, 230]).

(1) initialize $OPEN$:
for each vertex $w \in V_I \cup \{\epsilon\}$ create a mapping $p = \{(v_1, w)\}$
Set $C(p) = C(f(v_1) \rightarrow f_I(w))$
$OPEN = OPEN \cup \{p\}$
(2) if $OPEN$ empty then exit
(3) select $p \in OPEN$ such that $C(p)$ is minimal
remove p from $OPEN$
(4) if $C(p) > acceptThreshold$ then exit
(5) if p represents complete mapping from G to G_I then output p
Set $acceptThreshold = C(p)$
go to (2)
(6) Let $p = \{(v_1, w_i), ..., (v_k, w_j)\}$, $V_I^p = \{w | (v, w) \in p, w \in V_I\}$
(7) for all $w \in (V_I - V_I^p) \cup \{\epsilon\}$
set $p' = \{(v_1, w_i), ..., (v_k, w_j), (v_{k+1}, w)\}$, compute $C(p')$
$OPEN = OPEN \cup \{p'\}$
(8) go to (2)

Fig. 2.2. Algorithm for error correcting subgraph isomorphism detection by A^*

Rys. 2.2. Algorytm z korekcją błędów dla detekcji izomorfizmu grafów wykorzystujący algorytm A^*

Given L models, each of which with n vertices and an input graph with m vertices, the computational complexity of the algorithm shown in Fig. 2.2 is bounded by $O(Ln^2m)$ in the best case, while in the worst case $O(Ln^2m^n)$. The performance can be greatly improved if also the future edit costs of p are estimetaed and taken into account ([366]). The search space of A^* algorithm can be greatly reduced by applying heuristic error-estimation functions ([291, 100, 347]). All of these methods are guaranteed to find the optimal solution but require exponential time in the worst case. Continuous optimization methods (for example based on a probabilistic relaxation [185, 63], neural networks [106, 378], linear programming [12], eigen-decomposition [353]), on the other hand are polynomialy bounded in the number of computation steps, but may fail to find the optimal solution. In order to overcome another problem of comparing an input graph with a large number of models, some authors proposed to organize the database of graphs such that the number of error-correcting subgraph isomorphism searches can be reduced ([286, 55]).

2.3. Relational structures

2.3.1. Relational description

Let O_A be an object, and A be the set of its parts or important features. An *N-ary relation* R over A is a subset of a Cartesian product: $R \subset A^N = A \times ... \times A$.

Definition 2.10 A *relational description* ([292, 293]) D_A is a set of *relations* $D_A = \{R_1, ..., R_I\}$, where for each $i = 1, ..., I$, there exists a positive integer n_i with

$R_i \subseteq A^{n_i}$. The relations R_i indicate various relationships among the parts of O_A.

Graphs are a special case of relational structures for $n = 2$, that means in graphs only binary (2-ary) relations between nodes are allowed.

2.3.2. Similarity measures for relational structures

Let A, B be the part sets of an object O_A and O_B, respectively. Let $R \subseteq A^N$ and $S \subseteq B^N$ be the two N-ary relations defined over part sets A and B, respectively, and $f : A \rightarrow B$ be a function that maps elements of set A into set B.

Definition 2.11 The *composition* $R \circ f$ of R with f is defined by:

$$R \circ f = \{(b_1, ..., b_N) \in B \mid \text{there exists } (a_1, ..., a_N) \in R \text{ with } f(a_i) = b_i, i = 1, ..., N\}$$

Definition 2.12 A *relational homomorphism* from an N-ary relation R to an N-ary relation S is a mapping $f : A \rightarrow B$ that satisfies $R \circ f \subseteq S$ (e.g. see Fig. 2.3). A *relational monomorphism* from an N-ary relation R to an N-ary relation S is a relational homomorphism that is one-one. A *relational isomorphism* f from an N-ary relation R to an N-ary relation S is a one-one relational homomorphism from R to S, and f^{-1} is a relational homomorphism from S to R.

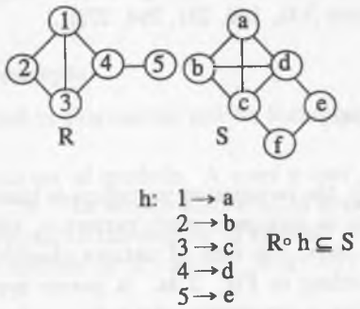


Fig. 2.3. Relational homomorphism h from binary relation R to binary relation S (ref. [292])

Rys. 2.3. Homomorfizm relacyjny h z binarnej relacji R do relacji S (ref. [292])

Let $D_A = \{R_1, ..., R_I\}$ and $D_B = \{S_1, ..., S_I\}$ be two relational descriptions of objects O_A and O_B with part sets A and B, respectively. It is assumed that $|A| = |B|$ (if this is not the case, enough dummy parts are added to the smaller set). Let f be any one-one, onto mapping from A to B.

Definition 2.13 The *structural error* of f for the i-th pair of corresponding relations (R_i and S_i) in D_A and D_B, is given by ([292, 293]):

$$E_s^i(f) = |R_i \circ f - S_i| + |S_i \circ f^{-1} - R_i|$$

Definition 2.14 The total error of f with respect to D_A and D_B is the sum of the structural errors for each pair of corresponding relations:

$$E(f) = \sum_{i=1}^{I} E_s^i(f)$$

Definition 2.15 The *relational distance* ([292, 293]) between D_A and D_B is then given by:

$$GD(D_A, D_B) = min_{f:A \to B}\ E(f)$$

where f is one-one, onto mapping from A to B.

That is, the relational distance is the minimal total error obtained for any one-one, onto mapping f from A to B. A mapping f that minimizes total error is called *a best mapping* from D_A to D_B.

2.3.3. *Algorithms for relational matching*

Relational matching is the process of comparing two relational descriptions to determine correspondence between their part sets, and to decide based on this correspondence, how similar they are. The purpose of object recognition using relational descriptions is to find the best match between an unknown relational description and a set of stored relational models. In this case, the stored model that has the least relational distance to the unknown description is the best match. The only known algorithms that can solve arbitrary relational matching problems employ a *backtracking tree search* ([292]). Such a tree search algorithm has been improved upon by adding nonheuristic methods for pruning the tree (for example [145, 146, 291, 294, 273]).

3. Syntactic methods

In syntactic approach ([57]), the recognition procedure is based on the concept of formal language *parsing*, i.e. given an unknown input pattern x, and N grammars $G_1, ..., G_N$, each describing a pattern class, the task of pattern classification can be solved by a recognition procedure according to Fig. 2.4a. A *parser* symbolized by a rectangle in Fig. 2.4a tries, given a string x representing an unknown input pattern, to construct a *derivation tree* of x according to G_i, for $i = 1, .., N$. That is, a parser checks whether an unknown input pattern x is in accordance with the rules of grammar G_i that describe all members of a pattern class. If does not exist any G_i such that $x \in G_i$, then the input pattern is rejected. The *derivation tree* of an unknown input pattern x provides not only a decision about its class, but also structural information describing how x is hierarchically built up from its primitive constituents.

There are problems with syntactic approach when patterns which are corrupted by noise and distortions are involved. The "pure"concepts from the theory of formal languages and abstract automata are not powerful enough to handle the majority of "real-world"applications, where distorted patterns with a greater degree of complexity occur. An approach used successfuly in many applications, is an *error-correcting parsing* (*ec*

parsing) [333]. The resulting approaches, however, do not fall into the categories of formal language theory any longer, and lack, therefore, a strong theoretical basis. A recognition procedure based on error-correcting parsing is shown in Fig. 2.4b.

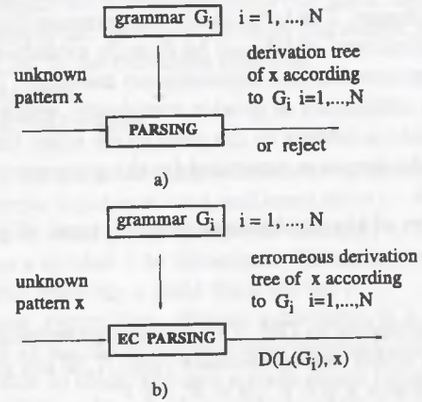


Fig. 2.4. Block diagram of syntactic (a), and ec syntactic parsing (b)

Rys. 2.4. Schemat blokowy analizy syntaktycznej (a), oraz analizy syntaktycznej z korekcją błędów (b)

3.1. String grammar

3.1.1. *Fundamental concepts*

All the formalisms presented in this section can be found in the reference literature (for example [11]).

An *alphabet* A is a finite set of symbols. A *word* x over A is a sequence of symbols $x = a_1...a_n, a_i \in A; i = 1, ..., n$. The empty word ϵ is the sequence with no symbols. The *length* $|x|$ of a symbol x is equal to the number of symbols contained in it. The set of all words over an alphabet is denoted by A^*, and $A^+ = A^* - \{\epsilon\}$.

Definition 2.16 A *string grammar* is a four-tuple:

$$G = (N, T, P, S)$$

where N is a finite set of *nonterminal* symbols, T is a finite set of *terminal* symbols ($N \cap T = \emptyset$), P is a finite set of *productions*, $S \in N$ is the starting symbol. The union $V = N \cup T$ is called a *vocabulary*. Each production $p \in P$ is of the form $\alpha \to \beta$, where $\alpha \in V^+$ is the *left-hand side*, and $\beta \in V^*$, the *right-hand side* of a rule, respectively.

Definition 2.17 Let G be a grammar and $\alpha \to \beta$ be a production in P. Any word $v = x\alpha y$ with $x, y \in V^*$ can be *derived* into the word $w = x\beta y$. We write $v \to w$ in this case, and $v \overset{*}{\to} w$ if there exist words $v = v_0, v_1, ..., v_n = w$ such that $v_i \to v_{i+1}$, for $i = 0, 1, ..., n-1$.

Definition 2.18 The *language* generated by a grammar G is given by $L(G) = \{x | x \in T^*, S \xrightarrow{*} x\}$.

Formal grammars in syntactic pattern recognition are used for the generation of structural representations of images. The *terminals* of the grammar correspond to the primitive pattern constituents (primitives) which can be directly extracted from an input pattern by means of suitable preprocessing and segmentation methods. The set of grammar *non-terminals* correspond to subpattern of greater complexity, which are hierarchically built up from primitive elements according to the description rules, that means *productions* of the grammar. Finally, the *language* generated by the grammar represents a whole class of patterns.

According to the form of the production, different types of grammars can be distinguished.

Definition 2.19

- A grammar according to Def. 2.16 is called *unrestricted* (*type 0*).
- A grammar is called *context-sensitive* (*type 1*), iff any production is of the form $xAy \rightarrow xzy$, where $x, y \in V^*, A \in N, z \in V^+$.
- A grammar is called *context-free* (*type 2*) iff any production is of the form $A \rightarrow z$, where $A \in N, z \in V^+$.
- A grammar is called regular (*type 3*), iff any production is of the form $A \rightarrow aB$ or $A \rightarrow a$, where $A, B \in N, a \in T$.

Definition 2.20 A context-free grammar is in *Chomsky normal form* (CNR) if each of its productions is either of the form $A \rightarrow BC, where A, B, C \in N$, or of the form $A \rightarrow a, with A \in N, a \in T$.

Definition 2.21 Let $G = (N, T, P, S)$ be a context-free grammar. A *derivation tree* is a tree where:

1. each node is labeled with a symbol $z \in V$ such that: 1) each leaf is labeled with a symbol $a \in T$, 2) each non-leaf is labeled with a symbol $A \in N$, 3) the root is labeled with the initial symbol S;
2. if there exists a node with label $A \in N$ such that its successor nodes are labeled with $x_1, ..., x_n \in V$, then there exists a production $A \rightarrow x_1 ... x_n$ in P.

The task of deciding if $x \in L(G)$ is equivalent to the construction of a derivation tree. This problem is also called *parsing* of x according to G. There are two principal classes of parsing algorithms. In *top-down* parsing, the derivation tree is constructed from the root towards the leaves. Starting with the inital symbol of the grammar, successive application of productions is tried until x is generated. In *bottom-up* parsing, the derivation tree is built from the bottom towards the root by rule application in reverse order ([11]).

An *automaton* is a formal tool allowing one, to analyze (recognize), whether a string belongs to a language generated by a given grammar.

Definition 2.19 A *finite state automaton with output* A is a six-tuple:

$$A = (Q, \Delta_1, \Delta_2, \delta, q_0, F)$$

where:
Q is a finite set of states,
Δ_1 is a finite nonempty set of input symbols (an *input alphabet*),
Δ_2 is a finite nonempty set of output symbols (an *output alphabet*),
$\delta : Q \times \Delta_1 \rightarrow Q \times \Delta_2$ is the transition function,
$q_0 \in Q$ is the initial state,
$F \subset Q$ is the set of final states.

The transition function δ describes how the automaton changes from one state to another. Let us assume that for some input symbol a and some state $q : \delta(q, a) = (q', x)$. It means that, if the automaton is in state q and it receives the input a, then the automaton goes to the state q' and writes a symbol x to its output. The automaton starts its work in the state q_0, and it stops after reaching a state from the set F.

In a syntactic pattern recognition, *formal automaton* is a formal model of image analysis. If we construct an automaton accepting a language representing a set of images of our interests, we are able to check whether a given image belongs to this set or not. An automaton is a theoretical formalism, in fact, a mathematical structure. Implementations of automata defined for certain classes of grammars gives us algorithms/programs called *syntax analyzers*, or *parsers*, recognizing languages generated by these grammars.

In many real applications, however, there are noise and distortions corrupting the patterns under consideration. If there is neither knowledge about relative frequencies, nor about possible distortions available, we can establish some general implicit assumptions about possible errors in patterns, and use *error-correcting parsing algorithm* [56]. The basic idea is to find, given a grammar G_i and a word x not in $L(G_i)$, the *distance*, or *similarity* $D(L(G_i), x)$ between a word x and $L(G_i)$ (that means to find that string $y \in L(G_i)$ which most resembles x), and construct its (*errorneous*) *derivation tree* according to G_i.

3.1.2. Distance measure from a language to a string

Consider a two class case. If an input string x is deformed, or garbled by noise (errorneous), the parsing technique cannot decide whether x belongs to the class 1 or 2. In this case, we have to know whether x is closer to $L(G_1)$, or $L(G_2)$. To realize this, a distance $D(L(G))$ from $L(G)$ to x have been defined. An algoritm to compute $D(L(G))$ is called an *error-correcting parser* or *recognizer*.

Definition 2.22 Let I be an input string, J - a string in $L(G)$ closest to I. An *error-correcting parsing problem* ([334, 333]) is a problem of finding the closest string(s) J to I, to make syntactic tree(s) of J, and to determine the distance, or similarity between I and J.

Let $X = x_1 ... x_m$ and $Y = y_1 ... y_n, m, n \geq 0$ be the two finite strings of symbols from a given alphabet and define M_s, a mapping from X to Y which prohibit one-to-many, many-to-one and cross-mappings. Let u_s be the number of elements (i, j) in M_s such

that $x_i \neq y_j$. Let $v_s = n - |M_s|$ and $w_s = m - |M_s|$, where $|M_s|$ denotes the number of elements in M_s (u_s, v_s, w_s are considered as the number of substitutions, insertions of extra symbols and of deletions to transform X to Y, respectively).

Definition 2.23 The *one-dimensional weighted Levenshtein distance* (1WLD) from X to Y, denoted by $D(X,Y)$, is defined by ([334, 333]):

$$D(X,Y) = min_s\{p \star u_s + q \star v_s + r \star w_s\}$$

where p, q, r are non-negative weights assigned to a substitution, an insertion, and a deletion, respectively. $D(X,Y)$ can be computed in the following way:

$$d(i,j) = min\{d(i-1,j) + r,\ d(i-1,j-1) + p(i,j),\ d(i,j-1) + q\}$$

where:

$d(i,0) = i \star r$,

$d(0,j) = j \star q$

$$p(i,j) = \begin{cases} p & if\ \ x_i \neq y_j \\ 0 & otherwise \end{cases}$$

$$D(X,Y) = d(m,n)$$

The computational complexity of computing $D(X,Y)$ is proportional to mn.

Definition 2.24 The 1WLD from a language $L(G)$ to a string I ($D(L(G),I)$) is defined as follows ([334, 333]):

$$D(L(G),I) = min\{D(\xi,I) | \xi \in L(G)\}$$

3.1.3. *Error-correcting parser for context-free languages*

Parsers can be classified into following four categories:

1. Parsers with pushdown lists ([11])
2. Direct parsers ([336])
3. Transition diagram parsers ([125])
4. Tabular parsers ([11]).

Parsers with pushdown lists ("backtracking parsing"[11]) are the reduced versions of direct parsers. The time and space complexities of both parsers are $O(c^n)$ and $O(n)$, respectively, for a string of length n, where c is a constant. Transition diagram parsers are usually used for a deterministic context-free language without cycles and left-recursion. Those three kinds of parsers are hard to extend to error-correcting parsers. Only *tabular* parsers have error-correcting versions. Here we will concentrate on tabular parsers of a general context-free language, because error-correction is indispensable to syntactic pattern recognition.

The bottom-up correcting parsers for a context-free languages were proposed in [125, 334, 373], while a top-down error-correcting parsers (based on Earley parser ([93])) were proposed in [9, 217]. In [335] the error-correcting parser which is an extension of the parser

for general context-free languages ([138]) has been proposed. This parser unifies both the top-down and bottom-up error-correcting parsers proposed in [217] and [334, 373], respectively. Ec parsers take much computing time and memory, so attempts are made to construct a high speed ec parser.

The *bottom-up error-correcting parser* for a context-free languages which is an extension of CKY (Cocke-Kasami-Younger) parser [11], is shown in Fig. 2.5 ([334]). The input to the algorithm are: $G = (N,T,P,S)$ in CNR, input string $I = a_1...a_n$, and weights p, q, r, the output is the parse table Ta. The space and time complexities to construct a parse table Ta are $O(n^2)$ and $O(n^3)$, respectively.

Define $h(A) = min\{|\alpha| \star r | A \xrightarrow{*} \alpha, \alpha \in T^*\}$, where $|\alpha|$ is the length of one of the shortest strings derived from nonterminal A. Let W_{max} be a predetermined threshold. Define the operation $t[i,j] \cup \{(A,w)\}$ as follows:

1. If $(A,w') \in t[i,j]$ and $w' \leq w$ do nothing.
2. If $(A,w') \in t[i,j]$ and $w' > w$, replace (A,w') with (A,w).
3. Otherwise, if $w \leq W_{max}$, add (A,w) to $t[i,j]$.

The operation described in lines 3-7 in Fig. 2.5 treats substitution errors. If $b = a_i$ and $A \rightarrow b$ is in P, $(A,0)$ is added to $t[i,1]$. This corresponds to tree T_1 in Fig. 2.5a. If $b \neq a_i$, (A,p) is created in $t[i,1]$. This is the hypothesis that a_i is substituted for b (Fig. 2.5b). Deletion errors are treated by the operations from lines 6-11 and 24-29. If $A \rightarrow BC$ in P and (B, w_B) is in $t[i,j]$, $(A, w_b + h(C))$ is created in $t[i,j]$. This is the hypothesis that α is deleted. Here α is one of the shortest strings such that $C \xrightarrow{*} \alpha(\alpha \in T^*)$ and $h(C) = |\alpha| \star r$ (Fig. 2.5c). Another case is illustrated in Fig. 2.5d. The operation from lines 30-33 treats insertion errors. If (A, w_A) is in $t[i,j]$, $(A, w_A + q)$ is added to $t[i,j+1]$ and $t[i-1,j+1]$. This means that a_{i+j+1} and a_{i-1} are assumed to be inserted (Fig. 2.5e and f).

3.2. Generalizations of string grammar

3.2.1. *Attributed grammar*

Attributed grammar ([11, 375]), being an extension of a string grammar, provide a solution to the problem of representing quantitative information such as length, or orientation. The idea is to augment each grammar symbol $Y \in V$ by a vector of attribute values: $m(Y) = (x_1, ..., x_k)$, where an attribute a is a function $a : Y \rightarrow D_Y$ mapping a symbol $Y \in V$ into a domain D_Y of numerical values. An attribute vector can be interpreted as a numerical feature vector in the sense of statistical pattern recognition. Considering a production $A_1...A_n \rightarrow B_1...B_m, A_i, B_j \in V$, there is usually a relationship between the attributes of the symbols in the left- and right-hand sides:

$$a(A_i) = f_i(a(B_1), ..., a(B_m)); i = 1, ..., n \ (1) \text{ or}$$
$$a(B_i) = g_i(a(A_1), ..., a(A_n); i = 1, ..., m \ (2)$$

In bottom-up recognition, one starts with the segmented pattern with all the primitives extracted in such a way that the attributes of the non-terminal symbols can successively be computed during analysis according to equation (1). Conversely, in top-down recognition, attribute values for the grammar start symbol must be known, and are successively inherited to other grammar symbols according to equation (2).

3.2.2. Programmed grammars

A *programmed grammar* differs from the conventional one in that each production rule has associated two fields: a success field $S(U)$, and a failure field $F(U)$, telling us which production rule is to be applied next.

Definition 2.25 A *programmed grammar* ([278]) is a five-tuple:

$$G = (N, T, J, P, S)$$

where:
N, T, S have the same meaning as in the Def. 2.16,
J is a set of productions labels,
P is a finite set of productions of the form $(r)\ \alpha \rightarrow \beta\ S(U)\ F(W)$ in which:
$\alpha \in V^*NV^*, \beta \in V^*$, $(r) \in J$ is the production label,
$U \subset J$ is the *success field*, $W \subset J$ is the *failure field*.

Definition 2.26 The *derivation* with a programmed grammar is defined in the following way ([278]). A production labelled with (1) is applied first. If it is possible to apply a production (r), then this production is applied, and the next production is chosen from the success field U of the r-th production. Otherwise, the next production from the failure field W of the r-th production is chosen.

A context-free programmed grammar allows one to generate much wider class of languages than a "pure"context-free grammar ([336]). However, parsers proposed for this kind of grammar are of non-polynomial complexity, since they are based on a backtracking derivation ([336]).

3.2.3. Higher-dimensional extensions

Since the only possible relation between the symbols in a string is a concatenation relation, string grammar is a very restricted scheme for two-dimensional, or even higher-dimensional patterns. There are two posible extensions leading to more effective representation of higher-dimensional patterns. The first, is the representation formalism which is intrinsically more dimensional, like arrays, trees or graphs. The second possible extension is to incorporate into a string representation relations which are more general than concatenation.

In *Picture Description Language* (PDL) ([297]) a set of primitives (terminal symbols) is not determined. The only assumption concerning a primitive is that it has a "tail"and a "head". There are four ways of joining a pair of primitives. These four binary operations

are denoted by "+"(head of one primitive linked to tail of the second primitive), "-"(primitives linked at their heads), "x"(primitives linked at their tails), and "*"(primitives linked at their tails and heads). Each entity resulting from linking of two primitives has a tail and a head, thus binary operations can be extended to higher level structures, i.e. more complex PDL expressions can be used as arguments of the operators. The language generated by a grammar G consists of a number of PDL expressions, each describing an individual pattern. A whole pattern class is represented by one grammar.

The main drawback of PDL is its limitation to only two concatenation points of subexpressions including primitives. This limitation is overcome by *plex structures* ([105]). A symbol with n attaching points is called an *n-attaching point entity* or *nape*. Structures formed by joining napes are called *plex structures*. Formally, a nape is represented by an identifier and a list of its attaching points. A plex structure is described by three components, a list of napes, list of internal connections between napes, and a list of attaching points, where the plex structure can be joined with other napes, or plex structures. A string grammar generating a set of plex structures is called a *plex grammar*. In such a grammar, the napes correspond to the terminals.

The main feature of *Jakubowski shape feature language* ([163, 166, 168]) is its hierarchical, multilevel structure. The shape primitives detected in a contour on the lowest level, are transformed by a sequential transducer into higher-level descriptions (*biquads, regions, grooves*) which can be used for recognition of certain complex properties of contours. Contours described in such a way are swept in 3D space defining 3D objects relating to parts modeled in CAD systems, and recognized in CAM systems ([165, 167]).

3.3. Graph grammar

Among all grammars, the *graph grammars* are the most general description formalism for representing image structures. The theory of graph grammars delivers a variety of types of graph rewriting formalisms ([95, 96, 97]). However, for only few classes of graph grammars efficient automata and parsers have been defined. This fact results from the NP-complete, or even NP-space membership problem for most non-trivial graph languages. Therefore, constructing efficient syntax analysis (parsing) algorithms is a key research problem in this field. There are two problems which make a construction of a syntax analyzer difficult. The first is in defining an *unambiguous* graph representation. The second is connected with the specific form of production of a graph grammar, especially with the *embedding transformation*. Some suggestions for solving this problem concerns graphs in a particular form ([298, 108]).

The first mathematical model for graph automata has been proposed in [40], another parsing algorithms for graph grammar have been described in [277, 45, 298, 50, 256]. A series of efficient parsers for the edge-labelled directed Node Label Controlled graph grammar (*edNLC*) which generates EDG-graphs (a directed node edge-labelled graph) have been published in [108, 109, 110]. Both a parsing algorithm ([108]), and an error-corecting parsing algorithm ([109]) have been proposed for the *ETL(1)*, embedding transformation-preserved linear subfamily of edNLC graph grammars ($o(n^2)$). On the other hand, ETL/1

grammars have too weak generative power to define languages representing pictures, in which some context relations among primitives have to be expressed. Therefore a restricted subfamily of edNLC graph grammars, an embedding transformation-preserved production-ordered k-left nodes unambiguous (*ETPL(k)*) graph grammars, which generates a wider class of languages have been proposed in [110].

Definition 2.27 An edge-labelled directed node-label controlled, *edNLC graph grammar* is a quintuple ([171]):

$$G = (\Sigma, \Delta, \Gamma, P, Z)$$

Σ is a finite, non-empty set of node labels,
Γ is a finite non-empty set of edge labels,
P is a finite set of productions of the form (l, D, C), in which:
$l \in \Sigma$,
D is a graph having labelled nodes and edges that are both labeled and directed,
$C : \Gamma \times \{in, out\} \rightarrow 2^{\Sigma \times \Sigma \times \Gamma \times \{in,out\}}$ is the embedding transformation,
Z is the starting graph, called axiom.

Definition 2.28 The definition of the *embedding transformation* for some production in the folowing way ([172]):

$$C(x, in) = \{(a, c, y, out)\}$$

means that:

- (1) each edge labeled by x and coming into the node of the left-hand side of the production ought to be replaced by:
- (2) the edge;
- (a) connecting the node of the graph of the right-hand side of the production and labeled by a with the node of the rest-graph, i.e. the derived graph without the removed node corresponding to the left-hand side of the production, and labeled by c,
- (b) labeled by y,
- (c) and going out from the node a.

A formal definition of a *derivation* for edNLC grammar is presented in [171], we introduce it in an intuitive way. A direct *derivation step* consisting of applying a production of the form (l, D, C) is divided into 2 steps:

1. A node labeled by l of a derived graph is replaced by a graph D.
2. The graph D is embedded in the rest of the graph according to the transformation C.

Let us consider for example a production whose left- (l) and right-hand (D) sides are depicted in Fig. 2.7a), and the embedding transformation is of the form ([111]):

$$C(\alpha, in) = \{(b, a, \alpha, in)\}$$
$$C(\gamma, out) = \{(d, E, \alpha, in)\}$$

Applying this production to the graph g shown in Fig. 2.7b), firstly a node labeled by A is replaced by a graph D. At the second step, we use the transformation C to embed D in a way described above (see Fig. 2.7c)).

In [109] an *error-correcting parser* ($o(n^2)$) for ETL/1-graph grammar based on the *distance measure* between two graphs generated by this grammar has been defined. The distance measure between two graphs has been defined based on node and edge distortions. Before constructing an algorithm of error-correcting syntax analysis, a considered grammar is modified. The grammar is not modified as in most error-correcting parsing algorithms, by adding error transformations in the form of production rules. Instead, it is modified by joining a description of all the *acceptable distortions* of a right hand graph to each production ([109]).

4. Pattern recognition using structural distances

In this section it will be shown how the structural distances described in the previous sections of this chapter (string, graph, relational distances, or distance from a language to a string) can be applied in pattern recognition. We describe it for the case of strings, but the concepts can be applied to graphs or relational structures if we use graph or relational distance instead of string distance.

Let $P_1 = \{x_1^1, ..., x_N^1\}$ and $P_2 = \{x_1^2, ..., x_N^2\}$ be sets of prototype strings representing pattern classes C_1 and C_2, respectively, and x be a string representing an unknown pattern.

A *decision procedure* for the classification of x can be based for example on the *nearest neighbor classification (NN)* ([67]). Let $D_1(x)$ and $D_2(x)$ be the distances between x and that elements in C_1 and C_2, respectively, which are closest to x:

$$D_1(x) = min\{d(x_i^1, x) : i = 1, ..., N\}$$
$$D_2(x) = min\{d(x_i^2, x) : i = 1, ..., M\}$$

An *NN decision rule* (modified by introducing a threshold t for rejection) is given by:

$$x \in \begin{cases} C_1 & iff \;\; D_1(x) \leq D_2(x) \wedge D_1(x) \leq t \\ C_2(x) & iff \;\; D_2(x) < D_1(x) \wedge D_2(x) \leq t \\ reject & otherwise \end{cases}$$

A straightforward extension of the above decision rule is the *k-NN classification rule* ([74, 122, 295, 115]).

Besides classification, the defined string distance can also be applied to the *clustering* of strings ([119, 162, 149, 13, 19, 176]).

```
1   begin
2   for i = 1 to n do begin
    * match or substitution *
3   for each A → b ∈ P do
4      if b = a_i then t[i,1] := [i,1] ∪ {(A,0)}
5      else t[i,1] := t[i,1] ∪ {(A,p)}
6   repeat * deletion *
7      for each A → BC ∈ P do begin
8         if ((B,w) ∈ t[i,1]) * C is deleted *
          then t[i,1] := t[i,1] ∪ {(A, w + h(C))}
9         if (C,w) ∈ t[i,1]) * B is deleted*
          then t[i,1] := t[i,1] ∪ {(A, w + h(B))}
10     end
11  until * no new items has been generated from lines 7-10 *
    * insertion *
12     for each (A,w) ∈ t[i,1] do begin
13     if i > 1 then t[i-1,2] := t[i-1,2] ∪ {(A, w + q)}
14     if i < n then t[i,2] := t[i,2] ∪ {(A, w + q)}
15     end
16  end *for i *
17  for j = 2 to n do begin
18  for i = 1 to n - j + 1 do begin
19     for k = 1 to j - 1 do begin * complete *
20        for each A → BC ∈ P do
21           if ((B,w_B) ∈ t[i,k] and ((C,w_C) ∈ t[i+k, j-k]
22              then t[i,j] := t[i,j] ∪ {(A, w_B + w_C)}
23           end
24     repeat * deletion *
25        for each A → BC ∈ P do begin
26           if ((B,w) ∈ t[i,j]) * C is deleted *
             then t[i,j] := t[i,j] ∪ {(A, w + h(C))}
27           if ((C,w) ∈ t[i,j]) * B is deleted *
             then t[i,j] := t[i,j] ∪ {(A, w + h(B))}
28        end
29     until * no new items has been generated from lines 25 -28 *
    * insertion *
30        for each (A,w) ∈ t[i,j] do begin
31           if i > 1 and j < n then t[i-1, j+1] := t[i-1, j+1] ∪ {(A, w + q)}
32           if i < n and j < n then t[i, j+1] := t[i, j+1] ∪ {(A, w + q)}
33        end
34  end
35  end
36  if (S,w) ∈ T[1,n] then I is accepted with weight w
37  else I is rejected
38  end
```

Fig. 2.5. An algorithm for bottom-up error correcting parsing

Rys. 2.5. Algorytm analizy syntaktycznej z korekcją błędów typu z dołu - do góry

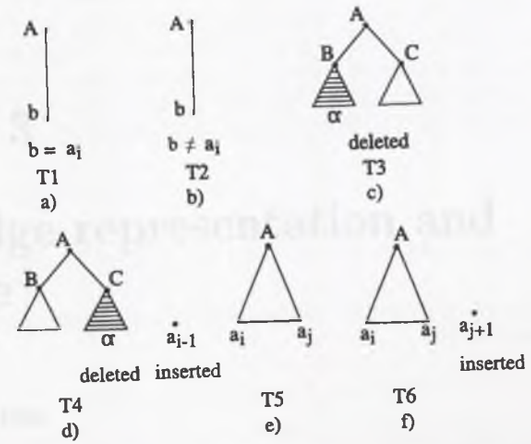


Fig. 2.6. Sub-syntactic trees corresponding to the operations in a bottom-up error-correcting parser

Rys. 2.6. Drzewa syntaktyczne odpowiadające operacjom w analizie syntaktycznej typu z góry - na dół, z korekcją błędów

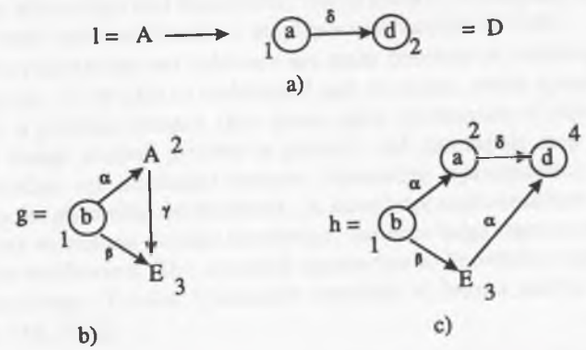


Fig. 2.7. The illustration of the derivation in edNLC grammar: a) left- (l) and right-hand (D) sides of a production of edNLC grammar, b) an input graph, c) the result of using a transformation C (ref. [108])

Rys. 2.7. Ilustracja wyprowadzenia w gramatyce typu edNLC: a) lewa i prawa strona produkcji w gramatyce edNLC, b) graf wejściowy, c) wynik zastosowania transformacji C (ref. [108])

Chapter 3

Knowledge representation and inference

1. Introduction

Knowledge is a symbolic representation (a model) of aspects of some named universe of discourse [116]. In order to achieve automatic description of patterns, generally two kinds of knowledge are distinguished: *declarative knowledge* and *procedural knowledge*. *Declarative knowledge* is a problem-specific knowledge, that means a knowledge about objects, events which should be modelled. A knowledge about a use of such declarative models build up an *inference process*. This is a *procedural knowledge*. Each of these types of knowledge has advantages and limitations, but in general, it is agreed that both types are needed and most systems, therefore, employ a combination of both.

Knowledge representation and inference are main concerns in building systems with artificial intelligence. To be able to understand and to reason, vision systems needs prior knowledge about a problem domain, that means prior information of objects to be seen. Since document image analysis systems in general, and geographic map image analysis systems in particular, are intelligent systems, knowledge representation then plays an important role in its successful development. A *knowledge representation* formalism consists of a structure to express domain knowledge, the *knowledge representation language*, and the *inference mechanism*. The principal approaches to knowledge representation are based on: 1) formal logic, 2) rules, 3) semantic networks, 4) frames, and 5) object-oriented paradigm ([250, 268, 325]).

2. Classical first-order logic

Formal logic is a classical artificial intelligence approach to knowledge representation and inference. There are several logic calculi in artificial intelligence. Presently, *first-order*

predicate calculus plays the most prominent role. The use of predicate calculus for knowledge representation is based on the idea of expressing knowledge about the actual problem domain by means of *formulas* which are constructed according to a particular syntax. In predicate calculus, logical conclusions can be inferred from formulas by means of *inference rules*, of which the most important is the *resolution rule*. Given a set of formulas, the axioms, and another formula, the theorem, the resolution rule can successively be applied in order to find out if the theorem *logically follows* from the axioms.

2.1. Syntax

All the formalisms presented in this section can be found in the reference literature ([213, 130, 209]).

An *alphabet of classical first-order logic* consists of primitive symbols from the following pairwise disjoint classes:

- A denumerable set V of *individual variables* $\{x_1, x_2, ...\}$.
- One truth-constant: 'True'.
- Sentential connectives: '$\neg$' (negation) and '$\rightarrow$' (implication).
- One quantifier: '$\forall$' (universal quantifier);
- Punctuation marks: '(',')', and ','.
- A countable set P of *predicate constants*, each of which has uniquely assigned non-negative integer, called arity of P; 0-ary predicate constants are called *proposition constants*.
- A countable set F of *function constants* each of which has uniquely assigned non-negative integer, called arity of f; 0-ary function constants are called *individual constants*.

Every alphabet AL of first-order logic uniquely determines three classes of expressions: *terms, atomic formulae* and *formulae*.

Definition 3.1

- The set TM_{AL} of *terms over AL* is the smallest set such that: 1) all individual variables and constants from AL are members of TM_{AL}, 2) If $\alpha_1, ..., \alpha_n \in TM_{AL}(n \geq 1)$ and f is n-ary function constant from AL, then $f(\alpha_1, ..., \alpha_n) \in TM_{AL}$.
- The set $AFORM_{AL}$ of *atomic formula over AL* is the smallest set such that: 1) 'True' and all proposition constants from AL belong to $AFORM_{AL}$; 2) If $\alpha_1, ..., \alpha_n \in TM_{AL}(n \geq 1)$ and P is n-ary predicate constant from AL, then $P(\alpha_1, ..., \alpha_n) \in AFORM_{AL}$.

- The set L_{AL} of *formulae over AL* (called also the *first-order language over AL*) is the smallest set such that: 1) If $A \in AFORM_{AL}$, then $A \in L_{AL}$; 2) If $A, B \in L_{AL}$ and x is a variable, then $(\neg A) \in L_{AL}, (A \rightarrow B) \in L_{AL})$ and $(\forall x(A)) \in L_{AL}$.

For convenience, additional sentential connectives: *disjunction* $\vee$, *conjunction* $\wedge$, *equivalence* $\equiv$, and *existential quantifier* $\exists$, as well as a truth-constant $False = \neg True$ are introduced:

1. $(A \vee B) =_{DF} ((\neg A) \rightarrow B)$
2. $(A \wedge B) =_{DF} ((\neg((\neg A) \vee (\neg B)))$
3. $A \equiv B =_{DF} ((A \rightarrow B) \wedge (B \rightarrow A))$
4. $\exists x =_{DF} (\neg(\forall x(\neg(A))))$

2.2. Semantics

Definition 3.2 *A frame for a first-order language* L_{AL} is a pair $M =< D, m >$, where: D is a non-empty set, called the *domain* of M, m is a function which assigns to each n-ary predicate constant from AL, an n-ary relation on D, and to each n-ary function symbol from AL, an n-ary function from D^n into D.

Definition 3.3 Let M be a first-order frame for L_{AL}, $|M|$ denotes the domain of M, $M|K|$ stands for $m(K)$, K is a predicate or function constant in AL, $As(M)$ denotes *a set of assignments over M*, i.e. the set of all functions from the set of variables into $|M|$, $[a]_x = \{a' : a' \in As(M) \wedge a'(y) = a(y), \forall y \neq x\}$. The *value* $V_a^M(\alpha)$ of a term $\alpha \in TM_{AL}$ *in M wrt* $a \in As(M)$ is an element from $|M|$ defined by the following recursion on α:

1. $V_a^M(x) = a(x)$
2. $V_a^M(a) = M|a|$
3. $V_a^M(f(\alpha_1, ..., \alpha_n)) = M|f|(V_a^M(\alpha_1), ..., V_a^M(\alpha_n))$.

Definition 3.4 The *truth-value* $V_a^M(A)$ of a formula $A \in L_{AL}$ *in M wrt* $a \in As(M)$ is an element of $\{0, 1\}$ defined as follows:

1. $V_a^M(True) = 1$
2. $V_a^M(p) = M|p|$
3. $V_a^M(P(\alpha_1, ..., \alpha_n)) = M|P|(V_a^M(\alpha_1), ..., V_a^M(\alpha_n))$
4. $V_a^M(B \rightarrow C) = 1$ iff $V_a^M(B) = 0 \vee V_a^M(C) = 1$
5. $V_a^M(\neg B) = 1 - V_a^M(B)$
6. $V_a^M(\forall x B) = 1$ iff $V_a^M(B) = 1$ for every $a' \in [a]_x$.

Definition 3.5 A formula A is said to be *satisfied in a frame M by an assignment* $a \in As(M)$ iff $V_a^M(A) = 1$. A is *satisfiable* if $V_a^M(A) = 1$, for some frame M and some $a \in As(M)$; otherwise A is called *unsatisfiable*. A is *true in M* iff $V_a^M(A) = 1$, for any $a \in As(M)$. If A is true in M, then M is called *a model of A*. A is *logically valid* iff A is true in every frame M.

2.3. First-order theory as a deduction system

Definition 3.6 A *first-order theory* is a pair $T =< L, S >$, where L is a first-order language, and S is a computable set of sentences from L (*axioms*). If $S = \{\}$, then T is called the *predicate calculus over L*. A frame M is a *model of a theory T* iff all axioms of T are true in M.

Definition 3.7 A formula A is a *semantic consequence of a theory T* , written $T \models A$, iff A is true in every model of T. If $T \models A$, then we say that A *(logically) follows from T*, or A *is entailed by T*. In particular, A is a semantic consequence of the predicate calculus (over L), written $\models A$ iff A is true in every frame (for L).

Definition 3.8 A *deduction system* for first-order logic is a triple $DS =< L, S, R >$, where L is a first-order language, S is a computable subset of L (logical axioms), $R = \{R_1, ..., R_n\}$ is a finite set of *inference rules*. Each R_i is a partial function from L^k into L ($k \geq 1$). If $A_1, ..., A_k$ are in the domain of R_i, then we say that R_i is applicable to $A_1, ..., A_k$, and $R_i(A_1, ..., A_k)$ is then called *a direct consequence* of $A_1, ..., A_n$ by virtue of R_i.

Definition 3.9 A formula A is *provable from T in* $DS =< L, S, R >$ or is a *syntactic consequence of T in DS* (alternatively, A is a theorem of T in DS), written $T \vdash_{DS} A$, iff there is a sequence $A_1, ..., A_m$ of formulae from L such that A is A_m and, for each $1 \leq i \leq m$, either $A_i \in S \cup T$ or A_i is a direct consequence of some preceding formulae by virtue of some rule from R. Such a sequence is called *proof of A from T in DS*.

Definition 3.10 A deduction system DS is *sound*, iff $\vdash_D SA$ implies that A is valid. DS is *complete* iff every valid formula from L is provable in DS.

Theorem 3.1 $\vdash A$ iff $\models A$.

Definition 3.11 A theory T over L is *consistent* iff there is a formula $A \in L$ such that $\neg(T \vdash A)$.

2.4. Clausal form

The resolution method applies only to formula in a special form, called the *clausal form*.

Definition 3.12 A formula A is said to be in *prenex conjunctive normal form* iff it is of the form:

$$Q_1 x_1 ... Q_n x_n [\forall_{i=1}^{m} \exists_{j=1}^{k(i)} A_{ij}]$$

where for each $1 \leq i \leq n, Q_i \in \{\forall, \exists\}$, $x_1, ..., x_n$ are distinct variables occuring in $[\forall_{i=1}^{m} \exists_{j=1}^{k(I)} A_{ij}]$; each A_{ij} is either an atomic formula or its negation. The part $Q_1 x_1 ... Q_n x_n$ is called the *prefix* of A, while the part $[\forall_{i=1}^{m} \exists_{j=1}^{k(i)} A_{ij}]$ is called the *matrix of A*.

Definition 3.13 A formula A is said to be in *prenex normal form* iff it is of the form $Q_1 x_1 ... Q_n x_n B$, where for each $1 \leq i \leq n, Q_i \in \{\forall, \exists\}$; $x_1, ..., x_n$ are distinct variables occurring in B, B contains no quantifiers.

Every formula A can be effectively transformed into its prenex conjunctive normal form by successively replacing subformulae of A by their equivalents using the following steps: 1) Elimination redundant quantifiers, 2) renaming variables, 3) elimination $\rightarrow$ and $\equiv$, 4) movement $\neg$ inward, 5) movement quantifiers to the left, 6) distribution $\wedge$ over $\vee$.

Literals are atomic formulae (*positive literals*), or their negations (*negative literals*). The symbol $|l|$ stands for the atomic formula of a literal l.

Definition 3.14 A *clause* is a formula C of the form: $l_1 \vee ... \vee l_n, n \geq 0$, where $l_1, ..., l_n$ are literals. If $n = 0$, then C is the *empty clause*, written $\square$.

Definition 3.15 A formula is said to be in a clausal form iff it is of the form:

$$\forall C_1 \wedge ... \wedge \forall C_n, n \geq 1$$

where $C_1, ..., C_n$ are clauses.

Any universal sentence (whose prenex normal form contains no existential quantifiers) can be effectively transformed into an equivalent sentence in the clausal form. A *clausal form* of a formula A can be obtained from A by the following construction:

1. Take the existential closure of A, put $A_1 = \exists A$.
2. Apply steps 1-4 from from the construction of prenex conjunctive normal form (formula A_2).
3. Move quantifiers to the right (formula A_3).
4. Eliminate existential quantifiers by repeating the following process, called *Skolemization*: take any subformula of the form $\exists y B(y)$. Suppose that $x_1, ..., x_n$ are all distinct free variables of $\exists y B(y)$ which are universally quantified to the left of $\exists y B(y)$. If $n = 0$, then replace $\exists y B(y)$ by by $B(a)$, where a is a new individual constant. If $n > 0$, then replace $\exists y B(y)$ by $B(f(x_1, ..., x_n)$, where f is any new n-ary function constant. The introduced symbols a, f are *Skolem functions*. (formula A_4).
5. Delete quantifiers from A_4 (formula A_5).
6. Distribute $\wedge$ over $\vee$ (formula A_6).
7. Eliminate $True$ and $False$ (formula A_7).
8. Add universal quantifiers $A_7 = C_1 \wedge ... \wedge C_n$, put $A_8 = \forall C_1 \wedge ... \wedge \forall C_n$, which is a desired clausal form.

2.5. Resolution rule

Since $\{A_1, ..., A_n\} \models B$ if $\models A_1 \vee ... \vee A_n \rightarrow B$, the problem of determining whether the formula B follows from theory $\{A_1, ..., A_n\}$, reduces to the task of determing whether the formula $(A_1 \vee ..., \vee A_n \rightarrow B)$ is valid. Since the formula is valid if its negation is unsatisfiable, one way to show validity of a formula A is to show unsatisfiability of $\neg A$. This is the approach taken by the resolution rule. If $\neg A$ is unsatisfiable (i.e. A is valid), the procedure will detect it and stop. Otherwise, the computation process may fail to terminate.

2.5.1. *Unification*

Definition 3.16

- A *substitution (of terms for variables)* is any set of the form $\{x_1 \leftarrow \alpha_1, ..., x_n \leftarrow \alpha_n\}$, where $x_1, ..., x_n$ are distinct variables, and each α_i is a term different from x_i. Given a simple expression E (literal or term), and a substitution $\vartheta = \{x_1 \leftarrow \alpha_1, ..., x_n \leftarrow \alpha_n\}$, we write $E\vartheta$ to denote the expression obtained from E by simultaneously replacing each occurrence of x_i by $\alpha_i, 1 \leq i \leq n$.
- The *composition* $\vartheta\tau$ of $\vartheta = \{x_1 \leftarrow \alpha_1, ..., x_n \leftarrow \alpha_n\}$, and $\tau = \{y_1 \leftarrow \beta_1, ..., y_m \leftarrow \beta_m\}$ is the substitution obtained from the set $\{x_1 \leftarrow \alpha_1\tau, ..., x_n \leftarrow \alpha_n\tau, y_1 \leftarrow \beta_1, ..., y_m \leftarrow \beta_m\}$, by deleting any element $x_i \leftarrow \alpha_i\tau$ for which $x_i = \alpha_i\tau$, and deleting any element $y_i \leftarrow \beta$ such that $y_i \in \{x_i, ..., x_n\}$.
- Given a non-empty set $EX = \{E_1, ..., E_n\}$ of simple expressions, a *unifier* ϑ of EX is any substitution for which $E_1\vartheta = ... = E_n\vartheta$. If EX has a unifier, then we say EX is *unifiable*. A unifier ϑ of EX is called a *most general unifier* of EX iff for each unifier τ of EX there is a substitution λ such that $\tau = \vartheta\lambda$.

2.5.2. *Basic resolution*

Definition 3.17 Let C_1, C_2 be the clauses such that no variable is common to both C_1 and C_2, $l_1, ..., l_n \in C_1$, $l_1', ..., l_k' \in C_2$. Assume that $l_1, ..., l_n$ are all positive, and $l_1', ..., l_k'$ negative, or vice versa. If ϑ is a most general unifier of $\{|l_1|, ..., |l_n|, |l_1'|, ..., |l_k'|\}$, then the clause $[C_1\vartheta - \{l_1\vartheta, ..., l_n\vartheta\}] \vee [C_2\vartheta - \{l_1'\vartheta, ..., l_k'\vartheta\}]$ is called a *binary resolvent* of C_1 and C_2.

A clause C_1 is said to be a *variant* of C iff $C_1 = C\vartheta$, where ϑ is a renaming substitution for C. A *renaming substitution* for a clause C is any substitution $\{x_1 \leftarrow y_1, ..., x_n \leftarrow y_n\}$ satisfying the following conditions: 1) $\{x_1, ..., x_n \subset V\}$ (V is the set of all variables occurring in C), 2) $y_1, ..., y_n$ are distinct variables, 3) $(V - \{x_1, ..., x_n\}) \cap \{y_1, ..., y_n\} = \{\}$.

Definition 3.18 A *resolvent of clauses* C_1 and C_2, written $RES(C_1, C_2)$, is any binary resolvent of a variant of C_1 and a variant of C_2.

Definition 3.19 The *resolution rule* is the following inference rule. For any clauses C_1 and C_2, $RES(C_1, C_2)$ is a direct consequence of C_1 and C_2.

Definition 3.20 Let CL be a set of clauses. A *refutation* from CL is a sequence $C_1, ..., C_n$ of clauses such that $C_n = \square$ and, for each $1 \leq i \leq n$, either $C_i \in CL$ or C_i is a resolvent of C_j and C_k, where $j, k < i$.

Definition 3.21 A *refutation tree* for a set CL of clauses is a binary 'upward oriented' tree D satisfying the following conditions:

1. All nodes of D are labelled by clauses.
2. Each leaf node of D is labelled by a clause from CL.
3. The root node of D is labelled by $\square$.
4. If C labels a non-leaf node, then C is a resolvent of the clauses which label the successors of n.

Definition 3.22 A *resolution proof* of a formula A from a theory T is any refutation from $CLAUSES(T) \cup CLAUSES(\neg\forall A)$, where $CLAUSES(T)$ denotes the set of clauses corresponding to an aribitrarily chosen clausal form of T.

Theorem 3.2 $T \models A$ iff there is a resolution proof of A from T.

2.5.3. *Linear resolution*

Automatic reasoning with the resolution requires a strategy of choosing clauses to be resolved. The most straightforward method, the *basic resolution*, amounts to systematically generating all possible resolvents. But using the resolution in an unconstrained manner leads to many useless inferences, making the basic resolution too ineffective to be of practical use. It can be substantially improved by putting various constraints on the clauses which are to be resolved at a given point of time. Some examples of such strategies are: *unit, input, linear, set of support, ordered, directed resolutions*, etc. ([130, 268, 59]).

One of these refinements, the linear resolution method ([213, 268]) is sketched below.

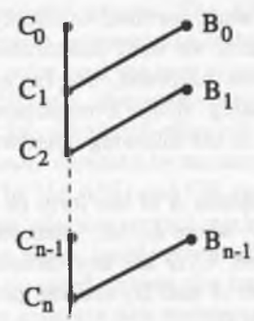


Fig. 3.1. Linear refutation

Rys. 3.1. Wywód klauzuli pustej w rezolucji liniowej

Definition 3.23 A *linear refutation from a set of clauses* CL has the form of Fig. 3.1, where:

1. $C_0 \in CL$; (C_0 is the *top clause of the refutation*);
2. for $0 \leq i \leq n, B_i \in CL$ or B_i is C_j, for some $j < i$;
3. for $1 \leq i \leq n, C_i$ is a resolvent of C_{i-1} and B_{i-1};
4. $C_n = \square$

2.5.4. *Resolution with equality*

The refutation completeness of resolution does not hold for databases involving the relation constant "="intended to be interpreted as the *equality relation*. There is simply no mechanism for substituting nonvariable terms that are known to be equal, so it is impossible to prove some results even though they are logically implied by the premises. The simplest way to circumvent this difficulty is to *axiomatize the equality relation*, and to supply appropriate substitution axioms. The set of equality clauses for CL, a given set of clauses in which "="occur, is the set consisting of the following clauses:

1. $x = x$
2. $x \neq y \vee y = x$
3. $x \neq y \vee y \neq z \vee x = z$
4. $x_1 \neq y_1 \vee ... \vee x_n \neq y_n \vee \neg(P(x_1, ..., x_n)) \vee P(y_1, ..., y_n)$, for each n-ary predicate constant P occurring in CL
5. $x_1 \neq y_1 \vee ... \vee x_n \neq y_n \vee f(x_1, ..., x_n) = f(y_1, ..., y_n)$, for each n-ary function constant f occurring in CL

Of course, for this approach to work, we need to supply substitution axioms for every function and relation within which we want substitutions to occur, which is tedious. Therefore, special rules have been proposed, like for example *paramodulation* ([269]), *resolution by unification and equality* ([75]), *E-resolution* ([239]), *demodulation* ([367]).

For example, *paramodulation* is the following inference rule.

Definition 3.24 For any two clauses A of the form $(a = b) \vee C$ or $(b = a) \vee C)$, and B of the form $L(...a...) \vee D$, where $L(...a...)$ denotes a literal L and the occurrence of a term a as an argument, C, D are any clauses, the clause $L(...b...) \vee C \vee D$ is a *paramodulant* of clauses A and B, and the process of its generation is called *paramodulation from A into B*:

$$[(a = b) \vee C, L(...a...) \vee D]/[L(...b...) \vee C \vee D]$$

Paramodulation rule, when added to the resolution principle, guarantees refutation completeness, even in the face of sentence involving equality.

3. Rules

Each *rule* ([268, 204]) has the form:

$$\langle IF \text{ precondition } THEN \text{ conclusion} \rangle$$

with the IF-part, the left-hand-side (LHS), being the antecedent comprising the preconditions which determine applicability of the rule, and the THEN-part, the right-hand-side (RHS), being the consequent consisting of the conclusion which describes the operation to be performed if the rule is applied.

A system containing an ordered set of IF-THEN rules is called a *production system*. It consists of *working memory* which stores temporary information (user-supplied facts, or intermediate conclusions generated by the system), a *rule base* which stores permanent information in the form of IF-THEN rules necessary for solving all problems in a domain, and an *interpreter*, as basic components. Control of inference is handled by the interpreter which matches facts with the antecedent of a rule, and executes the consequent of a rule.

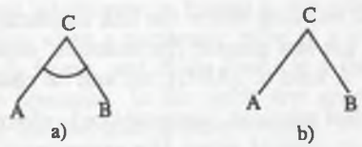


Fig. 3.2. a) AND tree, b) OR tree

Rys. 3.2. a) AND-drzewo, b) OR-drzewo

A *decision tree* is a means of structuring knowledge so that facts and rules can be dynamically combined in an inferential process. The *AND-OR tree* is a tree whose root represents the goal, and whose leaves represent the facts of an inference. The nodes of a AND-OR tree are clauses of the rules, and the arcs are arrows connecting the clauses. If all nodes are connected by the AND arcs, then we have an *AND-tree* (graphically depicted by a connecting arc - see Fig. 3.2a). Similarly, when all nodes are connected by the OR arcs, we have a *OR-tree* (graphically depicted by an unconnected arc - see Fig. 3.2b). A tree whose nodes are connected by the AND and OR arcs is a *AND-OR tree*.

To manage inferential process, effective control mechanisms are necessary. *Backward chaining* is a goal-driven process which builds a tree of a move sequences from goal to subgoal. The RHS of rules are matched against the current node (leaf or intermediate nodes), and the LHS are used to generate new nodes representing new goal states to be accomplished. The process continues until one of the goal states matches an initial state.

Forward chaining is a data-driven process which builds a tree of move sequences from initial configuration to goal. The LHS of rules are matched against the current state (root or intermediate nodes), and the RHS are used to generate new nodes until the goal is reached.

4. Semantic networks

As opposed to logic and rules, a *semantic network* ([268, 204]) is a a labeled, directed graph, that means a highly interconnected hierarchical representation of knowledge, consisting of a set of nodes connected to each other by a set of directed labeled links. The nodes represent objects which can be facts, sets, individuals, propositions, procedures, etc. The links represent diferent relationships between the objects.

Some nodes in the semantic network can be regarded as *general objects* having descriptions applicable to many individuals, while some nodes can be regarded as *specific objects* with descriptions applicable to a single individual.

For representing the *hierarchical structure* of objects and their properties, the IS-A, HAS-A, and IS-PART-OF links have been defined.

- The *IS-A link* is a taxonomic link which can be used for hierachical classification of objects. It creates a partial ordering of objects and facilitates the inheritance of properties along the hierarchy. The use of the IS-A links has been classified into the representations of two major classes of relations:
  - *Generic-generic* relations where the link connects node which is more general to those which are less general (for example, superset-subset, generalization-specialization, "A kind of"(AKO), set and its characteristic type.
  - *Generic-individual* relations (instantiation), where the link is used to connect an individual to a general object (for example set membership).
- The *HAS-A link* is employed to assign attributes and values to objects.
- The link *IS-PART-OF* indicates that an object consists of different parts.

Semantic network can also be used to represent other non-hierarchical relations: for example spatial, temporal, etc.

A variety of inferential schemes for semantic networks have been proposed. *Spreading activations* and *inheritance* are the two basic ones. *Spreading activation* is an intersection search technique ([268, 204]) which seeks relationships among objects by spreading activations out from each of two nodes and decides on where the activation meet. *Inheritance*, which is a more common mechanism of making inference in semantic networks, make use of the IS-A and HAS-A links to determine inheritance of attributes from superclasses.

5. Frames

Another hierarchical representation of knowledge is a *frame* ([268, 204]). A *frame* represents an entity in the domain world. It is a collection of *attributes* (*slots*) and their associated *values*. A slot can be viewed as a relation which maps a class (a set) to its

range (slot values). *Aspect* of an object impose ways and means by which a slot can take its values. The range is delimited by the slot aspect which may specify:

- Value of attribute.
- Classes to which the attribute can be attached.
- Value restrictions like: constraints on either the type or the value of the attribute.
- Functions or logical constraints.
- Number of values the slot may be filled with.
- Procedures for obtaining values, for example: routine to derive value whenever the slot value is modified, demons, rules for value inheritance, etc.

Two kinds of frames exist: *class frame* that represents a class of objects, and *instance frame* which represents a specific object. Class frames possess general characteristics from which all instance objects (frames) belonging to the class inherit. On the even higher level, we can have superclasses (frames) whose elements are individual classes.

The most straightforward inheritance in frames is the *value inheritance*. It relies on checking frames for slot value when a particular value of a slot is queried. If the value cannot be found in a certain frame, and if the frame is a class frame, the process will check the superclass up the hierarchy. In the *depth-first approach*, one path is searched first, and another only when the first path is not successful. *Forward chaining inheritance* can be performed by passing slots and values down the hierarchy to find instance of a general circumstance.

Another method of inference is through the process of *matching*. The slot-by-slot matching serves as a basis of this inference. When a query (a frame with specific slot values) comes in, the system will match it with existing frames in the knowledge base, and select one or more frames which have the closest match with the query. When a complete match cannot be found, a threshold can be used to decide under the what situation we can claim a match.

6. Object-oriented approach

According to the *object-oriented* paradigm ([231, 204]), the real-world is represented as a hierarchical set of *objects* linked by some *protocols of communication* among them. An *object* is an independent entity consisting of its own *attributes* which depict the state of an object, and *methods* (private procedures) that work on them, and are used for intra-object controls and communications. The only way to reach the data in an object is to use the methods to which the objects respond. In this way data are *encapsulated* inside objects with methods required to use them. Methods of an object are used through *protocols*, being sets of *messages* to which an object responds. *Messages* are used to

activate an object, i.e. to invoke a method, which is accomplished by a standard function call-return mechanism. A message consists of the following components: 1) the sender, 2) the receiver, 3) the selector(s) (method(s) which is to be invoked by the message, 4) the argument(s) (parameter(s) to be passed to the method). Message passing is handled by the message manager.

To structure knowledge in efficient way, objects are grouped into classes and organized into a hierarchy or a system of hierarchies by instantiation. A *subclass* inherits methods and attributes from its *superclass(es)* (parent class(es)). A *class* is a prototypical representation of objects in which all objects share the same set of attributes and respond to the same set of methods. An *object* is an *instance* of the class it belongs to.

Generalization, *aggregation*, and *associations* are the three most common methods of relations between objects in object-oriented paradigm. Objects related by *generalization* are structured hierarchically, with objets at the lower level (subclasses) inheriting attributes and methods from objects at higher level (superclasses). An *aggregation* relationship arrange objects in a hierarchy with lower level objects forming *parts* of a composite higher level object. Relationship by *association* arranges objects in one-to-one, one-to-many or many-to-many association.

7. Non-monotonic logic

7.1. Non-monotonic reasoning

All the formalisms presented in this section can be found in the reference literature ([213, 130, 268]).

Definition 3.25 Classical first-order predicate calculus is *monotonic* that means its provability relation, $\vdash$, enjoys the following property: For any sets of premisses S and S'

$$S \subseteq S' \text{ implies } \{A : S \vdash A\} \subseteq \{A : S' \vdash A\}$$

This means that adding new axioms will never invalidate old conclusions (theorems), or, equivalently, the set of conclusions *increases monotonically* with the set of axioms.

In a typical AI system based on first-order predicate logic, information about its domain is expressed as a finite set Δ of first-order formulas (*base set of beliefs* of the system). To take appropriate actions, the system typically will have to decide whether or not some formula ϕ is logically entailed by its beliefs. The system will make this decision by performing logical deductions on Δ, perhaps by using resolution on the clause form of $\Delta \wedge \neg\phi$.

This model has some limitations. Deductions from a base set of beliefs never produce *new* knowledge about the world. If ϕ logically follows from Δ, then all the models of Δ are also models of ϕ. Deriving ϕ in no way eliminates any of the models, and thus ϕ tells us nothing about the world that was not already described by Δ.

Many methods that violates the property of monotonicity have been proposed. They have been primarily studied in the context of human common-sense reasoning. What characterizes human reasoning, is its ability to deal with *incomplete information*. In everyday life we are constantly faced with situations where not all relevant evidence is known. In order to act, we must be able to draw conclusions even if the available evidence is insufficient to guarantee their correctness. Such conclusions may be invalidated in the light of new, more accurate information.

Definition 3.26 *Non-monotonic reasoning* ([213]) means the drawing conclusions which may be invalidated in the light of new information. A logical system is called non-monotonic iff its provability relation violates the property of monotonicity.

Non-monotonic reasoning has been given both practical and theoretical considerations in AI research. The most important formal approaches to non-monotonic reasoning are: Sandewall's formalism ([281]), modal non-monotonic logics ([225, 226]), default logic ([266]), circumscription ([223, 224]), reasoning under closed-world assumption ([267]), model-theoretic approach ([301]).

7.2. Poole's approach

Poole's ([258]) approach is a very simple non-monotonic formalism, in which knowledge is represented as a pair $\langle F, \Delta \rangle$, where F is a consistent set of first-order sentences, representing *facts* that are known to be true, and Δ is a set of *hypotheses* which represent *non-monotonic rules*. Formally, a hypothesis is any expression of the form:

$$(x_1, ..., x_n)\ ASSUME\ A(x_1, ..., x_n)$$

where A is a first-order formula whose free variables are precisely $x_1, ..., x_n$. We are to read this as "for all individuals $x_1, ..., x_n$, represented by ground terms $\alpha_1, ..., \alpha_n$, if $A(\alpha_1, ..., \alpha_n)$ can be consistently assumed, then assume it". The sentence $A(\alpha_1, ..., \alpha_n)$ is called an *instance* of the hypothesis.

Given a pair $\langle F, \Delta \rangle$, the ordinary first-order theories of the form $F \cup D$, called *theories* for $\langle F, \Delta \rangle$ are constructed, where D is any set of instances of members of Δ such that $F \cup D$ is *consistent*.

Definition 3.27 A sentence A (known as *observation*) is said to be *explainable* in $\langle F, \Delta \rangle$ iff there is a theory $F \cup D$ for $\langle F, \Delta \rangle$ (called the *theory that explains* A) such that $F \cup D \vdash A$.

A set S of sentences is explainable in $\langle F, \Delta \rangle$ iff there is a theory for $\langle F, \Delta \rangle$ that explains all members of S. In Poole's logic, any set of sentences explainable in $\langle F, \Delta \rangle$ may be viewed as the set of beliefs about the world represented by $\langle F, \Delta \rangle$.

Chapter 4

Geographic map image analysis

Despite the research efforts in geographic map image analysis, most of the research in this area still concentrates on the design of specific algorithms rather than on image understanding. A common approach to convert a rasterized map drawing into an object description is to develop *application specific algorithms* to extract the objects from the image, and apply these algorithms in a static, *bottom-up sequence* ([42, 71, 161, 327, 94, 17, 18]. An often encountered first step in such a sequence is separating the text symbols from the graphics using features like the size and collinearity of the text, for further recognition by OCR module ([181, 244]). Following this step, the text can be classified by OCR module, and the remaining graphics are approximated by using vectorization. Graphical objects can then be extracted by grouping appropriate vectors ([25, 134, 161, 181]). Some knowledge-based approaches to group vectors into objects using a rule-based system have been described in the literature ([327, 134]). Only 4 approaches have been described in the literature ([222, 306, 252, 151] in which domain knowledge is explicitly used, thus allowing for some understanding processes.

In this section the algorithms and approaches which were used in the existing geographic map image analysis systems will be outlined. The algorithms are presented in a sequence they are performed in a traditional, bottom-up approach to image analysis ([133]): first, the algorithms used in *low level* pixel processing, then, those used in feature level analysis (*intermediate level*), and finally, algorithms applied in high level vision tasks, that means *image understanding*.

1. Low level

1.1. Binarization

Built-in binarization methods of most scanners are sufficient for clean maps. But, when dealing with large map drawings which have been folded and stored away for a long period, we may have to use some *adaptive binarization* methods ([133, 320]). Such a

method is computationally costlier than the built-in tools, but it is necessary to avoid false objects due to folds, or to the merging of lines close to each other on a poor-quality map. Bear in mind the special nature of maps, the *contour-based approach*, that means finding contours, and filling the contours of black objects, yields the best results. Among algorithm in this group, the improved *integrated function algorithm* [344] which is a locally adaptive method, is especially interesting. This method applies a gradient-like activity operator on the image. Pixels with activity below a manually set threshold are labeld '0'. The remaining pixels are labeled '+' if the Laplacian edge operator of the pixel is positive, otherwise '-'. All '+'-marked regions are labeled object, and '-'-marked regions are labeled background. A '0'-marked region is labeled object, if '+'-marked pixels that are 8-connected to it are in majority, otherwise it is labeled background. The special post-processing step ([374]), based on the average gradient, is used to improve the binary image by removing "ghost"objects.

1.2. Text/Graphics segmentation

Many algorithms have been developed for text/graphics segmentation, which are either raster ([359, 38, 248, 112, 370]), vector-based ([199, 212]), or mixed ([5]), but only few methods are able to extract and recognize characters and symbols *overlapping* graphics with satisfied rate.

Recently, an efficient algorithm for separation of characters from map/graphics based on *directional morphological operations* ([371]) has been described ([214, 215, 216]). In the first step, directional edge line image is generated in which all of the edge-lines are decomposed into groups corresponding to eight directions. As the edge segments of character edges are shorter than those of line edges, character edges are deleted using recursively *directional open* operation. The number of iterations is determined based on length histogram from which a length threshold separating character from line edges is computed. Character image is generated by subtracting the line image from the original image. Missing, or distorted characters (due to connection between them and lines), are restored by *directional close operation* performed in four directions on character image. Each character string forms a connected component in at least one of such obtained four images. A minimal enclosing rectangle is computed for each connected component in those images. Then, the regions from the original image that correspond to those rectangles in four images are copied to character image, and the characters are restored.

Besides this common approach, there is an opposite one (for example [72]), in which characters are explicitly recognized in maps.

1.3. Thinning

With the text and solid regions extracted, the graphical elements left are composed purely by lines. These lines are usually reduced to graphs, which may be accomplished by obtaining a skeletons of these lines via a *thinning* algorithm. The *skeleton* of a region may be defined via the *medial axis transformation* (MAT) proposed in [39].

Definition 4.1 . The MAT of a region R with border B is defined as follows. For each point p in R, we find its closest (in a given distance metric) neighbor in B. If p has more than one such neighbor, it is said to belong to the *medial axis* (*skeleton*) of R.

As direct implementation of the above definition is typically computationally prohibitive, numerous algorithms have been proposed for improving computational efficiency while at the same time attempting to produce a medial axis representation of a region. These algorithms for thinning a digital pattern can be divided in two main groups:

- **Pixelwise** methods, further divided into:
  - *Iterative* thinning methods which iteratively execute many passes over the pattern, deleting edge pixels. These are divided into:
    - *Sequential* methods in which the deletion of pixel in the n-th iteration depends on all the operations that have been performed so far. These algorithms can be based on contour tracing (for example [21, 363]), or raster scanning (for example [153, 243]).
    - *Parallel* methods in which the deletion of pixel in the n-th iteration would depend only on the result that remains after the (n-1)th iteration, therefore all pixels can be examined independently in a parallel manner in each iteration. Among existing parallel algorithms, many use several passes in each iteration, each of which removes boundary pixels from a single given direction, so they are partially serialized (e.g see [324, 142, 381, 170]).
  - *Distance-transform based* thinning methods in which skeletal pixels are identified on a *distance transform* of the image (for example, [23, 22, 89]).
- **Non-pixelwise** methods. Produce a medial axis of a pattern directly, in one pass, without examining all the individual pixels ([254, 304, 328].

The major disadvantages of thinning are high time complexities (cubic for iterative thinning), loss of shape information (e.g. line width for iterative thining), distortion at junctions and false and spurious branches (at line ends and corners).

One of the best thinning algorithms which is used in many geographic map conversion systems is the *(3,4)-distance transform driven thinning algorithm* ([29]).

Definition 4.2 Let P and $\overline{P}$ determines the sets of the black and white pixels in a binary picture. The (w_{odd}, w_{even})-*distance transform* of a pattern P with respect to $\overline{P}$ (referred to as (w_{odd}, w_{even})-DT) is a multivalued replica of P, where each pixel p is labeled with its (w_{odd}, w_{even})-distance from $\overline{P}$. The two weights (w_{odd}, w_{even}) are used to weight the horizontal/vertical and diagonal unit moves in length computation, respectively.

Many algorithms for (w_{odd}, w_{even})-DT computation are available (for example [43]). The algorithm [29] is composed of the following steps:

- Computation (3,4)-distance transform of the pattern P.

- Extraction of a skeletal pixels from the $(3,4)$-DT of the pattern P which is composed of:
  - parallel extraction of pixels directly identified on distance transform,
  - sequential extraction of pixels (identified by growing increasing path along the direction of the steepest gradient in distance transform, necessary to link the components of parallelwise detectable skeletal pixels).
- Prunning, beautifying.

1.4. Vectorization

Vectorization, being a raster-to-vector conversion process, is one of the main components of the map analysis system. A typical vectorization process includes:

- Finding the lines (medial axis point sampling).
- Line tracking, which captures the medial axis points found into chains of points making up the lines.
- Feature detection on lines by means of *polygonal approximation* or *dominant point detection* which remove non-critical points from the point chains.
- *Post-processing*, to find better positions for the junction points, to merge some vectors and remove some others.

The first step in vectorization is to process a raster image, supposed to contain graphics in order to *extract a set of lines*, i.e. of chains of pixels. The most intuitive definition for these lines is that they represent the *set of significant medial axes* of the original image, considered as a shape. This set must be *linked*, to extract the chains making up the lines. The lines extracted from the previous step are to be represented by a set of segments. This is accomplished by *polygonal approximation* (and higher order curves fitting) based algorithms (see for example [305, 262, 228, 194, 203, 360]), or *dominant points detection* based algorithms (see for example [275, 263, 341, 15, 114, 159, 126]).

The objective of *polygonal approximation* is to approximate a given curve with connected straight lines such that the result is close to the original curve but the description is more succinct. The popular, *iterative* endpoint-fit algorithm ([262]), whose error measure is the distance between the original curve and its polygonal approximation, begins by connecting a straight line segment between end points of the data. If any perpendicular distance from the segment to each point on the curve is greater than a chosen threshold, the segment is replaced by two segments from the original segment end points to the curve point where the distance to the segment is the greatest. The processing repeats in this way, and the iterations are stopped when all segments are within the error threshold of the curve. Instead this overall approach, an approach can be used in which segments can be fit starting at an endpoint, then fit sequentially for points along the line

([305]). In another class of polygonalization techniques, area error (between the original line and the approximation) is used as a measure of goodness of fit ([360]). If a closer polygonal approximation is required, one method is to use a minimax measurement of error ([194]). One potential drawback of polygonal approximation is that the result is not usually unique. One approach to reducing this problem is to find a starting point along the curve that will always yield similar results ([203]). Beside polygonal approximation methods, there are higher-order curve- and spline-fitting methods ([228]) that achieve closer fits, but are computationaly more expensive.

As opposed to a polygonal approximation, the objective of the *dominant point detection* is to precisely locate the breakpoints, or so-called *dominant* or *critical points* (that means vertex of an angle, or a corner which is an intersection of a line segment and an arc, or an intersection of two arcs). One approach is based on a curvature estimation ([275, 15]). The concept of critical points derives from the observation that humans recognize shape in large part by curvature maxima in the shape outline ([27]). To locate corners, the curvature plot is thresholded to find all the curvature points above a chosen threshold value, which correspond to corners. The user must choose one method parameter: the length of the line segments to be fit along the data to determine the curvature. The length should be chosen as the minimum arc length between critical points. This length is called the *region of support* because it relates to the arc length on the curve that affects the curvature measurement at any single point. The drawback of these methods that the user must have an idea of the noise and feature characteristics to set the length of the line fit. Adaptive methods have been proposed which does not require any threshold ([263]).

The vectorization very often leads to the results which are contrary to what the draftsman expects. This is because it is 'blind', that means it does not assume about what a given curve is supposed to represent. Several authors propose a set of simple heuristics to correct the result, setting junction straight, merging those which are close to each other, etc. (see for example [61]). An interesting approach is to add geometric constraints describing the ideal geometry of the result, to the vectorization process itself ([174, 271, 272]) or after, as post-vectorization (for example [157]). The *post-processing* method for building vectorization proposed in [157] is based on a *generic building model*, which states that a relation between the sides of a building polygon is parallel, or perpendicular. If the buiding has a desirable shape, the histogram of vector orientation vice the frequence (both weighted by the vector length) should have two sharp peaks whose distance along the vertical axis is 90 degrees. In the post-vectorization stage, the vector sequence obtained from the pre-vectorization is modified (by shifting knot points) iteratively by the relaxation method so that the cost function (the variance for the histogram) derived from the generic model is minimized.

Many vectorization methods actually perform the above 4 steps of vectorization in a different order, or merge several of them into a single step, for example finding the lines (the first two steps), and approximating them simultaneously.

According to the method of finding the lines, the vectorization algorithms are divided into two classes:

- *Thinning based* methods.
- *Non-thinning based* methods:
  - Hough transform based.
  - Contour-based.
  - Run graph-based.
  - Mesh-pattern-based.
  - Sparse pixel based.

1.4.1. Thinning-based methods

In this group of methods, lines are found by *thinning* operation. One of the best methods in this group is that proposed in [279]. It is a recursive algorithm in which a list of edge pixels is segmented recursively into the two lists at the point of a maximum deviation, until a line segment is smaller than four pixels long, or the deviation is less than three pixels. The measure of goodness of fit, the *significance*, is defined as the ratio of maximum deviation divide by the line segment length. As the recursion unwinds, the resulted multilevel tree is traversed back to the root. At each level, if any of the line segments that passed up from the lower level are more significant that the line segment at the current level, they are retained and passed up to the next higher level as candidate line segments. Otherwise, the line segment at the current level is passed up. As opposed to other existing algorithms, in which a threshold have to be manually set, this method does not require any threshold. Instead, the most appropriate values are chosen dynamically throughout the procedure.

Sequences of line segments obtained from the polygonal approximation are then hypothesize as circle segments, and refined by subdividing at the point of maximum deviation. Finding the optimum arc is accomplished by minimizing the error for three parameters: the radius, and coordinates of the centre of the circle during a search along the line that is normal to the line joining the two arc endpoints. The measure of match, the total error between the data and the arc is determined using a least square error criterion. The error for each vertex is the shortest distance to the circle. A recursive binary splitting algorithm rapidly searches for the minimum error using the gradient descent method.

1.4.2. Non-thinning based methods

Hough transform based methods ([85, 144]) are used in case of isolated points, wheras in maps we have digital lines, so they have not been applied in line drawing analysis.

The main idea of *contour* based methods ([177]) is to find the edges of the line object, and then calculate the middle points of the pair of points on two opposite parallel edges. These algorithms sample and track the medial axis points simultaneously. The problem of all algorithms of this type is how to deal with junctions. The first one is the merging junction formed by intersection at a small angle, which is most likely to be missed during

tracking. The second problem is the cross intersection where how to join up the lines is problematic. This group of vectorization algorithms have quadric time complexity and are much faster than thinning based methods.

Run graph based methods ([42, 236]) are similar to contour-based methods, but the construction of intermediate representation is different. Disadvantages include inaccurate intersection points due to coarse location of junction areas, and false junction areas.

The basic idea of *mesh pattern* based methods ([206, 354, 355]) is to divide entire image using *meshes*, so as each side of a unit mesh border intersects mostly one line segment of the image. Unit mesh is analyzed according to the pixels on the 1 pixel wide border of a mesh. It is then labeled according to its characteristic pattern. An image is represented by a *control map* in which each unit mesh is replaced with its characteristic pattern label. The lines are recovered and tracked from mesh to mesh in the control map by analyzing the characteristic patterns of the meshes. Complicated meshes with unknown label are further split into several smaller but known mesh patterns whose shape may not be square. A method is fast due to sparse pixel access. However, the mesh size is hard to control. The method is not suitable for the detection of more complex line patterns (for example dashed line).

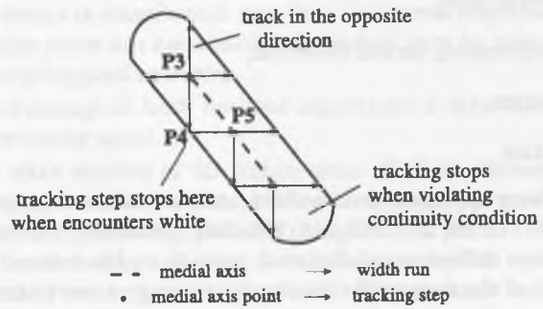


Fig. 4.1. Illustration of the main procedure of SPV

Rys. 4.1. Ilustracja głównej procedury algorytmu SPV

In this group of methods ([81]), only the *selected subset* of medial axis points is visited, so they are time efficient due to sparse sampling. One of the best algorithms from this group is the *Sparse Pixel Vectorization* (*SPV*) ([81]) algorithm. To start SPV, the reliable *starting medial axis point* and a *tracking direction* are found first. The starting point is approximately a middle point of both horizontal and vertical runs passing it, while tracking direction is determined by the longest run at starting point. Going from the starting point $P3$ in the tracking direction which is vertical, (Fig. 4.1) point $P4$ is reached. From point $P4$ the two opposite directed width runs are made, from which the width run is obtained, and its middle run point $P5$ that serves as the new medial axis point. This is the end of the tracking this cycle. The tracking cycle is repeated as long as some *continuation conditions* are satisfied (width preservation, length direction consistency). At junction some of these criteria are violated, and a *junction recovery* procedure is invoked. It is a 3-step procedure consisting of: 1) retreating the last medial axis point, 2)

adjusting the tracking step length by half-sizing the length of the current tracking step, 3) testing the conditions at the new position. After polygonal approximation, optional *refinement processes* are performed like linking collinear line segments into one, correcting defects at junctions (the real corner is recovered by computing the intersection), refining the vector endpoints, converting polylines with low enough curvature to straight line segments.

2. Intermediate level

In this level, the two processing, *text recognition* and *graphics recognition* are performed. Examples of graphics recognition in maps are:

- symbol recognition,
- line texture object detection (e.g. isolines, dash-, dash-dotted lines, etc.),
- 2D texture recognition,
- hatched area detection, arrows detection,
- polygon detection,
- arc segmentation.

Several papers were appeared that describe various statistical *geographic symbol* classifiers ([264, 372, 237, 16, 98, 261, 242, 219, 380, 280]). However, the statistical classifiers, which learn the shape differences of distorted symbols in hand-drawn maps are usually impractical because of the danger of either undertraining or overtraining.

A hybrid (structural/statistical) approach for scale- and rotation-invariant recognition of multi-component geographic symbols have been proposed in [35]. Each multi-component symbol is represented by a hierarchical connected-component representation (a component tree). A set of shape features like *circularity, number of holes, sides, lids, side-torsion, lid-torsion* (see [35]), computed on convex-hull of a shape are used to identify each individual component in the component tree. A *decision-tree classifier* is used to identify the shapes of the individual components of a symbol. Structural matching, based on exact tree matching of a candidate component tree with a series of prototype trees is then used to determine the class of a symbol. Again, a classifier relies on learning, as well as a number of prototypes per class to capture a variability of shapes.

The problem of *dashed line* detection is a common problem in many computerized document analysis applications, such as the analysis of engineering drawings and maps. Several algorithms have been proposed to solve this problem (mainly for engineering drawings), which are either vector- ([86, 81, 244]) or raster-based ([8]).

In [86, 81] a *vector-based* dashed-line detection algorithm which is a specialization of the unified *Generic Object Recognition Methodology* (*GORM*) developed for engineering

drawings ([83]) have been proposed. The *dashed line detecion algorithm*, as each GORM algorithm, consists of two main phases: 1) *hypothesis generation*, in which the *first-key component*, being a *dash* is detected, and 2) *hypothesis test*, in which the detected dash is extended in two extension directions from its two endpoints to the maximum possible extent in order to find all the other components which is accomplished by an *extension process*. The candidates are found gradually, in the square stretching from the current extending point outward, along the direction tangent to a dash at this point, and sorted by their nearest distance to the first dash. If a candidate pass the *candidacy test*, as well as the *extendibility test*, it is accepted, otherwise it is rejected. *Candidacy test* consists of thickness, collinearity and proximity tests. The *extendibility test* consists of shape test which requires the satisfaction of: gap, length and pattern similarity, and style test which requires that distances of both endpoints of the candidate dash from the line's medial axis be less than half the current line width. The final *credebility test* prevents the inclusion of false positives due to accumulative error.

In [8] a *raster-based* algorithm for *dashed line detection* have been proposed. This algorithm uses *adaptive directional morphological operators* with morphological kernels being non-isotropic. The adaptation is based on the local characteristics of the data. First, the dashes image is transformed into four directional edge planes, and dashes in each directional edge plane are associated with dashed lines by using the newly defined *tube-directional morphological operators*.

The main disadvantage of both outlined algorithms is that they cannot cope with situation when overlaping occur.

Map-drawings often consists of *2D texture* areas which are formed by a repetition of some texture elements, for example grass, bush or tree geographic symbols. A general task of such texture objects processing, includes recognition of texture type, and extraction of texture region border. One technique for the texture border extraction is described in [309], and is based on a distance transform. The distance map contains a set of "waves"and some of them could be directly chosen as a texture border.

Accurate *arc segmentation*, that means recognizing arcs from scanned line drawings, essential for high-level engineering drawing interpretation is very difficult due to noise, clutter, tangency, and intersection with other geometry objects. *Hough Transform* was the first technique to be employed in arc segmentation ([144]), but its requirement in time and space is very high. Another group of arc segmentation methods (e.g. [26]) is based on *curvature estimation*, whose aim is to extract meaningful features from objects by estimating their edge curvature, but this requires heavy pixel level preprocessing. An interesting algorithm for segmenting curves into a combination od arcs and straight lines has been proposed in [279] and described in this chapter. The *Perpendicular Bisector Tracing* (PBT) ([84]), being a vector-based arc segmentation method developed for engineering drawings, which takes an input the vector pieces of the image generated by the vectorization procedure ([364]), utilizes the fact that three different points determine an unique circle, and the circle center is the intersection of three perpendicular bisectors of the triangle formed by the three points. An improved version of PBT is *Incremental Arc Segmentation algorithm* (IAS).

Examples of recognition of *hatching* areas can be found in [18, 208].

3. High level

At this level, the extracted graphics objects together with the textual information from intermediate level, are assembled into entities having application dependent meaning, for example parcels, roads, buildings, etc.

Most of the existing map analysis systems ([42, 71, 161, 327, 94, 17, 18]) are based on traditional bottom-up approach, and a priori knowledge is *implicit* and *hidden* in the source code, which makes impossible its utilization to control the analysis process, and for image understanding. Only in 4 map image analysis systems ([222, 306, 252, 151]), the explicitly represented domain knowledge allows to perform some understanding.

In a system for Bavarian cadastral map interpretation [222], the knowledge is represented by a semantic network which utilizes only hierarchies. Top-down processing is started with a goal defined by the user, all necessary concepts to achieve this goal are computed and put into an order based on a semantic network, and the bottomp-up instantiation process is started to instantiate a goal. The main disadvantage is the fact that no verification with a model, or previously obtained results is proposed, which implies rather low reliability of the obtained results.

In [306] a semi-automatic system for Dutch utility map conversion is described in which a priori knowledge is represented by a semantic network. This mechanism starts from initial segmentation of all leaf objects in a semantic network in a strictly bottom-up manner, and then the interpretation is accomplished by iteration of the a cycle composed of hypothesis generation, selection and verification.

In a system for French cadastral map interpretation [252] a strictly bottom-up processing is followed by a feedback cycles between high - and low-level processings to reconsile several possible inconsistencies.

The main disadvantage of the above two approaches is that thy rely on a conflict resolution strategy which requires specification of all possible conflicting situations that may appear. Though in [252], this strategy offers a memorization possibility, but very complicated administration of remedies with user intervention is required. Moreover, in this approach none of the knowledge representation formalisms is used.

The most interesting contribution is the approach presented by J. den Hartog [151] for Dutch utility maps conversion. The system decomposes the binary utility map image into primitives (not vectors !) represented as nodes of a semantic network. The *contextual reasoning* is performed based on a loop that includes:

- search action generation in a small region of interest,
- primitive recognition,
- inconsistency detection.

The mechanism to generate new search actions is based on the perception that each object type is related to other object types. The interpretation flow is guided by a semantic network through priorities assigned to links which determine the execution order of bottom-up primitive recognition by template matching. Detection of inconsistencies is based on three types of spatial relationships: optional, essential (if the first object is found, the related object must also be found), and negative (opposite of essential).

The serious drawback of [151] is that its model only describes binary spatial relationships between simple objects (primitives, not vectors), and its is therefore not possible to obtain an understanding of the structure of the map at a high level. In the case of more complex maps, this kind of system is penalized because of the drastically increasing number of relations. Moreover, the knowledge representation does not integrate hierarchical relations.

A common disadvantage of all the above mentioned systems and methods is the lack of reliability and accuracy.

Chapter 5

The proposed framework

Bearing in mind the two main aims of the presented thesis, and to cope with the mentioned problems in chapter one, we have developed the *general reasoning framework* for *knowledge-based* geographic map image analysis. The primary objective of the developed framework, is to construct a symbolic description of the map depicted in an image to be digitally stored in a database of Geographic Information System.

A *geographic map* is a spatial representation of geographic objects which are symbolized using the legend of the map and some drawing rules. Some objects (e.g. roads, pavements) are not symbolized in the legend, but can be recognized through their functionality ([124, 320]). Maps are predominantly composed of graphics components: different lines, symbols and regions, textual elements are used only to augment the information given by the graphics ones.

The conversion of the paper large-scale maps, that means maps with a scale ranging from 1:500 to 1:5000 in general, and of the situational layer of a Polish Fundamental Land Map (PFLM) [124] in particular, have been chosen as the application to test the new techniques and methodologies developed within the context of our research.

Geographic objects of a PFLM are classified into 13 groups (e.g see Fig. 5.1 for some examples) and maintained in 4 layers:

- (E) land register,
- (S) situational objects,
- (U) networks (electricity, telecom, gas, water, etc.),
- (W) isolines and geodesic base (different altitude points).

The examples of map fragments from the situational layer of a PFLM are shown in Appendix A. PFLM provides a complete, topographically correct map which can be used as a basis for making another maps.

GIS represents geographic objects by means of so-called *spatial data models* of which the *vector* model is the most important. *Vector* representation is typically based on two types of data: 1) *geometric data* (represented by points, lines, or areas), and 2) *textual*

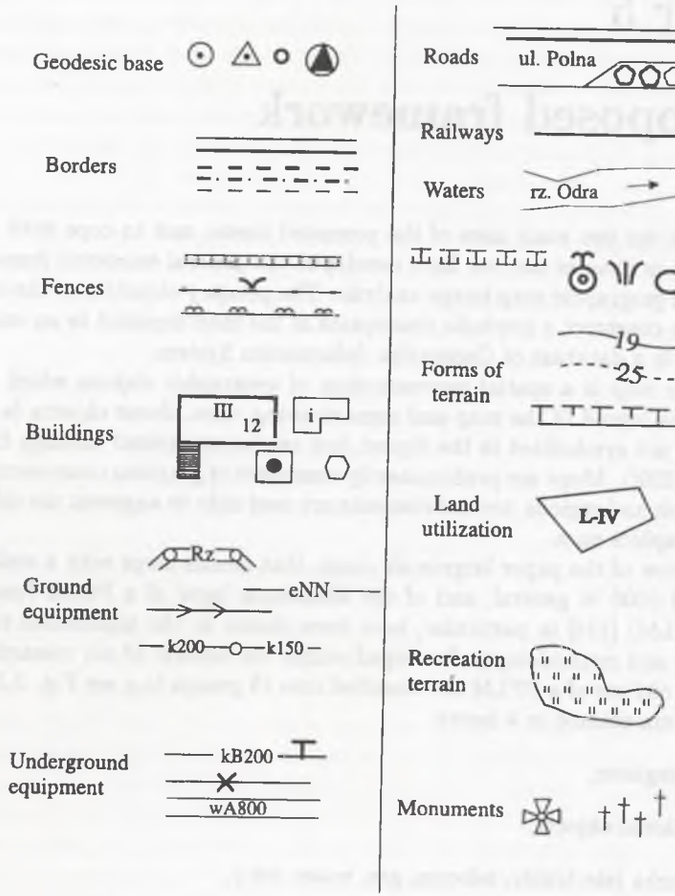


Fig. 5.1. Examples of geographic objects from a PFLM

Rys. 5.1. Przykłady obiektów geograficznych z Polskiej Zasadniczej Mapy Kraju

data. Typical GIS software, such as Environmental Research Institute's *ARC/INFO* ([99]), is composed of *software tools* which contain components to input, code, analyze, and display geographic objects, and a *relational database* which stores geometric and textual descriptions of these objects (see Fig. 5.2). Among the most important modules of software tools are indexing methods, enabling to perform spatial analysis in an efficient way (see for example ([312, 313])).

The proposed knowledge-based approach to map image analysis has been applied in the semi-automatic map conversion module *MAPIN* which replaced the manual input into the *Integrated Spatial Organization System SIT* ([260]). SIT is built on a top of a popular GIS software 'ArcInfo/ArcView' ([99]) (see Fig. 5.2).

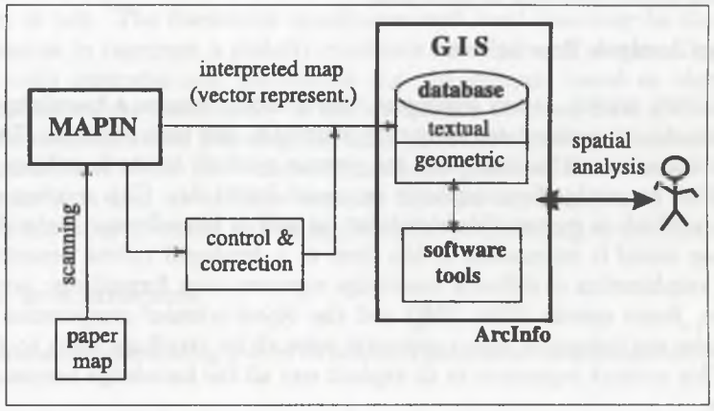


Fig. 5.2. The scheme of the proposed technology for semi-automatic map conversion with the developed framework MAPIN

Rys. 5.2. Schemat zaproponowanej technologii półautomatycznej konwersji map wraz ze strukturą wnioskującą *MAPIN*

1. General characteristics of the framework

1.1. The components of the framework

In this thesis we prove that to obtain the optimal map analysis results, the *knowledge-based approach* with mixed control strategy yields the best solution. Thus, the developed framework (published in [321]) is characterized by the following features:

- All a priori domain knowledge is explicitly described using some general-purpose knowledge representation formalism, and is stored as the model (*representation based on a model*).

- The analysis process uses the model to correct, improve the analysis results, and to overcome the insufficiency of input information (*analysis method based on a model*).
- The control strategy is mixed.

The proposed framework for knowledge-based geographic map image analysis is composed of three schemas called:

- **Map model** (a declarative-procedural representation of domain knowledge).
- **Object detection methods (detectors)** (a mostly procedural knowledge for specific object detection in the image).
- **Image analysis flow scheme**.

Map analysis system, as any intelligent system should possess a knowledge base containing facts and concepts related to a problem domain, and their relations. There should also be an inference mechanism which can process symbols in the knowledge base, and derive implicit knowledge from explicitly expressed knowledge. This requires a choice of formalism in which to express this knowledge, as well as the inference method.

The *map model* is represented in the form of a developed *hybrid semantic network* which is a combination of different knowledge representation formalisms: *semantic network*, *rules*, *frame* system ([204, 268]) and the *object-oriented* computation paradigm ([231]). Rules are integrated into a semantic network by attaching them to the slots of a frame. This network represents in an explicit way all the knowledge necessary for map description.

Each *object detection method*, specifies features of an object, and provides recognition algorithms for correspondence detection between image and object features. These detectors are *integrated* into a whole analysis process by the developed, suitable analysis strategy described by *image analysis flow scheme*. The *image analysis flow scheme* is composed of two structures:

1. Control mechanism.
2. Reasoning method.

The *control mechanism* is composed of the *initial recognition* step followed by the iteration of the *interpretation cycle* consisting of:

1. Hypotheses generation.
2. Compatibility examination.
3. Scenario selection.
4. Scenario verification.

In the proposed framework, the bottom-up and top-down control strategies are integrated into a unified, *mixed* control mechanism in which the low-level process is controlled by a context information from high level model.

The proposed *reasoning method*, called the *complementary reasoning*, is a non-monotonic reasoning method exploring different types of relations between geographic objects in a domain. The logical framework of the proposed complementary reasoning is based on a modified Poole's approach to non-monotonic reasoning ([258]).

When map object is recognized, the corresponding object class is instantiated to produce an object instance, representing the recognized object. Each object instance uses its own knowledge (the set of rules stored in an object class) to generate hypotheses about its related objects, and to perform reasoning depending whether the hypotheses are verified or not. The framework coordinates such local reasoning by the reasoning object instances to construct a globally consistent description of a map. The reasoning process actively generates new information (i.e. hypotheses) based on object models which complements the insufficiency of the information obtained from the image. As a result, multiple descriptions denoting the same object are generated and accumulated. Such accumulation of object instances and hypotheses generated from a model by different object instances, is used to establish relations between object instances, or to activate appropriate detectors to extract new image features corresponding to the missing objects.

1.2. The level structure

We have proposed the following *4-level structure* of geographic map image analysis process (Fig. 5.3):

- The *Pre-processing* level which is performed on the input, scanned map, includes:
  - *Binarization*
  - *Filled region extraction*
  - *Thinning*.
- *Vector Graph Construction* level is composed of the following processings:
  - *Line structure tracking*,
  - *Vector Graph Segmentation*
  - *Small Graph vectorization*.
  - *Big Graph vectorization*.
- *Graphics and Text Recognition* level. These two kind of processing are performed by specialized structural detectors, each of which is based on appropriate model of a shape. Text recognition is performed on a Small Graph, while Graphics Recognition, except for symbols, on a Big Graph. Graphics recognition comprises geographic symbol recognition, line texture extraction, 2D-texture recognition, polygon detection and classification.

- In *Understanding* level, a meaning is given to the image, that means an image is analyzed to *interpret* the map in terms of the domain geographic object models given to the framework as knowledge about the domain. The extracted graphics objects, together with textual information are assembled into entities having application dependent meaning, for example parcels, roads, buildings, etc. In this process all a prori *knowledge* about a problem domain is used. The *interpretation* refers to the *mapping* between the *description of the map*, (i.e. *interpretation network*), and the structure of an image (Fig.5.4). It associates objects in the map (house, road) with image features in the image (lines, polygons, points, etc.). Having such interpretation network computer system can answer various queries about a map (for example, "how many trees grow in a given pavement"). Thus, in this sense, it understands the map.

The process of map image analysis should be followed by the *reconstruction*. The interpreted map suffers from scale, translation and rotation distortions. The goal of the *reconstruction* is to correctly position the map elements of the interpreted map in the coordinate system of the base map, which is geometrically correct.

The proposed levels correspond to the traditional three levels of map image analysis: low level, intermediate level and high level ([133]). The difference is a more detailed representation of the intermediate level, that means in the form of two levels: *Vector Graph* and *Graphics and Text* levels, which is useful for line drawings like map. This separation in two levels is done because in maps, classified and symbolized information (*Graphics and Text*) can be separated from simple topologic vector data with no graphical meaning (*Vector Graph*).

1.3. The modular structure

The following modules: Reasoning Engine (*RE*), Detectors Manager (*DM*) Detectors (*De*), Vector Graph Construction module (*VGC*), Resegmentation module (*ReSe*) and Evidence Pool (*EP*) compose the modular structure of the proposed framework. These modules and their connections are depicted in Fig. 5.5.

The modules: *RE*, *DM* and each of the *Detectors*, are the reasoning modules to construct the description of the map. These modules are loosely coupled and communicate with the others via simple question and answer protocol.

Evidence Pool (EP) is a database where accumulation of object instances and hypotheses (called **evidence**, or database entities (**DE**)) is performed during analysis. Each piece of evidence is represented in two forms: 1) the *geometric form* representing its spatial location, 2) the *symbolic form* representing its attributes and relations to others.

Reasoning Engine (RE) is a global coordinator of reasoning performed by independent object instances. It examines the compatibility among all pieces of evidence in the database to form *scenarios* which represents a local environment context, and selects the most reliable scenario. Then, it activates either the *data-driven* or *model-driven* reasoning, depending on the nature of the selected local environment.

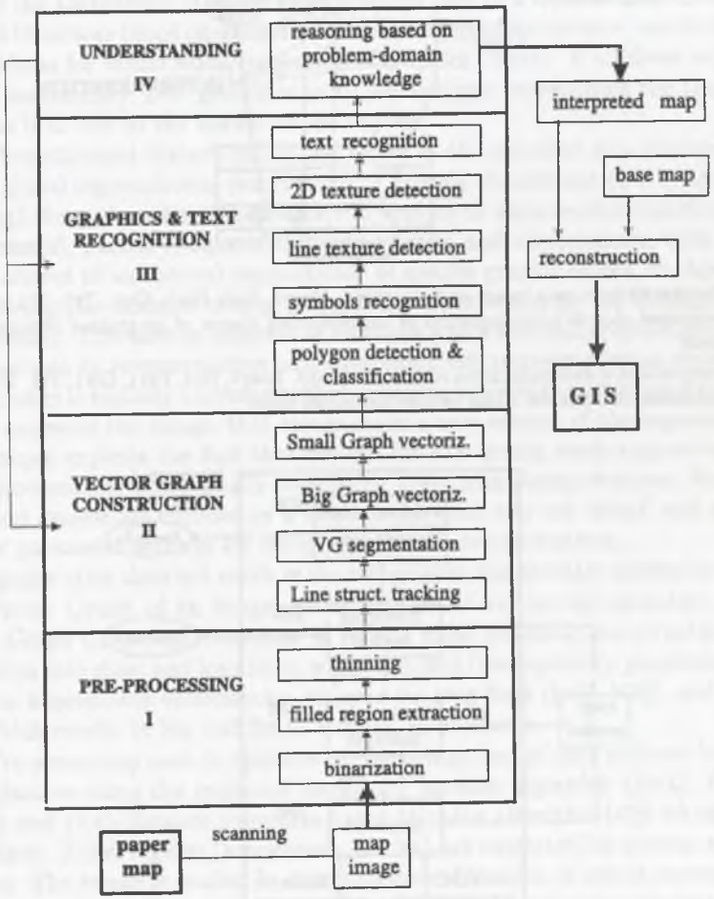


Fig. 5.3. The proposed four levels of geographic map image analysis

Rys. 5.3. Zaproponowana 4-poziomowa struktura półautomatycznej analizy map geograficznych

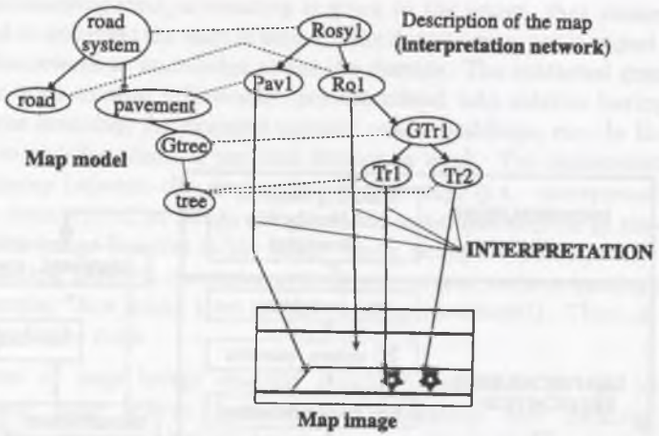


Fig. 5.4. Interpretation in map image understanding. Rosy1, Ro1, Pav1, Gtr1, Tr1, Tr2 are labels of recognized objects being instances of corresponding classes of geographic objects from map model

Rys. 5.4. Interpretacja w automatycznym rozumieniu mapy. Rosy1, Ro1, Pav1, Gtr1, Tr1, Tr2 - etykiety rozpoznanych obiektów z klas tworzących model mapy

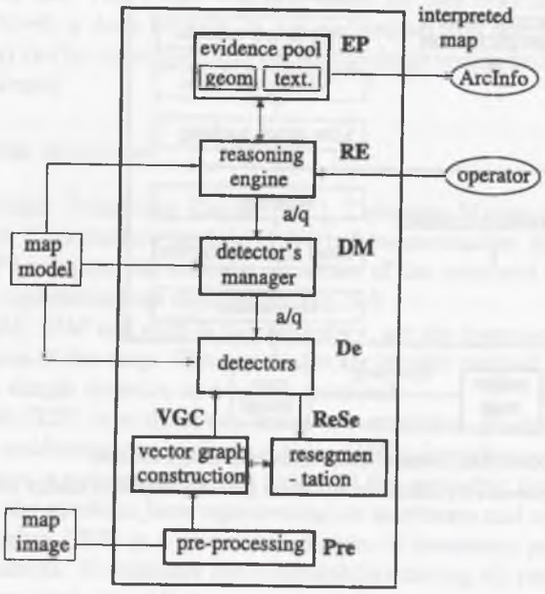


Fig. 5.5. The modular structure of the proposed framework

Rys. 5.5. Moduły zaproponowanej struktury wnioskującej

Detectors Manager (DM) is activated by *RE* to search for the most promising appearance of the object to be detected in a model-driven reasoning. After determining the appropriate object model, *DM* activates an appropriate Detector to extract the graphics object (as the instance of the selected appearance model) from a Vector Graph of an image.

Each of the **Detectors (De)** performs analysis only in a local area specified by *DM*. Specialized Detectors based on different pattern recognition techniques, use intrinsic properties of objects for *model-based recognition* of graphics objects. If all these processes are completed successfully, *DM* generates an object instance representing the target object, and returns it to *RE* as the answer to the request.

If the hypothesized feature cannot be found in the specified area (because of poor quality of global segmentation, deformation of graphic objects and text/symbol/graphic overlapping), first, the *alternative detector* is applied to improve the classification. If it is not successful, partial re-runs of the segmentation and vectorization, with tuned parameters tailored to an optimal segmentation of specific graphic object, or object-specific post-processing (for example closing polygons), are performed by the **Resegmentation** module (**ReSe**). This usually happens at locations where text and graphics itersect. Such *refining matches by resegmentation* technique effectively recovers missing object features when the image is initially undersegmented. Domain knowledge and image data are combined to *resegment* the image, that means make a new version of the segmented image. This technique exploits the fact that objects features giving weak supportive evidence may be discovered by semantically associating them with strong features. ReSe module uses a priori knowledge encoded in a map model, thus it is not 'blind' and can control the proper parameter settings for resegmentation, or revectorization.

If resegmentation does not result in the recognition, the operator assistance is invoked.

The Vector Graph of an image, being its efficient vector representation, is created in *Vector Graph Construction* module in several steps including line structure tracking, segmentation into short and long lines, which is in fact (text,symbol)/graphics separation, followed by appropriate vectorization schemes for long lines ([167, 168]), and short ones ([315]), which results in Big and Small Graphs, respectively.

The *Pre-processing* module operates on input map image, and includes locally adaptive binarization using the improved integrated function algorithm ([344]), filled region extraction and (3,4)-distance transform based thinning algorithm [43]) which is used to find the lines. Filled regions (arrowheads, circles) are extracted by erosion and dilation operations. The image is eroded by a predetermined number of pixels to completely remove all of the lines. The eroded image is then dilated back to its original dimension, and a logical AND operation is performed to recover the solid symbols. The boundaries of these symbols are then found and used in symbol recognition.

1.4. Model-based object recognition

Model-based object recognition problem is essentially this: given *models* of objects (some knowledge of how certain objects may appear), plus an image of a scene possibly con-

taining those objects, report which objects are present in the scene and where. Typically, a model includes shape, texture and context knowledge about the occurrence of such objects in a scene.

Recognition is accomplished by finding a correspondence between certain features of the image and comparable features of the model. The two most important issues that a model-based recognition method must address are:

- what constitute a *feature* (*shape primitive*),
- how is the correspondence found between image features and model features.

Selection of *feature extraction* method is probably the single most important factor in achieving high recognition performance. Devijver and Kittler [74] define feature extraction as the problem of "extracting from the raw data the information which is most relevant for classification purposes, in the sense of minimizing the within-class pattern variability, while enhancing the between-class pattern variability".

The 2D shape representation methods loosely fall into five classes: 1) global features, 2) local features, 3) boundary description, 4) skeleton, 5) two-dimensional parts ([36, 44, 234]).

Global features, such as area, perimeter, circularity, elongation, Fourier descriptors ([379, 139, 257, 69]), and moments ([339, 340, 158, 183, 34, 207])) are not sufficient for distinguishing all shapes, they can be very sensitive to noise, and their values drastically change with occlusion. A 2-dimensional object can be characterized by its *local features* like *corners* (significant changes of curvature), their attributes and relations. Local features must be organized into some type of structure for matching ([36, 234]). The most common type of structure is a *graph* whose nodes represent local features and their properties, and whose edges represent relations among the features. There have been two common methods for organizing shape primitives into some sort of structure. The first is to arrange them hierarchically according to part/whole relations. The second, is to arrange them according to adjacency relations so that each primitive is related to others nearby. An adavantage of these organizations is that they provide a convenient way to organize additional information about the relative geometry of primitives.

The three main ways to represent the *boundary* of an object: as a sequence of points, by its chain code ([113]) and as a sequence of line segments, although, give full information on the shape of an object, may not be the most suitable information for matching particularly for line like objects.

An example of *skeleton* description of an object is its symmetric axis ([41]) (the locus of the centres of these maximal disks) plus the set of distances of these centres to the boundary of the object.

One early approach to object representation by *two-dimensional parts* was the decomposition of shapes into primary convex subsets ([296]).

The type or format of the extracted features must match the requirements of the chosen classifier. *Graph descriptions*, or *grammar-based descriptions* of the objects are well suited for *structural classifiers*, where correspondence is found by graph (prototype) matching or syntactic parsing, respectively. On the other hand, discrete features that may assume only, say, two or three distinct values, are ideal for decision trees, while real valued feature vectors are ideal for statistical classifiers.

In this framework, the models used are based only on object *shape*. *Shape* will mean the geometry of a locus points, which will be the points on an intensity edge or region in an image. *Shape representation* is a language for describing shape or some aspects of a shape. It includes a set of shape *descriptions*, and a mapping between shape descriptions and shapes.

For the different graphics/text components of map drawing we have developed within the framework *detectors* (classifiers), each of which is an algorithms for the detection of a specific graphics/text object, based on the appropriate *graphic/text object model*. These are the polygon classifier, and three structural classifiers: the classifier based on the combined global, quasi-topological and geometric representations of a 2D-curve, the classifier based on inexact graph matching, and the syntactic detector that relies on error-correcting parsing.

- The polygon detector is a simple *polygon detection* and *classification* procedure based on appropriate *polygon model*. It is used for the recognition of polygonal-type objects like *road, house, parcel*, each of which is a polygon of different type. Polygons are detected in Vector Graph representation of an image by locating of all minimal cycles using algorithm from [363]. Polygon classifier uses such criteria like S/L^2 (S, L - polygon area and perimeter), l/d (l, d are polygon axis length and breath), or the criterium of parallelity of opposite polygon sides. Polygon classifier is mainly used during *initial recognition* procedure.

- The developed second detector for different *geographic symbols*, as well as *text* ([314, 317]), described in chapter 9, relies on *relational model of a general 2D curve*. This model is described in terms of: global, qualitative, quasi-topological features which have been derived based on local features using systematic rules, as well as a geometrical structure of each *multiple (branch) point*. This qualitative and global object description is *robust* against *shape deformations*. Attached to structural components is the *quantitative* information (geometrical parameters such as size and position). The recognition method applied in this detector consists of: *hybrid matching*, that means matching *relational structures*, followed by *distance calculation* in a parameter space.

- The third developed detector ([323]), for recognition of *geographical symbols* and *text*, described in chapter 10, does not require learning phase as the previous one, and, can also cope with *topological deformations* caused by symbol connections and breaking. Objects recognized by this detector are represented by *attributed graphs*. The recognition method relies on *error-tolerant graph matching* procedure between two graphs representing a candidate, and a prototype, and is an example of *multiple-to-one graphs homomorphism*. Noise and shape variations are modelled as a series of *graph transformations* from group of features in the data to features in each prototype.

- The most general detector is the fourth one ([318]), described in chapter 11, which is a syntactic detector especially designed for *line objects* (like *scarps, borders, isolines*) but can also be applied to all other geographic objects which follow characteristic pattern (for example all geographic symbols, some sort of buildings). Objects are represented by the *programmed, higher-dimensional extension of a string grammar*. The recognition is performed by *error-correcting* (mixed top-down/bottom-up) *parsing* to handle specific kinds of errors in noisy and distorted objects in hand-drawn maps.

As the geographic objects are handprinted, different shape deformations are inevitable, the developed *structural detectors* are error-correcting ones.

All detectors are translational and rotational invariant. Rotation invariance in the first three detectors is achieved by a preliminary computation of the shape's orientation, and then, through a normalisation and a rotation step a pattern is obtained in a reference position. The computation of the orientation can be easily done, because symbols as well as text are drawn parallel or perpendicular to the axis of the geographic object they symbolized, and these axis are known before recognition due to the analysis strategy described in chapter 8. The fourth syntactic classifier is based on rotation invariant object description.

We have proposed the strategy for recognition of **overlapping shapes**. Due to special post-vectorization technique, linking neighboring primitives, the syntactic detector ([318]) together with the intelligent map analysis strategy enables correct recognition of overlapping objects in map drawings.

Chapter 6

Vector Graph construction module

1. Introduction

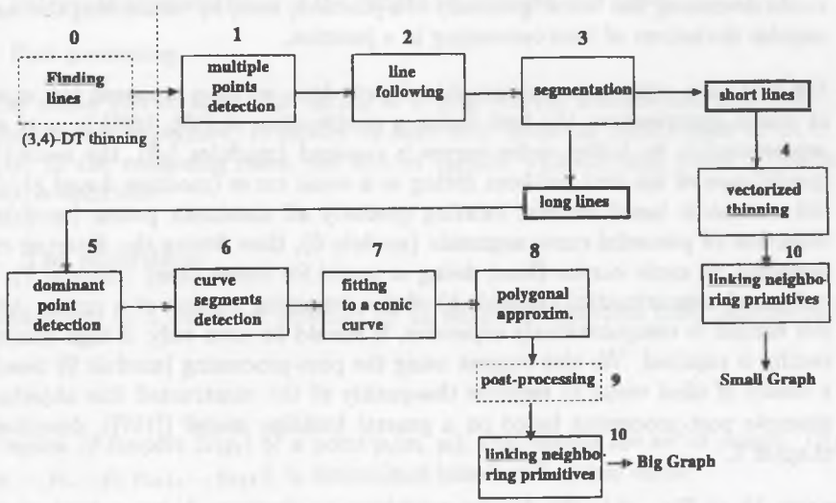


Fig. 6.1. The outline of the Vector Graph construction process. Dashed box denotes optional process. Process 10 is performed only in a limited region specified by a hypothesis

Rys. 6.1. Schemat konstrukcji grafu wektorowego. Prostokąt zaznaczony linią przerywaną oznacza proces opcjonalny. Proces 10 jest wykonywany tylko w ograniczonym regionie wyspecyfikowanym poprzez hipotezę

We have proposed the efficient vector representation of an image in the form of the Vector Graph ([319]), which is constructed in the *model-based vectorization* scheme. The outline of the overall Vector Graph construction process is depicted in Fig. 6.1. After (3,4)-DT thining algorithm ([23]) *finds the lines* (skeletons of objects) (module 0 in Fig. 6.1), the procedure for *detection of multiple points* (module 1) is performed by hit-or-miss

transform ([287]). Next, pixels composing the found lines are tracked between detected multiple points by the *line following algorithm* (module 2) whose outline is depicted in Fig. 6.14). This line following algorithm is based on that presented in [343], and takes advantage of topological properties of the skeleton. It starts after deleting isolated pixels, and marking all pixels of degree $\neq 2$, or of degree $= 2$ when 4 pixels make up a square. The next step is segmentation of the detected lines by length, into *short lines* representing characters and symbols, and *long lines* representing real geographic object edges (module 3). As each of these two types of lines has different characteristics, it requires separate processing. We have proposed the following two different processing schemes, each of which is appropriate for specific line.

- For *short* lines, representing symbols and text we proposed *vectorized thinning* (published in [315]) (module 4), which starts again from thick lines. Due to special method of feature extraction based on *decomposition approach*, the major disadvantage of all thinning based vectorization methods: distortion at junctions, is avoided. Finding correct position of a junction is accomplished by fitting to the *model* describing the "ideal"geometry of a junction, here, by minimizing the sum of angular deviations of lines converging in a junction.

- For *long* lines, representing geographic object edges, we have proposed two variants of *mixed vectorization*: the first (being a modification of [166, 168]) is used when representation by higher order curves is required (modules 5-8), the second is a special case of the first, without fitting to a conic curve (modules 5 and 8). This full variant is based on first locating precisely all dominant points (module 5), detection of potential curve segments (module 6), then fitting the detected curve segments to conic curves (here, being a *model* for curve lines) (module 7), and polygonal approximation (module 8) of the remaining portion of a curve. As the full variant is computationaly expensive, it should be used only, if high quality of results is required. We also suggest using the post-processing (module 9) based on a *model of ideal result* to improve the quality of the constructed line objects, for example post-processing based on a general building model ([157]), described in chapter 4.

Process 10 in Fig. 6.1, the *linking neighboring primitives*, being a kind of post-vectorization procedure, is required by the syntactic detector described in chapter 11. It is performed not on whole image, but only on vectors in a limited regions precisely specified by hypotheses about given, sought object, according to the analysis strategy described in chapter 8. This post-vectorization procedure is performed each time a syntactic detector is called to detect a certain line object, or symbol in a specified area (if not yet performed).

Characters and symbols may overlap graphics (lines), or themselves. We do not use the existing, 'blind' procedures for text/symbols/graphics segmentation, described in chapter 4. Instead, in the proposed framework this segmentation is performed by the proposed, intelligent analysis strategy described in chapter 8, together with the syntactic detector described in chapter 11, capable of overlapped objects recognition.

The following two sections describe the above two processings of segmented long and short elements.

2. Big Graph vectorization procedure

For processing long lines representing geographic object edges, we have proposed the following scheme (being a modification of [166, 168]) by different critical point detection procedure).

1. Critical point detection.
2. Potential curve segments detection.
3. Fitting curve segments to a conic section by the least square technique.
4. Polygonal approximation of the remaining portions of a curve.
5. Post-processing.

The above full variant, with fitting to a conic curves is somewhat specialized, and computationaly expensive. It should be used only when the results need to be of high quality. In the remaining cases, the limited variant, without steps 2 and 3 (modules 5 and 8), is sufficient.

2.1. The algorithm

Critical points on a curve are detected by an adaptive algorithm [263], shortly outlined here.

⋆

The *region of support* $D(p_i)$ of a point $p_i(x_i, y_i)$, that means the set of points: $D(p_i) = \{p_{i-k}, ..., p_{i-1}, p_i, p_{i+1}, ..., p_{i+k}\}$, is determined based on k-cosine value:

$$Cos_{ik} = a_{ik} \circ b_{ik}/|a_{ik}||b_i k|$$

where:
$a_{ik} = (x_i - x_{i-k}, y_i - y_{i+k})$,
$b_{ik} = (x_i - x_{i-k}, y_i - y_{i-k})$
The value of k that determines the radius of a region of support of a point p_i, is the first value for which $|Cos_{i,k+1}| > |Cos_{ik}|$, or if $Cos_{ik} = Cos_{i,k+1}$, the greatest k for which this relation holds, or if $|Cos_{ik}| = |Cos_{i,k+1}|$ but are of opposite signs, then the least value of k for which it happens gives the radius k of a region of support of a point p_i.

To detect significant points, as a measure of significance, the following *smooth cosine value* is used:

$$Cos_i = 1/k_i \sum_{j=1}^{k_i} Cos_{ij}$$

where k_i is a found radius of a region of support of a point p_i. The procedure for critical points detection is performed in two passes. In the 1-st pass only those points p_i are retained which are either curvature maxima points:

$$Cos_i \geq Cos_j \text{ for all } j \text{ satisfying:}$$
$$|i-j| \leq k_i/2,\ k_i > 1$$
$$|i-j| \leq k_i,\ k_i = 1$$

or curvature minima points:

$$Cos_i \leq Cos_j \text{ for all } j \text{ satisfying:}$$
$$|i-j| \leq k_i/2,\ k_i > 1$$
$$|i-i| \leq k_i,\ k_i = 1$$

In the 2-nd pass, if a curvature minimum point falls within the region of support of a curvature maximum point, then only the maximum point is retained.

⋆

The procedure which *detects the potential curve segments* examines all detected critical points, and forms appropriate groups, which could constitute a curve. At each critical point R_k, the points R_{k-1}, R_k, R_{k+1} are labelled as belonging to curve segment, if the following conditions are satisfied:

- The angle between line segments $R_{k-1}R_k$ and R_kR_{k+1} is greater than a prescribed value θ, (in the presented method $\theta = 135°$ has been chosen). This condition identifies sharp corners and terminates continuing arcs at this point (Fig.6.2a)).
- The ratio of the length of these segments does not exeed w, where w is the function of the local curvature (in the presented method w between 2 and 3 has been chosen). This condition terminates continuing arcs at tangent points (Fig.6.2b)).
- The sign of the angle $R_{k-1}R_kR_{k+1}$ is the same as that of $R_{k-2}R_{k-1}R_k$. This condition ensures that as the critical points are traversed, the curvature of the arc has the same sign (Fig. 6.2c)).

As a result of applying the above conditions, the critical points constituting a given curve are partitioned into two groups: those that could belong to a curve segments (potential curve segments), and the remaining which are approximated with line segments.

Each detected potential curve segment l is *fitted to a conic section*, (being a *model* of a curve) which can be mathematically represented by the second degree polynomial:

$$F(x,y) = ax^2 + bxy + cy^2 + dx + ey + f$$

The coefficients a through f of the approximating conic section l, with its end points Cb^l and Ce^l, and intermediate points: $R_i^l:\ i = u, \ldots, v;\ R_u = Cb^l;\ R_v = Ce^l$ (e.g. see Fig. 6.3a)) are obtained by minimizing the sum of squares:

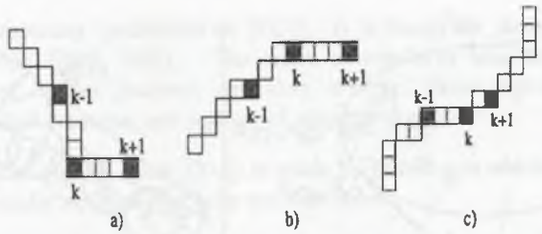


Fig. 6.2. Examples of curves with detected critical points which are gray: a)-c) illustrate the conditions necessary to partition a curve into potential curved segments

Rys. 6.2. Przykłady krzywych wraz z punktami krytycznymi (zaznaczone na szaro). a)-c) ilustracja warunków podziału krzywej na segmenty

$$\sum_{i=u}^{v} F^2(x_i, y_i)$$

If the curve segment is not a closed one, two additional constraints are introduced, since the curve is required to pass through the end points Cb^l and Ce^l:

$$F(x_u, y_u) = F(x_v, y_v) = 0$$

The conic section obtained using the least square technique is then compared against the actual data points to determine the acceptability of the fit using an error norm:

$$N = \max_i(\min(D_{xi}, D_{yi}))\ i = 1, \ldots, n$$

where:
D_{xi} is the horizontal distance from the data point i to the conic,
D_{yi} is the vertical distance from point i to the conic (e.g. see Fig. 6.3a)).

If the computed value N is less than the specified error tolerance Tm, the fit is considered to be acceptable, otherwise such curved segment l is fitted to a straight line segments in polygonal approximation.

The algorithm of *polygonal approximation* searches for a subset $\{1, \ldots, n\}$ of a given set of data points $\{1, \ldots, N\}$ $(n \leq N)$ such that the polygonal curve formed by joining every pair of successive points from a found subset $\{1, \ldots, n\}$ by a straight line segment, lies within distance T_p (where T_p is a specified treshold) from all the points $\{1, \ldots, N\}$. Fig. 6.3b) shows an example of the polygonal approximation of data points $\{1, 2, \ldots, 9\}$. For each point i $(i \leq 8)$ all its preceding points j $(j < i)$ lie within a specified distance T_p from a line segment *1-i* (from 1 to i). Fig. 6.3b) shows the situation after successful examining of the threshold condition for line segment *1-8* (circles drawn with continuous line), and at the moment of testing for line segment *1-9*, in which the case threshold conditions are violated (for instance for a point $i = 8$ the distance from the segment *1-9* exceeds T_p). So, at the point $i = 8$ a new line segment should begin and as a result the polygonal curve *1-8-9* (from 1 through 8 to 9) approximates a given set of points $\{1, 2, \ldots, 9\}$.

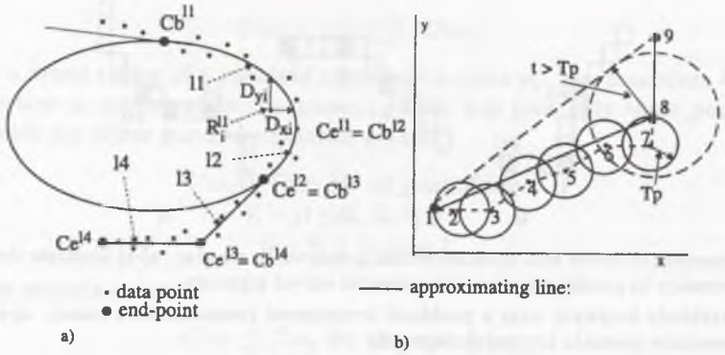


Fig. 6.3. Vectorization. a) Fitting the data points with two conic curves l1, l2 and two line segments l3, l4. b) The idea of the algorithm for polygonal approximation. Data points 1, ... , 9 are fitted with a piecewise linear curve 1-8-9

Rys. 6.3. Wektoryzacja. a) Dopasowanie punktów do krzywych stożkowych l1, l2 oraz dwóch odcinków prostych l3, l4

The proposed, mixed vectorization scheme is based on a combination of both feature detection methods: first dominant points detection, which are sufficient to describe a shape of an object, then polygonal approximation. This allows to avoid the main disadvantage of polygonal approximation that some feature points may be missed, or shift. Fitting procedure enables accurate description of curve segments.

As it was said in chapter 4, the vectorization very often leads to the results which are contrary to what the draftsman expects, mainly due to its blindness. In order not to lose the quality of the vector data while constructing primitives into line objects, and to satisfy the user's requirements, it is often necessary to add some contextual knowledge as post-processing step. Examples of such constraints are geometric constraints like requiring segments to be parallel, or perpendicular, circular arcs to be concentric, tangents of the primitives to be equal at their connection point, etc. We suggest to use the post-processing based on a *model of ideal result* like for example post-vectorization based on general building model ([157]), described in chapter 4.

3. Small Graph vectorization procedure

Short lines representing symbols and characters are recognized by structural classifiers which require skeletons not distorted by feature extraction process. The classical thinning-based vectorization algorithms (based on pixelwise thinning) have the intrinsic limitation due to local processing. Their output is hard to be used for structural analysis of curves because the figures of branch points and corners are distorted and counterintuitive. Therefore, for short lines we have developed the vectorization method based on non-pixelwise thinning, being the example of contour-based vectorization,

called *vectorized thinning* (published in [315]). It is based on *decomposition approach* to feature detection ([303, 328]). The main *principle* of this approach is: "First find the regions of *regular features*, and then subtract these regions from the iconic representation. What remains, are regions of *singular features*".

Definition 6.1 The *regular class* ([303]) is made up of features of which iconic representation is periodic or quasi-periodic such as lines.

Definition 6.2 The *singular class* ([303]) is made up of features which are the complements of the features of the regular class. These are 'accidents' of the regular features: extremities, crossings, forks, endpoints.

Thus, the proposed vectorized thinning is based on decomposition of an image according to the suggestion from [303], into *regular parts* (lines), and *singular parts* (branching, crossing, end-point). First, the regular parts are analyzed and described as thin *lines*. Next, the residual parts (singular parts), are analyzed in conjunction with their neighboring regular parts, and they are described as junctions of lines, or endpoints (*multiple points*). The output from the vectorized thinning algorithm is a piecewise linear curve expressed in terms of the undirected graph $G = (J, F)$, where J is a set of nodes, and F is the set of edges. Nodes of a graph represent multiple points, while edges correspond to to simple, or closed curves.

The main advantages of the proposed vectorization algorithm lie mainly in the better extraction of multiple points representing corners, branch or crossing regions of 2D objects, and less sensitivity to a boundary noise which are the main problems in the thinning-based vectorization algorithms.

3.1. The algorithm

The vectorized skeleton of a 2D-curve produced by the proposed vectorized thinning algorithm consists of *medial lines* and *multiple points* constructed from two kinds of regions: *simple regions* (e.g. see Fig. 6.6a) and *complex regions* (e.g. see Fig. 6.7), respectively. Each *simple region* corresponds to a straight or curve line segment, while each *complex region* is a representation of an end-point region, a corner, a branch, or a crossing region of line-like 2D-objects. The algorithm for construction of a *vectorized skeleton* of an object consists of the following three steps:

1. Links finding and simple region extraction.
2. Complex region extraction and multiple point finding.
3. Transformation into vectorized skeleton.

3.1.1. Links finding and simple region extraction

The overall flow of this step is as follows. Boundary pixels of a given 2D-object are extracted during contour-following algorithm ([254]). A 3x3 Sobel operator is used to

find gradient of edges which results in a perpendicular to an edge direction at each boundary pixel. Each boundary pixel P has assigned a label, $Label(P)$, and initially the labels of all the boundary pixels are set to 0 which means that they belong to no *simple region*. For each boundary pixel, its possible *links* are calculated, and during *simple region* construction described in detail below, only one is selected.

Before we describe the links calculation, some needed definitions are introduced. It is assumed that a set of boundary pixels in a raster image is 8-connected, an image has N contours, and each contour is traced in such a way that the interior lies on the left side of the trace. On the j-th contour boundary pixels are indexed cyclically from a specified starting pixel as $P_j(i)$, for $i = 0, ..., n_{j-1}$, and $P_j(i) = P_j(i \bmod n_j)$, for $i \geq n_j$, where n_j is the number of pixels on the j-th contour. The pixel $P_j(i+1)$ is called the *after-neighbour* of the pixel $P_j(i)$, and denoted $aft(P_j(i))$. Similarly, the pixel $P_j(i-1)$ is called *before-neighbour* of the pixel $P_j(i)$, and denoted $bef(P_j(i))$.

Let P be a boundary pixel. The line is drawn by Bresenham's algorithm from a pixel P, along a direction perpendicular to the edge direction at a pixel P (found by Sobel operator) until a boundary pixel R, lying on the opposite side is found, which we call an *orthogonal neighbour* of a given boundary pixel P, and denote it $ort(P)$ (e.g. see Fig. 6.4a)). If a background pixel R', not belonging to a boundary is found (e.g. see Fig. 6.4b)), the *orthogonal neighbour* is the nearest boundary pixel R belonging to a boundary, which is a 4-neighbour of R'.

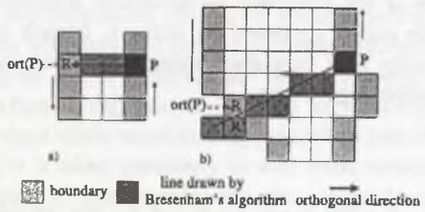


Fig. 6.4. Illustration of an orthogonal neighbour of a given boundary pixel. a) The case in which a boundary pixel is found in the orthogonal traversal, b) the case in which a background pixel is found in the orthogonal traversal

Rys. 6.4. Ilustracja pojęcia ortogonalnego sąsiada punktu brzegowego. a) Przypadek, w którym znajdowany jest punkt brzegowy, b) przypadek, w którym znajdowany jest punkt tła

The *link* between boundary pixels P and R, denoted $L(P, R)$, is an ordered pair of the boundary pixels P, R, such that at least one of the following conditions is satisfied:

1. $P = ort(R)$ and a distance $d(R, ort(P)) \leq 1$,
2. $R = ort(P)$ and a distance $d(P, ort(R)) \leq 1$,

where $d(P_1, P_2)$ denotes a distance (number of boundary pixels) between two boundary pixels P_1, P_2. Figures 6.5a), b) illustrate the above conditions for extraction of a link between two boundary pixels. From the above definition it follows that the direction of each link $L(P, R)$ is almost perpendicular to the direction of a line.

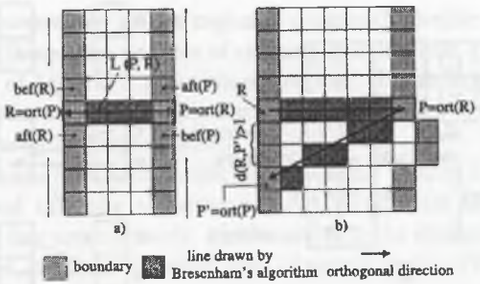


Fig. 6.5. Illustration of an extraction of a link between a pair of two boundary pixels P, R. a) The case in which pair of pixels P, R could create a link, b) the case in which pair of pixels P, R could not create a link

Rys. 6.5. Ilustracja procedury ekstrakcji przekroju pomiędzy parą dwóch punktów brzegowych P, R. a) Przypadek, w którym para punktów P, R może tworzyć przekrój, b) przypadek, w którym para punktów P, R nie może tworzyć przekroju

Before we describe the algorithm for construction of a *simple region*, each of which is created from the calculated as described above links, we first give some needed definitions. Let $L1 = L(P, R)$ and $L2 = L(P', R')$ be the two, different links. A link $L2 = L(P', R')$, is defined as the *next link* of a link $L1 = L(P, R)$ and denoted $next(L1)$ if the following condition is satisfied:

- $(P' = aft(P)$ or $P' = P)$ and $(R' = bef(R)$ or $R' = R)$ (e.g. see Fig. 6.4a)

Similarly, a link $L1 = L(P, R)$ is defined as the *previous link* of a link $L2 = L(P', R')$ and denoted $prev(L2)$ if the following condition is satisfied:

- $(P = bef(P')$ or $P = P')$ and $(R = aft(R')$ or $R = R')$ (e.g. see Fig. 6.4a)

Formally, a *simple region* SR is defined as a list of k links ($k > 0$):

$$SR = (L_0, L_1, ..., L_{k-1})$$

such that for any pair of links: (L_i, L_{i+1}) (for $0 \leq i \leq k-2$) the following condition is satisied:

- $L_{i+1} = next(L_i)$ $(L_i = prev(L_{i+1}))$ and $next(L_{k-1}) \in \{L_0, \emptyset\}$ and $prev(L_0) \in \{L_{k-1}, \emptyset\}$.

Links L_0 and L_{k-1} are defined as *tail* and *head* of a given simple region SR and denoted as $tail(SR)$ and $head(SR)$, respectively (e.g. see Fig. 6.6a).

Now, we describe the algorithm for construction of a simple region according to the definition given above. A simple region is constructed by repeating the procedure that selects the next link (to the current link), and concatenates it to the current link, thus forming a growing region. Let $L(P, R)$ be the current link created from pixels P and R, and $P' = aft(P)$, $R' = bef(R)$ be the two boundary pixels which are the after-neighbour

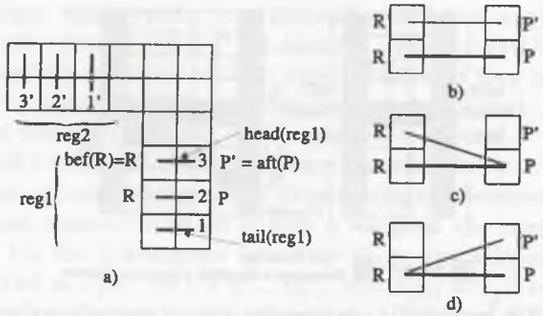


Fig. 6.6. a) Illustration of the basic notions in simple region definition, b), c) d) illustration of conditions for simple region extraction. Links are depicted as segments connecting a pair of creating it pixels

Rys. 6.6. a) Ilustracja pojęcia regionu prostego, b), c), d) ilustracja warunków ekstrakcji regionu prostego. Przekrój jest zaznaczony jako para tworzących go pikseli

and before-neighbour of the pixel P and R, respectively (e.g. see Fig. 6.6b)-d)). At each elementary step of the concatenation process, only one of the following adjacent links: $L(P', R')$, $L(P, R')$, $L(P', R)$ or none is selected as the next link, and then concatenated to the current link. The rules for selection of one, next link are the following.

1. The link $L(P', R')$ is selected as the next to the current link $L(P, R)$ if: 1) pixels P', R' could create a link, and 2) both do not belong to any simple region yet (their labels are set to 0) (e.g. see Fig. 6.6b)). If link (P', R') could not be selected because one or both of the above two conditions are violated, the two remaining adjacent links are considered: $L(P', R)$ and $L(P, R')$.

2. Trying the potential link $L(P, R')$, it is selected as a next adjacent link if pixels P, R' could create a link, and pixel R' does not belong to any simple region yet (e.g. see Fig. 6.6c)).

3. Trying the potential link $L(P', R)$, it is selected as the next link if pixels P', R could create a link, and pixel P' does not belong to any simple region yet (e.g. see Fig. 6.6d)).

In the case of a situation when both remaining links: $L(P, R')$ and $L(P', R)$ are possible, we choose the link $L(P', R')$ as the next link.

Each simple region has its own, unique label. If a link has been chosen during construction of the given simple region, its two forming pixels are assigned labels equal to the label of the given simple region, indicating the fact that they belong to it.

If the adjacent link is selected in one of the above cases, it extends the current simple region after the modification of the labels of pixels P' and R', and the concatenation process continues. Otherwise the concatenation process is terminated and the current link is regarded as the head of the current simple region.

Next, each such constructed simple region is examined whether it is necessary or not in terms of its length (being the number of creating it links) and its thickness (measured as an average length of links in a given simple region). If the length of a simple region is smaller than its thickness, such simple region is removed (e.g. see Fig. 6.8b), and this process is called *cleaning* of simple regions.

Sampling several links from each simple region every T (where T is a chosen treshold), including its head and tail, the algorithm creates vertices on their center points and connects them with a line appropriately, which results in the *medial line* of simple region. (e.g. see Fig. 6.8c) for example of medial lines of simple region $SR1, SR2, SR3$).

3.1.2. Complex region extraction and multiple points finding

A *complex region* is extracted from 2D object as a region remaining after simple region extraction. The algorithm for extraction of a complex region traces the boundary pixels not belonging to any simple region (those with labels equal to 0). If it meets the boundary pixel p with label greater than 0, it searches for the head (or tail) of the simple region containing p, and picks out the other boundary pixel in it, to continue tracing. The search is terminated when it returns to the starting pixel, and a region which is surrounded by such performed tracing is extracted as a complex region. In the above tracing the bounding box of a complex region is calculated which will be used to examine the complex region.

Each complex region is then represented by the pixel, called *multiple point*, lying inside a given complex region, so as this point joins up smoothly lines, being medial lines of simple regions surrounding a given complex region. The computation of a multiple point is as follows. (Fig. 6.7 is an illustration of obtaining a multiple point in a given complex region).

Let L_k $(k = k_1, ..., k_n, n > 0)$ be the simple regions surrounding a given complex region CR, where n is the degree of connections of CR (the number of simple regions converging at a given complex region). Let vL_k be the vector having the direction equal to the direction of medial line of the *k-th* simple region near a given complex region. (It represents the direction of L_k near a given complex region CR). Let w_k be the vector $(x - x_k, y - y_k)$ where (x, y) are the coordinates of an object pixel X inside CR, and (x_k, y_k) are the coordinates of the center of the $head(L_k)$ (or $tail(L_k)$). To find a position of a multiple point of a given complex region, the following value A is calculated for each pixel X in a complex region:

$$A = \sum_{k=k_1}^{k_n} angle(vL_k, w_k)$$

where $angle(vL_k, w_k)$ is the absolute value of an angle between vectors: vL_k and w_k.

The closer the point X is to the vectors which show the directions of the adjacent to a given complex region simple regions, the lower the evaluated value A is. The pixel X which minimizes the evaluated value A is selected as a proper, multiple point (in Fig. 6.7 pixel $M1$ is selected).

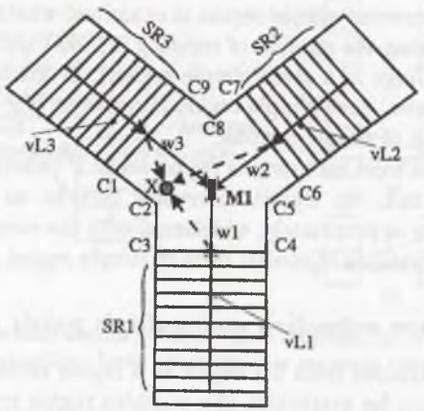


Fig. 6.7. Illustration of finding a multiple point M1 in a complex region C1...C9. Links are depicted as segments connecting a pair of creating it pixels

Rys. 6.7. Ilustracja procedury detekcji punktu wielokrotnego M1 w regionie złożonym C1...C9. Przekrój zaznaczono jako parę tworzących go pikseli

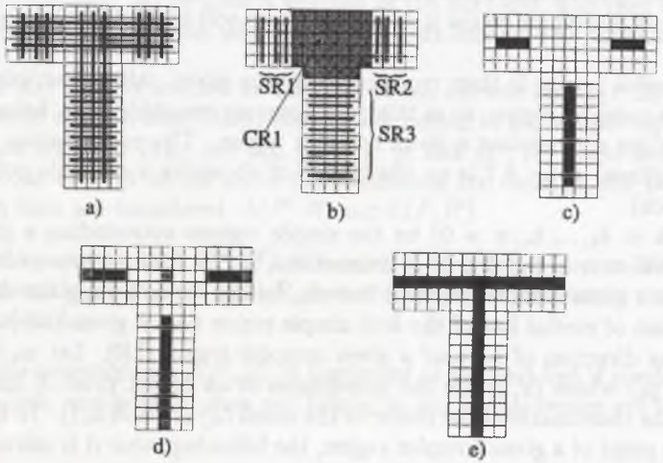


Fig. 6.8. Illustration of the stages of an algorithm for finding a vectorized skeleton of an T-shaped object: a) links finding and simple region extraction, b) intermediate structure after cleaning: three simple regions: SR1, SR2, SR3 and one complex region CR1, c) medial lines of simple regions found, d) multiple points found in a complex region, as well as in end-point regions, e) the resulting skeleton

Rys. 6.8. Ilustracja kolejnych kroków algorytmu wyznaczania szkieletu wektorowego litery T: a) znajdowanie przekrojów i ekstrakcja regionów prostych, b) struktura pośrednia po operacji czyszczenia: trzy regiony proste: SR1, SR2, SR3 oraz jeden region złożony CR1, c) linie środkowe regionów prostych, d) punkty wielokrotne w regionie złożonym oraz regionie punktu końcowego, e) uzyskany szkielet wektorowy

3.1.3. Transforming an intermediate structure into vectorized skeleton

From cleaned, intermediate representation, consisting of medial lines and multiple points, a vectorized skeleton of an object is constructed. In order to obtain a connected skeleton of a given object, the medial lines of all the simple regions converging at a given complex region must be connected to each other in a multiple point representing it. Each multiple point is used as a vertex and connected to the midpoints of heads or tails of adjacent simple regions. Figure 6.8 shows the steps in creating a vectorized skeleton for the T-shaped object: links finding and simple region extraction (a), complex region extraction, cleaning (b), medial lines of simple regions finding (c), multiple points finding (d), connecting medial lines with multiple points into skeleton of an object (e).

3.2. Experimental results

In order to examine the properties of the skeletons produced by the presented, vectorized thinning algorithm, we compare them with skeletons obtained from Hildith algorithm [284], being the pixelwise algorithm. The Hilditch algorithm has been chosen as the representative among other, pixelwise thinning algorithms tested (Pavlidis [254], SPTA [243], Guo/Hall [142], Zhang/Suen [381], Wang/Zhang [363], OPPTA [170]), because of its relatively good properties in most situations.

The examples of skeletons of the selected cartographic symbols and characters resulting from the presented algorithm, and from Hilditch algorithm, are shown in Figures: 6.10, 6.11, 6.12, 6.13. These figures show the typical examples of differences between the pixelwise algorithms and the presented one. Multiple points are extracted correctly by the proposed thinning algorithm, while it is one of the main problems of pixelwise methods. A proper extraction of multiple points is due to a fact that a complex region can be analysed from the adjacent simple regions.

Figure 6.12a) is an example of a noisy version of a character H, to illustrate the robustness against noise. Visual inspection indicates that the proposed vectorized thinning algorithm is less sensitive to a boundary noise than Hilditch algorithm. This is due the fact that the line image data are observed globaly.

Although there is no exact definition of a digital skeleton, it has been generally accepted that a good thinning algorithm should possess a number of fundamental properties: preserving connectivity of skeleton, converging to skeleton of unit width, closely approximating the medial axis, possessing insensitivity to a boundary noise. Preserving connectivity of connected component is essential for shape analysis, because the topological features of an 8-connected component may change completely, if it becomes diconnected.

The described thinning algorithm preserves connectivity and produces skeletons of unit width, that means 8-connected, which results directly from a construction method in the presented algorithm (with one exception, in multiple points where it is more important to preserve not distorted image of such regions). Connectivity is based on the connectivity in the vectorized skeleton which preserves the connectivity of the boundary pixels. Convergence to medial axis is good, and follows directly from the definition of the

medial lines of simple regions.

The main advantages of the proposed vectorized thinning algorithm are: correct extraction of multiple points representing corners, branch or crossing regions of objects, less sensitivity to a boundary noise which is particularly suitable for a further, structural recognition.

4. Vector Graph representation

Vector Graph ([319]) of an image, being the representation of spatial data in vector form, is a set of attributed undirected graphs. Each single graph of a Vector Graph of an image corresponds to one connected component (CC) in the image, and represents its topology. Nodes of the graph corresponds to line crossings, arcs to lines. The Vector Graph of an image is encoded in a designed, efficient structure stored in two tables. The first table, the *node table*, stores information about end-points and multiple points. For each point its (x,y) coordinates are stored, as well as an ordered list of lines converging in this point. The second table, the *line table*, stores information about each line in the graph: its begin and end points, all intermediate points, as well as its width. In case of conic curve segment, six coefficient are stored in a table. The example of a Vector Graph and its tabular representation are shown in Fig. 6.9 and Tables 6.1, 6.2, respectively.

Spatial analysis, being the main function of each GIS ([99]), needs efficient indexing methods to be able to answer spatial queries in reasonable time. Though the output from *MAPIN* is stored in a commercial GIS system, we also developed our own, efficient algorithms for spatial data indexing ([310, 311]) which could be used in each GIS engine.

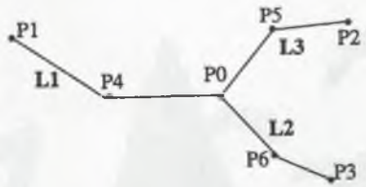


Fig. 6.9. An example of the Vector Graph (for one CC)

Rys. 6.9. Przykład grafu wektorowgo (dla jednego CC)

multiple point	x, y	line list
P0	3.2	L1, L3, L2
P1	1.1	L1
P2	4.5	L3
P3	5.0	L2

Table 6.1. The structure of Vector Graph in Fig. 6.13: multiple points representation (node table)

line	width	intermediate points	begin	end
L1	0.18	P1, P4, P0	P1	P0
L2	0.18	P0, P6, P3	P0	P3
L3	0.18	P0, P5, P2	P0	P2

Table 6.2. The structure of Vector Graph in Fig. 6.13: line representation (line table)

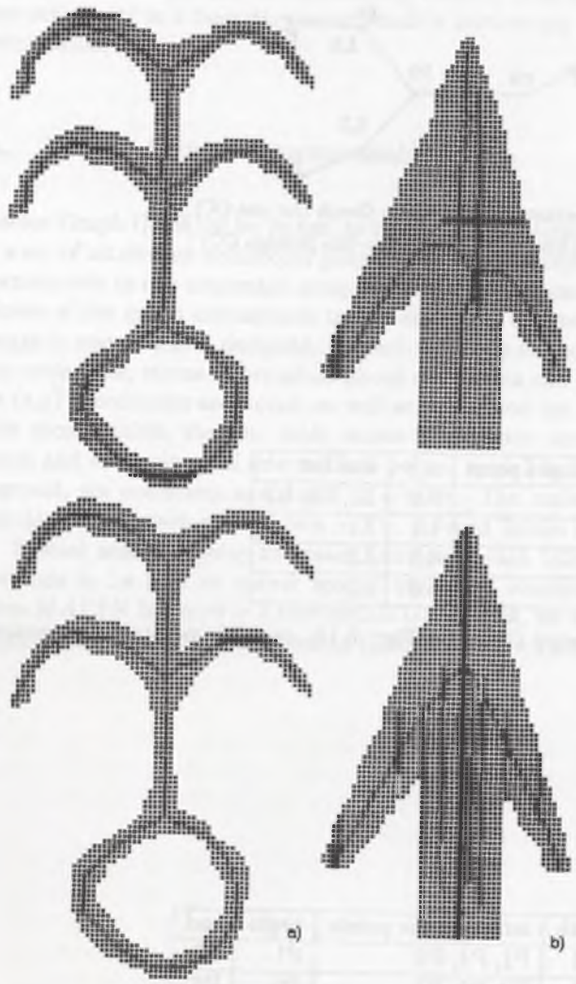


Fig. 6.10. Examples of skeletons produced by the Hilditch algorithm (1-st row) and the presented one (2-nd row) for the selected geographic symbols: a) fountain, b) river arrow

Rys. 6.10. Przykłady szkieletów uzyskanych za pomocą algorytmu Hilditch (1-szy rząd) oraz ścieniania wektorowego (2-gi rząd) dla wybranych symboli geograficznych: a) fontanna, b) strzałka rzeki

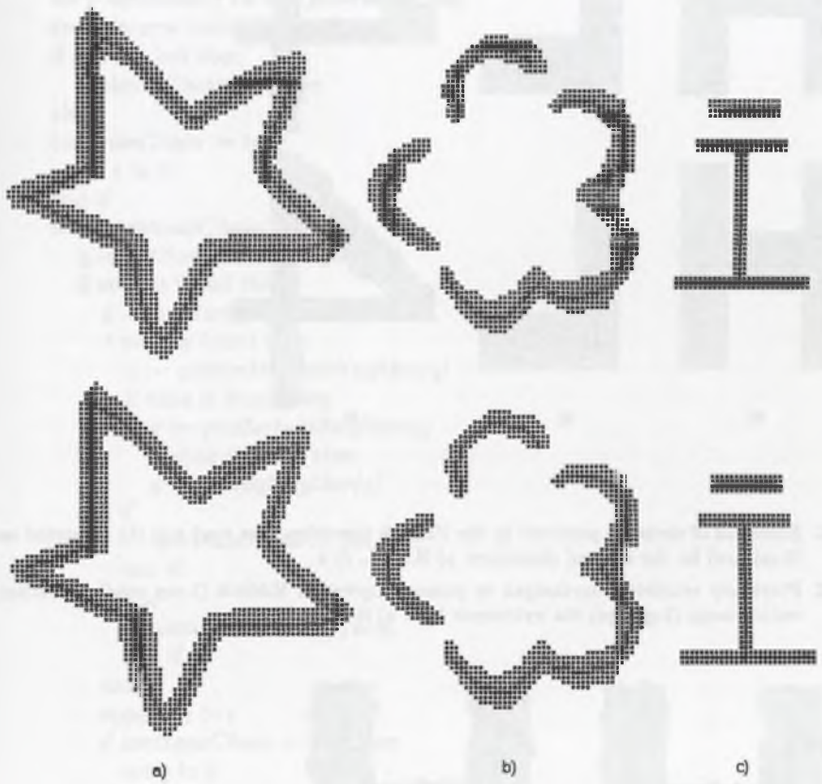


Fig. 6.11. Further examples of skeletons produced by the Hilditch algorithm (1-st row) and the presented one (2-nd row) for the selected geoographic symbols: a) leafy tree, b) coniferous tree, c) blind tripod on water supply

Rys. 6.11. Dalsze przykłady szkieletów uzyskanych za pomocą algorytmu Hilditch (1-szy rząd) i ścieniania wektorowego (2-gi rząd) dla wybranych symboli geograficznych: a) drzewo liściaste, b) drzewo iglaste, c) trójnik zaślepiony na przewodzie

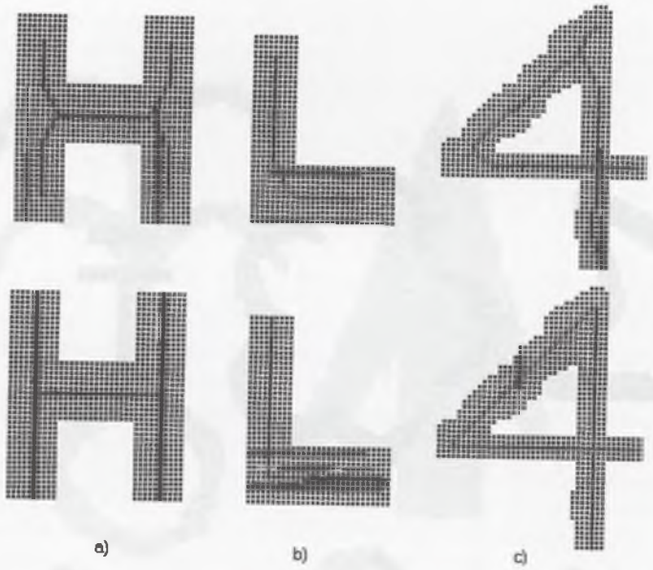


Fig. 6.12. Examples of skeletons produced by the Hilditch algorithm (1-st row) and the presented one (2-nd row) for the selected characters: a) H, b) L, c) 4

Rys. 6.12. Przykłady szkieletów uzyskanych za pomocą algorytmu Hilditch (1-szy rząd) i ścieniania wektorowego (2-gi rząd) dla wybranych liter: a) H, b) L, c) 4

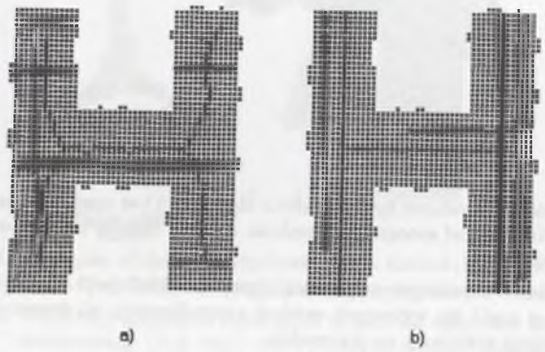


Fig. 6.13. Example of skeleton of noisy character H produced by the Hilditch algorithm (a) and the presented one (b)

Rys. 6.13. Przykład szkieletu zaszumionej litery H uzyskany za pomocą algorytmu Hilditch (a) oraz ścieniania wektorowego (b)


```
while there are marked points left do
  choose a marked point p
  create list l_p of all non-null neighbors of p
  while l_p ≠ 0 do
    choose a point q from l_p
    set p temporarily to 0 to prevent looping
    create a new chain c as [p, q]
    if q is marked then
      continueChain := false
    else
    continueChain := true
      set q to 0
    end if
    while continueChain = true do
      q := getNonMarked4Neighbor(q)
      if none is found then
        q := getMarkedneighbor(q)
        if none is found then
          q := getNonMarked8Neighbor(q)
          if none is found then
            q := getMarked8Neighbor(q)
            if none is found then
            q: =getAnyNeighbor(q)
      end if
            continueChain := false
          end if
          else
            continueChain := false
          end if
        end if
        append q to c
        if continueChain = true then
          set q to 0
        else if we are on a loop then
      add p to c again to finish the loop
        end if
    end while
    add c to the list of chains
    restore mark on p
  end while
  set p to 0
end while
while there are pixels still unprocessed do
  start over again by marking a non-null pixel
end while
```

Fig. 6.14. Line following algorithm

Rys. 6.14. Algorytm śledzenia linii

Chapter 7

The map model

1. Syntax

The proposed *map model* is represented in the form of a *hybrid semantic network* which is a combination of the different knowledge representation formalisms: *semantic network*, *rules*, *frame* system ([204, 268]), and the *object-oriented* computation paradigm ([231]). This can contribute to a great improvement with respect to expressive power. Similarities of various knowledge representation formalisms have constitute a basis to formulate hybrid knowledge representation which takes adavantage of the salient features of individual ones.

Semantic network is suitable for the representation of complex objects, particularly for describing how an object is built up from the simpler parts, and how it is related with the other objects. When regarding map analysis, the nature of the knowledge to be represented, i.e. a network of objects and their spatial relationships, seems to be reflected most adequately by the semantic network. Rule-based approach is most suited for a domain with many independent states, when the control flow consists of a sequence of independent actions and the knowledge can be represented as a collection of independent chunks of knowledge.

As opposed to rules, where we find fix inference strategies based on forward- or backward chaining, an inference in semantic network is application-dependent, and is to be provided by the user which results in a great number of different reasoning procedures. Thus, in this sense semantic network has more expressive power than rules.

Frame representation obey a major weakness of semantic networks, i.e. the difficulty to represent procedural knowledge. Rules are integrated into a semantic network by attaching them to the slots of a frame, which play also a role of a procedural component.

A weakness common to semantic networks and frames is that knowledge representation lacks a well-defined structure. Knowledge abstraction, encapsulation, and modularity which are considered as salient features of knowledge representation, are not efficiently realized in the semantic networks and frames. These characteristics, however, can be captured by the object-oriented approach. Through encapsulation, it improves the consistency, understandability, and maintainability of a knowledge base. It supports knowledge

abstraction, through the process of inheritance.

In the proposed framework, each object instance uses its own knowledge (a set of stored rules specifying its behaviour) to generate hypotheses about its related objects, and to perform local reasoning based on whether the hypotheses are verified or not. This makes the presented scheme an *object-oriented* one.

The proposed hybrid semantic network is organized in four levels: Image (level I), Vector Graph (level II), Graphics/Text (level III), Domain Model (level IV). The *Image* level represents the raster image supplied by the scanning process. The *Vector Graph* level stores an image in an efficient vector representation. The *Graphics/Text* level describes the taxonomy of 2D basic and composite graphics and text primitives. In the *Domain Model* level (also called the *Geographic Objects* level), the knowledge about the investigated types of maps is represented.

Figures 7.1, 7.2 show two fragments of a hybrid semantic network for situational layer of a PFLM. This network consists of *nodes* describing properties and structure of a class of objects, and the following four types of *relations*:

- *Hierarchies*:
  - *Part* (*PO*): connects a class with other classes which represents its components, or parts.
  - *Specialization* (*S*): connects a general class with its specialized class.
- *Geometric* (*GE*) relation: connects a pair of classes and puts some geometrical constraints on instances of both classes, like adjacency, intersection, parallelism, inclusion, colinearity, etc.
- *Reverse* (*RV*) relation: each pair of classes which do not satisfy some GE relation specified in a model, are connected by appropriate RV relation (not explicitly shown in a model). This is a GE relation which cannot exist in the domain, for example the relation *INSIDE* between a *HOUSE* and a *ROAD* instances.

The term *relation* is used in this paper to denote a link between two classes, as well as between two object instances of the corresponding classes.

The GE and RV relations are allowed only at the Domain level. Moreover, with each GE and PO relation in a Domain level a *weight* is assigned (not shown in Fig. 7.1), and both relation types are used for hypothesis generation.

As the data structure defining the nodes of network we use *frame* which is a complex data structure composed of three groups of slots: *Attributes*, *Relations* and *Control*:

object-class name;
Attributes:
Relations:
GE: rule
PO: rule
S: rule

Control:
Kernel-list: list of classes
PO-strategy: list of classes
S-strategy: list of classes
end-object-class

Slots in *Attributes* section are used to record property values of objects. *Relations* section describe the knowledge about GE, PO and S relations in which an object participate by means of *rules* which are attached to the slots of a frame. In this way, rules play the role of procedural component in a semantic network. They define the semantcis of relations in a semantic network. The *Control* section describes a control knowledge by means of *rules* and by three slots: *S-strategy*, *PO-strategy*, and *Kernel-list*. The *S-strategy* slot defines the specialization strategy, that means an ordered list containing the set of all its specialized classes. Similarly, the *PO-strategy* defines an ordered list of its parts. A *Kernel-list* slot specifies a subset composed of those parts of given class which must be instantiated to regard an instance of a given class *fully instantiated*. Otherwise, it is only *partially instantiated*. Each *rule* is composed of two parts:

I Pre-condition:
II Activity
Hypothesis: procedure
Function: procedure
Task: procedure

In the interpretation process, first the *Pre-condition* of each rule is examined and, if it evaluates to *True*, the procedure in the *Hypothesis* field of *Activity* part is executed to generate and insert a hypothesis (i.e. an expected description entity) into the database. A rule which generated a hypothesis is recorded in the *Task list*. After computing a solution for a hypothesis by the reasoning engine, the procedure in the *Task* part is executed again using now the computed solution, allowing for adding facts to, or deleting them from a database. The *Function* field (used only in case of PO and S relations) refers a procedure which converts a description of a general (whole) object into its specialized one (its part). It is used during the search for an object appearance in the model-driven reasoning, or bottom-up instantiation.

The proposed hybrid semantic network, together with the proposed image analysis flow scheme described in chapter 8, provides a general reasoning framework which allows the *integration* of a variety of pieces of knowledge and reasoning methods into one consistent framework.

2. Semantics

Figures 7.1 and 7.2 show two fragments of the map model for situational layer of a PFLM. For simplicity, not all relations between objects in a Domain level are depicted.

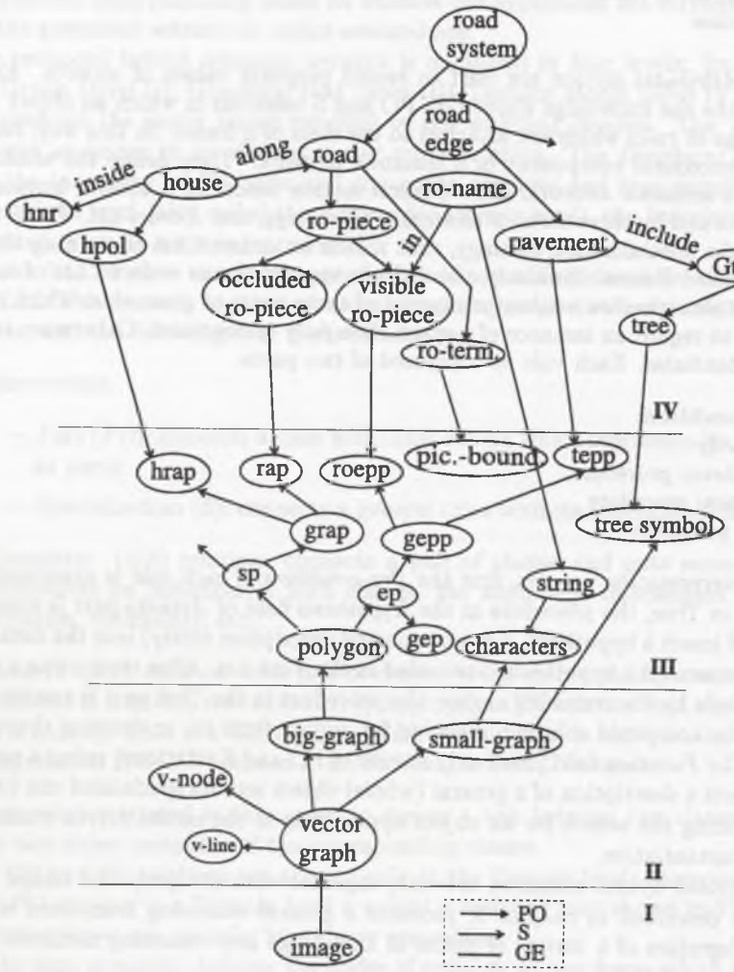


Fig. 7.1. The fragment of the map model for situational layer of a PFLM. For simplicity, not all objects and relations in a Domain level are depicted

Rys. 7.1. Fragment modelu mapy dla warstwy sytuacyjnej Mapy Zasadniczej Kraju. Dla uproszczenia, nie wszystkie obiekty oraz związki poziomu dziedziny przedmiotowej zostały zaznaczone

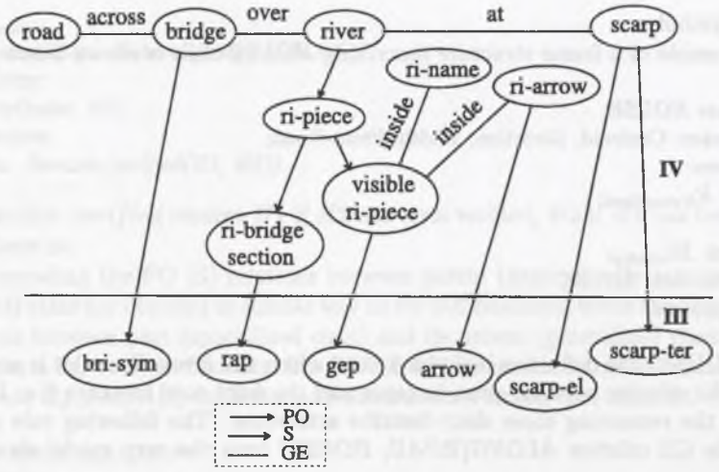


Fig. 7.2. The fragment of the map model (levels III, IV) for situational layer of a PFLM. For simplicity, not all objects and relations in a Domain level are depicted

Rys. 7.2. Fragment modelu mapy dla warstwy sytuacyjnej (poziomy III, IV) Mapy Zasadniczej Kraju. Dla uproszczenia, nie wszystkie obiekty oraz związki poziomu dziedziny przedmiotowej zostały zaznaczone

The Domain level represents such geographic objects like *ROAD* composed of *RO-PIECE* parts, *PAVEMENT*, *HOUSE* (composed of *HPOL* (house polygon), HNR (house number)), *RO-NAME* (road name), *RIVER*, *SCARP*, *BRIDGE*, *TREE* and some others. Some of these objects are specialization of more general ones (for example *ROAD* is a specialization of *ROAD-SYSTEM*), or are further specialized (for example *RO-PIECE* into *OCCLUDED-* and *VISIBLE-RO-PIECE*). Additional to weights, the GE relations are labeled, for example the label *ALONG* denotes the relation between *ROAD* and *HOUSE*. The Graphics/Text level include the *STRING* class composed of *CHARACTERS*, geographic symbols (for example *TREE SYMBOL, BRIDGE SYMBOL*), primitive elements of line objects (for example *SCARP-ELEMENT*), and different specializations of *POLYGON* class: *SP* - short, *EP* - elongated, *GRAP* - general right-angled, *HRAP* - general right angled drawn with thick lines, *GEP* - general elongated, *GEPP* - general elongated with pairs of parallel opposite sides, further specialized into thick *ROEPP* (road elongated parallel polygon) and thin *TEPP* (thin elongated parallel polygon).

The graphic objects *HRAP, RAP, ROEPP, RO-TERM, TEPP, CHARACTERS, TREE-SYMBOL, SCARP-ELEMENT, SCARP-TERM* from the 3-rd level of the map model shown in Fig. 7.1 define the *apearance model* of appropriate geographic objects from the 4-th level.

The nodes of the Vector Graph which is on the 2nd level of hybrid semantic network, corresponds to image line crossings (class *V-NODE*), while the edges to the image lines (class *V-LINE*). Vector Graph is further specialized based on a vector length into *BIG-GRAPH* representing long line elements of graphics and *SMALL-GRAPH* representing

text and symbols.

The example of a frame structure describing *HOUSE* class is shown below:

object-class HOUSE;
Attributes: Centroid, Direction, Width, Front-Road;
Relations:
GE: $R_{HouseRoad}$;
S:
Control: $R1_{control}$
Kernel-list: HOUSE
end-object-class

The *HOUSE* class definition includes 4 slots, where the *Front-Road* slot is used to store the symbolic relation between given instance and the front road instance (i.e. ID number of), while the remaining three slots describe attributes. The following rule $R_{HouseRoad}$ encodes the GE relation *ALONG(ROAD, HOUSE)* from the map model shown in Fig. 7.1:

Pre-condition: *True*
Activity:
Hypothesis: H=$P_{HouseRoad}$
Function:
Task: $Self.Front\text{-}Road = H$

The similar rule ($R_{RoadHouse}$) is stored in *ROAD* class, by which an instance of *ROAD* generates a hypothesis for *HOUSE*. The rule $R_{HouseRoad}$ specifies the following knowledge. Each instance $house_0$ of *HOUSE* class performs reasoning using this rule to detect the *ROAD* instance satisfying the relation *ALONG*, since *Pre-condition* is set to *True*. $P_{HouseRoad}$ represents a procedure which generates a hypothesis for an unknown *ROAD* instance $road_x$. The procedure $P_{HouseRoad}$ uses the properties of $house_0$ to generate the following *ROAD* hypothesis:

$P_{HouseRoad}(house_0)$
$house_0.direction - 20^o \leq direction\ d \leq house_0.direction + 20^o$
$house_0.width \times 0.6 \leq width\ w \leq house_0.width \times 0.8$
centroid c resides within $REGION(house_0.centroid, d, w)$
cc Region is a function that generates an expected location of the target object cc

The values of constants in the hypothesis have been experimentally determined on several samples of real maps. *Self* is a special variable, which is bound to an object instance to which a rule is applied. If the target *ROAD* instance $road_x$ is detected, it is identified by a variable H. The *Task* field indicates that the *Front-Road* slot should be filled with the variable H representing the computed solution, the $road_x$ instance.

The metarule $R1_{control}$ in *Control* section is used to retract an alternative hypothesis *H1* or *H2* about *ROAD* generated by *HOUSE* instance after one of them has been verified:

$R1_{control}$:
Pre-condition: *Not-null(verified(H1,H2))*
Activity:
Hypothesis: *NIL*
Function:
Task: *Remove(verified(H1, H2))*

The function *verified* returns $H1$ if $H2$ has been verified, $H2$ if $H1$ has been verified or *NIL* otherwise.

Rules encoding the PO (S) relations between parent (generalized) class and its part (specialized) class are encoded in similar way as for *GE* relations, while the "opposite"PO (S) relations between part (specialized class) and its parent (generalized class), are not. For example, a PO relation between *RO-PIECE* and its parent *ROAD* is encoded by the following rule $R_{PO}(ROAD)$ for *bottom-up instantiation* stored in a class *RO-PIECE*:

Pre-condition: *True*
Activity:
Hypothesis: *NIL*
Function: *F(ROAD))*
Task: *Make-instance (ROAD,F(ROAD))*

In this case a *Hypothesis* part is equal to *NIL* which means the immediate instantiation of a parent object *ROAD* without hypothesis generation. In general, every rule with hypothesis slot equal to *NIL* is not put into the *Task list*, but is evaluated immediately. Procedure *F(ROAD)* is used to compute a parameter object based on the current values of *RO-PIECE* instance.

Chapter 8

The image analysis flow scheme

The proposed *image analysis flow scheme* is composed of:

- Control mechanism.
- Reasoning method.

The image analysis flow scheme is based on the proposed *intelligent analysis strategy*. The main principle of this strategy is that: *Objects and their relations can be recognized much more successfuly if algorithms know what to look for, and do so in a constrained search*. This strategy uses *contextual knowledge* and *constraints* to formulate and then verify interpretation hypotheses.

The intelligent analysis strategy is accomplished in the proposed framework by having the *initial recognition* of features (*seed objects*) based on a "crude"vectorization and character/symbol separation, followed by progressively accomplished extraction of the remaining layers of geographic objects. Based on a map model, seed objects produce feature hypotheses which drive the search for particular (missing) part of object in a *restricted area*, precisely determined based on properties of the object which generated a hypothesis. If the hypothesized feature cannot be found in the specified area, an alternative detector is called, and if it fails, partial re-runs of segmentation and vectorization modules with tuned parameters are necessary to recover missing object features. This usually happens at locations where text and graphics itersect.

1. The control mechanism

Fig. 8.1 illustrates overall control mechanism of the image analysis scheme. It starts with the *initial recognition* of the *seed object* instances from which the interpretation of the entire map is constructed gradually. After the seed object instances are stored in evidence pool, control is passed to reasoning engine (*RE*) which starts constructing the interpretation based on the information stored in evidence pool. The goal of the analysis (i.e. termination condition) is defined as follows: "detect from the image all possible

object instances which match the map model". *RE* iterates the following *interpretation cycle* (depicted in Fig. 8.1):

- **Hypotheses generation**
- **Compatibility examination**
- **Scenario selection**
- **Scenario verification.**

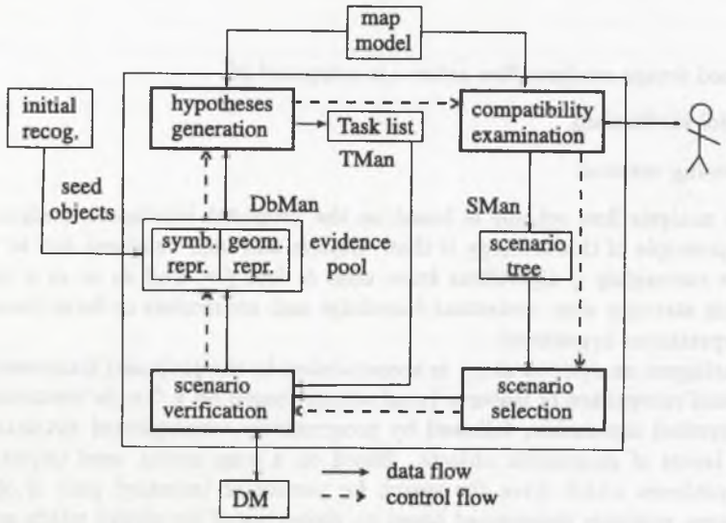


Fig. 8.1. The interpretation cycle. TMan, SMan, DbMan - Task-, Scenario-, Database- Managers

Rys. 8.1. Cykl interpretacji. TMan, SMan, DbMan - moduły zarządzania zadaniami, scenariuszami, bazą danych

The interpretation cycle is terminated when no more hypotheses can be generated, and every hypothesis has been verified or refuted. To avoid infinite generation of hypothesis, we prohibit hypothesis from generating a new hypothesis. At each interpretation cycle, one, the most reliable *scenario* is selected and verified, and evidence pool is modified based on the solution of the selected scenario. The modification of evidence pool enables new rules to be activated and new hypothesis as well as scenarios to be generated.

Many hypotheses are actively generated to complement the insufficient input information. From a computational point of view, generation of such multiple descriptions need large memory space and much computation time. We used several mechanisms to reduce the amount of evidence in the database, for example hypothesis generation by hypothesis is prohibited, verified/refuted hypothesis are immediately removed. The global selection

method in interpretation cycle also helps to increase the efficiency by removing as early as possible conflicting objects. Thus, the computational complexity itself is not great.

To increase the reliability and accuracy of the existing methods in geographic map image analysis, in our control mechanism the hypotheses generated from different sources are accumulated according to an idea taken from a system Sigma [49] - a simple system for interpretation of buildings and roads in aerial images. The proposed control mechanism adapts the Sigma control algorithm for scanned map interpretation, and increases its reliability. Our major modifications are: (1) the more powerful completion of insufficiency of input information through the usage of model relations to control quantitatively each step of a cycle, (2) compatibility examination in each step to detect contextual inconsistences as early as possible, and (3) a global selection method in which the context knowledge controls quantitativly the choice among the hypothesis.

1.1. Hypotheses generation

At the beginning, several important tasks like the bottom-up instantiation process based on PO relations, the retraction of alternative hypotheses, unification of duplicated instances are performed by *DbMan*. Then, for each existing object instance *DbMan* evaluates *Pre-condition* of each rule in *Relations* section of its class. If this evaluation result is *True*, *DbMan* generates a hypothesis based on properties of the object instance and inserts it into evidence pool. It also inserts *Task* part of a rule into the *Task list* (delayed tasks). Only instances are allowed to generate hypotheses about related objects.

1.2. Compatibility examination

The compatibility test is performed for every pair of *pieces of evidence* (object instances or hypotheses) whose geometric descriptions overlap. A pair of such pieces of evidence are regarded as *compatible* if the following conditions are satisfied:

1. their geometric descriptions satisfy some requirements on the size and shape of the intersecting area,
2. their object classes must be compatible (the same or linked by G/S or PO link, or a sequence of them),
3. the intersection of sets of linear inequalities associated with both pieces of evidence, which constrain permissible values of the denoted object attributes must be nonempty.

A group of compatible pieces of evidence are combined into *scenario* which represents a local environment context. A pair of pieces of evidence forms a *2-compatible* scenario if they are compatible. A set of n pieces of evidence is *n-compatible* if all possible *(n-1)* pieces of evidence in the scenario are *(n-1)-compatible*. A partial ordering is defined on scenarios in the following way. One scenario is *less than* other if the set of pieces of evidence participating in it is a subset of the other. The *maximum compatible scenario* are

those scenarios which are not less than any other scenarios. The compatibility relations among pieces of evidence are represented by means of a *scenario tree* in which leaves represent single piece of evidence, and a root node a maximum compatible scenario. The scenario tree is updated after a new piece of evidence is inserted into the database. The space and time complexities of algorithm for tree updating are both of $O(N^2)$ (N being the number of pieces of evidence in a database), but N, usually, does not increase very much during the interpretation process.

The RV relation provides a generally applicable mechanism to detect *contextual inconsistences* during compatibility checking (2-nd step) in the following way. Each time two object instances with overlapping areas are found, that means some sort of GE relation *Rel* between these two object instances exist, their classes are not compatible, and they share reverse relation (the detected relation *Rel* in the image is not specified in the map model), a misclassification has been occured. Such object instances are called *conflicting object instances*. In this case the help of an operator is invoked.

1.3. Scenario selection

First, a set of maximum compatible scenarios from the scenario tree is selected, then for each selected scenario its overall cost is computed, and the scenario with the greatest cost is chosen. The *overall scenario cost* is a sum of cost values associated with pieces of evidence participating in a scenario. Each object instance participating in the scenario is assigned a unit cost, while each hypothesis a weight value assigned to a relation by which the hypothesis has been generated.

1.4. Scenario verification

After constructing a composite hypothesis from those contained in the selected scenario, the data- or model-driven construction of relations through complementary reasoning is performed, depending on the fact whether or not the selected scenario contains a piece of evidence representing an object instance. In this way, the control strategy does not require ad hoc control knowledge about when to activate one analysis or the other. Instead, the top-down analysis is activated dynamically, depending on the contents of the current scenario under analysis.

2. Complementary reasoning

As it was said in chapter 1, the map image analysis process should be based on non-monotonic reasoning scheme. It should actively generate new information (hypotheses) based on the model, to cope with the insufficiency of the input information. The logical framework of the proposed reasoning method, called *complementary reasoning*, is based on a modified Poole's approach ([258]) to non-monotonic reasoning. The Poole's approach

could be suitable, as its goal is to find the set of logical hypothesis which complement the incompleteness of the knowledge, that means to construct explanations of the observations in terms of both the given knowledge, and the found, logical hypotheses.

The complementary reasoning is based on the assumption that the objects share clear relations (GE, PO). During this reasoning the relations from the model are utilized to *complement* the mentioned insufficiency of input information and imperfect segmentation, that means are used to generate new information (hypotheses). Accumulated hypotheses and object instances (evidence) originating from different sources are then used by reasoning engine to *construct relations* in model- or data-driven way. Because of their different semantics, the GE and PO relations are used differently in the complementary reasoning.

The complementary reasoning can be initiated at any time based on the incomplete information thus far obtained, and the accumulation of multiple pieces of evidence from different sources increases the reliability of the analysis. The description of the map can be constructed even if some pieces of evidence are missing. The redundancy of the system complements the insufficiency of the input information.

2.1. Construction of GE relations

In the *data-driven* reasoning a GE relation between given two object instances is constructed as a result of (successful) compatibility examination of the scenario composed of at least two pieces of evidence: one of the two object instances, and the hypothesis about this object instance generated from the model by the second object instance. Fig. 8.2b) ilustrates the data-driven reasoning based on the relation $REL(O_1, O_2)$ from the map model shown in Fig. 8.2a). Suppose the instance $obj1$ of object class O_1 generates a hypothesis $H_{REL}(obj1)$ for object class O_2 based on relation $REL(O_1, O_2)$, and the region associated with $H_{REL}(obj1)$ overlapes with an instance of class O_2, $obj2$. If the two pieces of evidence $H_{REL}(obj1)$ and $obj2$ form 2-compatible scenario, they are unified by RE which reports that result to both $obj1$ and $obj2$. Then, each object instance examines the returned by RE result, and performs the procedure specified in *Task* field of a rule which was used to generate the hypothesis H_{REL}, that means records the symbolic relation REL, retracts the related hypothesis and eventually, generates other hypotheses based on stored rules.

In *model-driven* reasoning the hypotheses generated for a missing object are unified into a composite hypothesis (by *AND*-ing the sets of inequalities associated with each hypothesis), the appropriate detector is called to extract the instance denoted by the composite hypothesis, and if its is successfuly detected, the relations between the new instance and the source instances which generated the hypotheses are established. Fig. 8.2d) ilustrates the model-driven reasoning based on GE relations $REL1(O_1, O_3)$ and $REL2(O_2, O_3)$ from the map model shown in Fig. 8.2c). Suppose the instances $obj1$ and $obj2$ of object class O_1 and O_2, respectively, generate hypotheses $H_{REL1}(obj1)$ and $H_{REL2}(obj2)$ for object instance of class O_3, and the regions associated with $H_{REL1}(obj1)$ and $H_{REL2}(obj2)$ overlap. If the appropriate detector extracts succesfully the instance

denoted by the composite hypothesis $H_{REL1}(obj1) \cap H_{REL2}(obj2)$, they are unified by RE into the instance $obj3$ of class O_3, which reports the new instance to both $obj1$ and $obj2$. Then, similarly as in data-driven reasoning each object instance records the appropriate symbolic relation (*obj1* the relation *REL1*, *obj2* the relation *REL2*), and retracts the related hypothesis. If pieces of evidence in a scenario are not compatible, or the required object instance cannot be found, a failure is reported to all source instances which may in turn activate rules to respond to such failure in the next interpretation cycle.

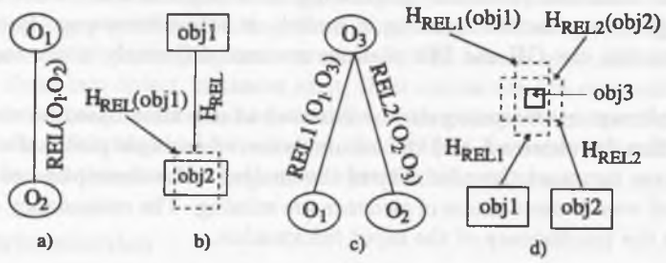


Fig. 8.2. An illustration of complementary reasoning based on GE relations: a), c) fragments of a map model, b) the data-driven reasoning, d) the model-driven reasoning

Rys. 8.2. Ilustracja wnioskowania uzupełniającego opartego na związkach GE (geometrycznych): a), c) fragment modelu mapy, b) wnioskowanie oparte na danych, d) wnioskowanie oparte na modelu

2.2. Construction of PO relations

In the *data-driven* reasoning, a PO relation between an object instance and its parent object is constructed by directly instantiating a parent object through a PO relation from a model (without generating a hypothesis). Only a fully instantiated object instance can instantiate its parent object through a PO relation.

Construction of PO relations in *model-driven* reasoning is performed similarly as for GE relations, irrespectively of the state of an instance: partially instantiated or fully instantiated.

Each instance of a leaf object in the PO hierarchy instantiates its parent object and constructs a partially instantiated PO hierarchy independently of the others. This causes multiple descriptions of the same parent object, which in turn calls *unification* of such multiple instances into one (e.g. see Fig. 8.3 for ilustrative example). An instance $obj1$ (class O_1) instantiates its parent object $obj31$ (class O_3), which generates a hypothesis $H^{PO}(obj31)$ for its missing part object of class O_2 (Fig. 8.3b)). Suppose an instance $obj21$ (class O_2), corresponding to the missing part has already been detected. Since $obj21$ is an instance of O_2, it has also instantiated its parent object $obj32$ (class O_3). After compatibility checking between $H^{PO}(obj31)$ and $obj21$, they are unified into $obj2$ and PO relations between $obj2$ and $obj31$, as well as between $obj2$ and $obj32$ are established (Fig.8.3c)), which causes $obj2$ to have two parent instances: $obj31$ and $obj32$, at the same

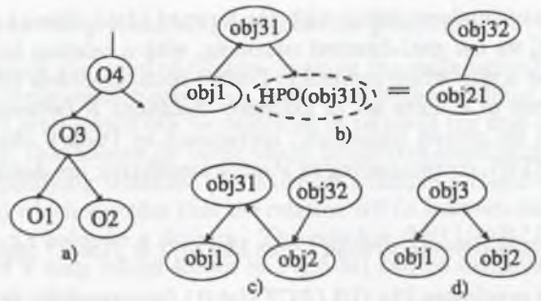


Fig. 8.3. Reasoning based on PO relations: a) PO hierarchy, b) data-driven instantiation and model-driven hypothesis generation, c) an object *obj2* has two parents, d) unified PO hierarchy

Rys. 8.3. Wnioskowanie oparte na związkach PO (być częścią): a) hierarchia związków PO, b) oparte na danych ukonkretnienie oraz generacja hipotez na podstawie modelu, c) obiekt *ob2* posiada dwóch rodziców, d) zunifikowana hierarchia związków PO

time. After compatibility checks for $obj31$ and $obj32$ by RE, they are unified into a new instance $obj3$ of class O_3 (see Fig. 8.3d)).

2.3. Formalization of complementary reasoning

2.3.1. *Modifications of Poole's non-monotonic reasoning*

The major purpose of this section is to present the logical foundation of the proposed complementary reasoning in terms of the first-order predicate calculus. Formal logic allows us to precisely describe the knowledge and reasoning mechanisms required in image understanding.

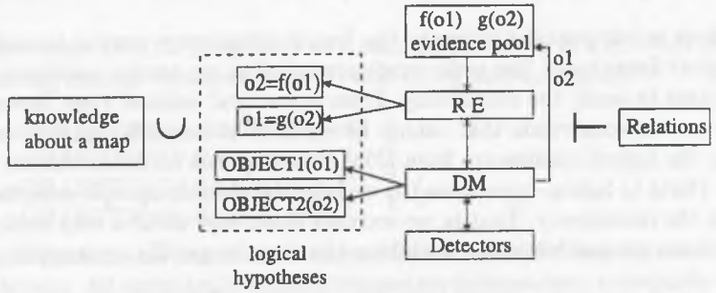


Fig. 8.4. The logical framework of the complementary reasoning

Rys. 8.4. Struktura logiczna wnioskowania uzupełniającego

As it was mentioned, a logical framework of the complementary reasoning in MAPIN (shown in Fig. 8.4) is based on a modified Poole's nonmonotonic reasoning method ([258]). The Poole's approach has been previously used in the mentioned system Sigma

([221]) for aerial image understanding with the forward (data-driven) reasoning method. As opposed to [221] we use goal-directed reasoning, with a relation being a goal.

Knowledge about a map which represents Poole's *facts*, describes the knowledge about the domain (in terms of a finite set of axioms). *Relations R* between object instances in a map image to be proved (construct) correspond to Poole's *observations*. *Logical hypotheses H* in MAPIN, corresponding to Poole's *hypotheses*, are described by two types of formulas:

1. equality relations between ground terms like $obj2 = f(obj1)$ (generated by RE)
2. unary ground predicates like $OBJECT1(obj1)$ (generated by DM).

Since the given knowledge is incomplete, the relations cannot be derived (proved) based on the knowledge about a map alone. Thus, it is searched for a set of logical hypotheses which complement the incompleteness of the knowledge, so as the following conditions are satisfied:

- knowledge about a map $\cup$ logical hypotheses $\vdash$ relations
- knowledge about a map $\cup$ logical hypotheses are compatible.

The complementary reasoning strategy attempts to accumulate compatible sets of pieces of evidence as explanations for which the relations are logical consequences. That is, the explanations of the relations are constructed in terms of both the given knowledge, and the selected logical hypotheses.

This reasoning scheme is different from ordinary deductive reasoning ([129, 213, 130]), in which given axioms and inference rules, the set of derivable theorems is fixed. Ordinary deductive reasoning does not generate new information, (i.e. logical hypotheses) to explain the observations. Two major modifications in the Poole's method [258] have been introduce in complementary reasoning.

1. As there is not possible to verify the logical consistency among an arbitrary set of logical formulas of first order predicate calculus, we cannot construct a general algorithm to verify the consistency. From a practical point of view, however, there exist various constraints that cannot be violated in a specific application domain. Thus, the logical consistency from [258], meaning that no contradiction is derived from {facts $\cup$ logical hypotheses} is replaced by *domain-specific procedures* to examine the consistency. That is, we consider *knowledge about a map model* $\cup$ *logical hypotheses* compatible unless it violates the domain-specific constraints.
2. Facts are *dynamically* created during the reasoning. RE examines the compatibility among finite set of ground terms (i.e. constants and instantiated functions like a and $f(a)$) to find a group of ground terms denoting the same object and generates *new facts* describing equality relations between such ground terms like $obj2 = f(obj1)$. Also, DM generates *new constants* $obj1$ and *facts* about their object classes like $OBJECT1(obj1)$.

2.3.2. Complementary reasoning as theorem proving

In this section we show the proof of *relation* (making the explanation) in first-order predicate calculus in complementary reasoning, performed in the modified Poole's framework. The knowledge of a map model can be described in terms of the first-order predicate calculus with equality, that means by using unary predicates specifying object classes and n-ary predicates specifying relations. Consider for example a fragment of a map model shown in Fig. 8.5a) which specifies that the relation $REL1$ between the two object classes $Object1$ and $Object2$ holds in a domain. The relation $REL1(OBJECT1, OBJECT2)$ from a fragment of a map model shown in Fig.8.5a) can be described by means of the following logical formulas:

$$\forall x[OBJECT1(x) \rightarrow \exists y\{OBJECT2(y) \wedge REL1(x,y)\}] \tag{8.1}$$

$$\forall x[OBJECT2(x) \rightarrow \exists y\{OBJECT1(y) \wedge REL1(y,x)\}] \tag{8.2}$$

which specify that for every $OBJECT1$ instance, there exists an $OBJECT2$ instance which satisfies the relation $REL1$ to it and vice versa. The detailed knowledge about each object and relation from a map model can also be precisely described by the appropriate axioms, for example the following 2 axioms specify some properties of $OBJECT2$ and $REL1$:

$$\begin{aligned}\forall x[OBJECT2(x) \rightarrow\ & GREATER(width(x), MinWidth)\\ & \wedge LESS(width(x), MaxWidth))]\end{aligned} \tag{8.3}$$

$$\begin{aligned}\forall x,y[REL1(x,y) \rightarrow\ & PARALLEL(direction(x), direction(y))\\ & \wedge DISTANCE(centroid(x), centroid(y), MinDist)]\end{aligned} \tag{8.4}$$

where *width, direction, centroid* denote functions to compute attributes of objects, and their semantics are defined by attached procedures. The axioms like those above are stored in object classes based on which hypotheses and constraints during reasoning are derived.

The knowledge about a (fragment) of a map model shown in Fig. 8.5a) is composed of the following axioms (obtained from (1) and (2) by skolemization):

$$S = \left\{ \begin{array}{l} \neg OBJECT2(x) \vee OBJECT1(g(x)),\\ \neg OBJECT2(x) \vee REL1(g(x), x),\\ \neg OBJECT1(x) \vee OBJECT2(f(x)),\\ \neg OBJECT1(x) \vee REL1(x, f(x)), \end{array} \right\}$$

Predicate symbols are described in capital letters, variables and function symbols in lowercase letters. All variables in different clauses are independent, universally quantified, and all clauses are logically connected by $\wedge$ (depicted as ','). Functions $f(x), g(x)$ are Skolem functions. $f(x)$ denotes the certain object which satisfies the relation $REL1$ to object x (clause 4) (i.e. $REL1(x, f(x))$ is $True$ if $OBJECT1(x)$ holds), and the meaning of the function is to map object x to such an object. In order to avoid infinite generation of instantiated Skolem functions (..g(f(a))..) we prohibit a hypothesis from generating a new hypothesis.

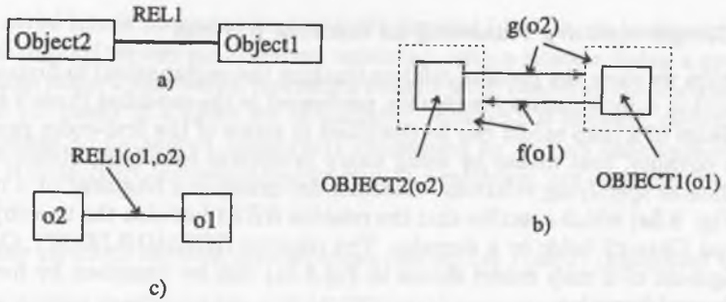


Fig. 8.5. a) The fragment of the map model, b) the fragment of evidence pool during data-driven reasoning based on relation $REL1$, c) the constructed relation

Rys. 8.5. a) Fragment modelu mapy, b) fragment bazy faktów podczas opartego na danych wnioskowania na podstawie związku $REL1$, c) skonstruowany związek

Suppose, that $OBJECT1$ instance $o1$ and $OBJECT2$ instance $o2$ are recognized (that means, the predicates $OBJECT1(o1)$ and $OBJECT2(o2)$ are both $True$). DM transforms the required image features extracted by proper detectors of $OBJECT2$ and $OBJECT1$ to constants representing object instances, say $o1$ and $o2$, based on the knowledge from the map model. Then, DM inserts constants $o1$ and $o2$ into the evidence pool and generates a logical hypothesis $OBJECT1(o1)$ and $OBJECT2(o2)$. When $OBJECT2(o2)$ as well as $OBJECT1(o1)$ are given as facts, they generate the hypotheses $f(o1)$ and $g(o2)$. The evidence pool stores all the ground terms generated so far: constants generated by DM and ground functions instantiated by RE.

The following describes the two steps in reasoning according to the modified Poole's method:

1. Compatibility checking: $(f(o1) = o2)$ or $(g(o2) = o1)$.

2. Proof: $S \cup H \vdash REL1(o1, o2)$,
   where:
   $$H = \left\{ \begin{array}{l} OBJECT1(o1) \\ OBJECT2(o2) \\ f(o1) = o2, \\ g(o2) = o1 \end{array} \right\}$$
   is the set of logical hypotheses.

In the first step, the compatibility between $o2$ and $f(o1)$ (or $o1$ and $g(o2)$) is checked by RE. Suppose that $o1 = g(o2)$ holds, which means that RE regards $o1$ and $g(o2)$ as denoting the same object (they augment the set H of logical hypotheses). Thus, compatibilty examination by RE introduces the equality relations between ground terms. RE reports this fact to object instance $o2$, which derives $REL1(o1, o2)$ by an ordinary proving procedure (shown in see Fig. 8.6), that means using resolution principle with paramodulation. Then the object instance $o2$ reports $REL(o1, o2)$ to $o1$, which in turn derives $o2 = f(o1)$ based on its own local knowledge in the similar way. Note that even

if we first verify $o2 = f(o1)$ rather than $o1 = g(o2)$, we can derive the same conclusions. The reasoning can be initiated from either $o1$ or $o2$.

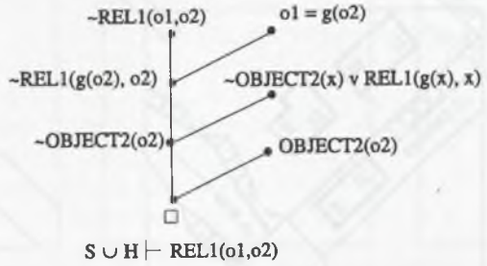


Fig. 8.6. Linear resolution proof with paramodulation for REL1(o1,o2)

Rys. 8.6. Wywód za pomocą rezolucji liniowej z paramodulacją związku REL1(o1,o2)

The formally stated reasoning process is exactly the same as the complementary reasoning described in the previous section. The object instance corresponds to a constant in logic, which generates a hypothesis for its related object by applying a function to itself. The generated hypothesis coresponds to an instantiated Skolem function (i.e. $f(o1)$) in the logic, with which a set of constraints is associated (for example $REL1(o1, f(o1))$, specifying that $f(o1)$ must satisfy relation $REL1$ to $o1$).

The reasoning for the hypothesis generation is local. Given a constant, the reasoning can be performed independently without using the other constants. This corresponds to an idea of considering object instances as active reasoning agents; they can generate hypothese based on their own knowledge.

3. Examples of constructing interpretations

This section illustrates by examples the complementary reasoning performed in situational layer of a PFLM. Fig. 8.7 shows the fragment of such map with a road and pavement containing a group of trees, 3 houses, a river with two scarps and a bridge located at a crossing of a road and a river. Fig. 8.8a) and b) show the Big and Small Graph representations of the image shown in Fig. 8.7.

The interpretation starts with the initial recognition process with *HPOL*, *RO-PIECE*, *PAVEMENT* and *RI-ARROW* selected as seed object classes. Based on a model from Figures 7.1 and 7.2 from chapter 7, *DM* generates the appearance of the seed objects: *HRAP* for *HPOL*, *ROEPP* for *RO-PIECE*, *TEPP* for *PAVEMENT*, and *ARROW* for *RI-ARROW*.

The instance *Arrow1* of the *ARROW*, detected by the *arrow* detector (algorithm [335]) at the segmented solid region layer instantiates the *RI-ARROW* class, and the generated instance *Ri-arrow1* is inserted into evidence pool. The remaining seed object instances are detected by the *polygon* detector which extracts all polygons in Big Graph

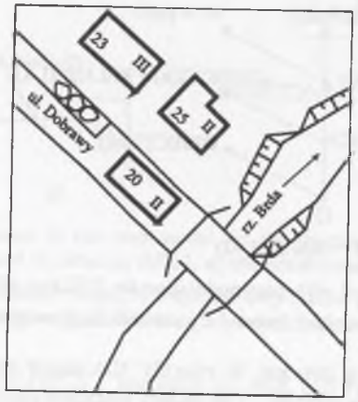


Fig. 8.7. The fragment of situational layer of a PFLM

Rys. 8.7. Fragment warstwy sytuacyjnej Mapy Zasadniczej Kraju

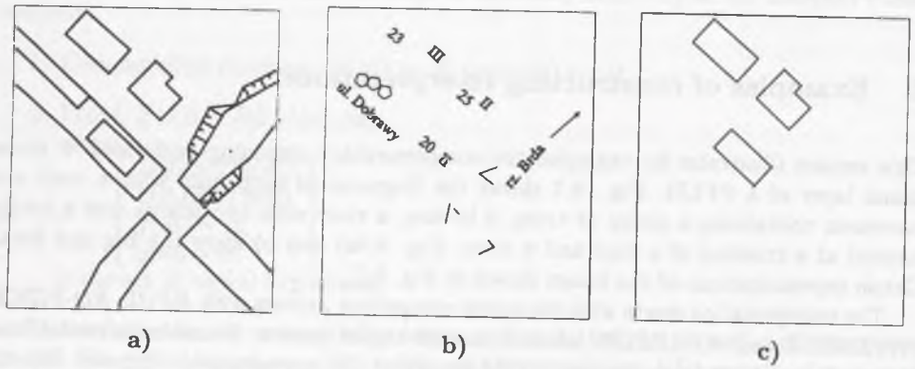


Fig. 8.8. a) Big Graph, b) Small Graph of the map shown in Fig. 8.7, c) HRAP polygons detected by the polygon detector

Rys. 8.8. a) Duży graf, b) mały graf dla mapy pokazanej na Rys. 8.7, c) znalezione wielokąty HRAP

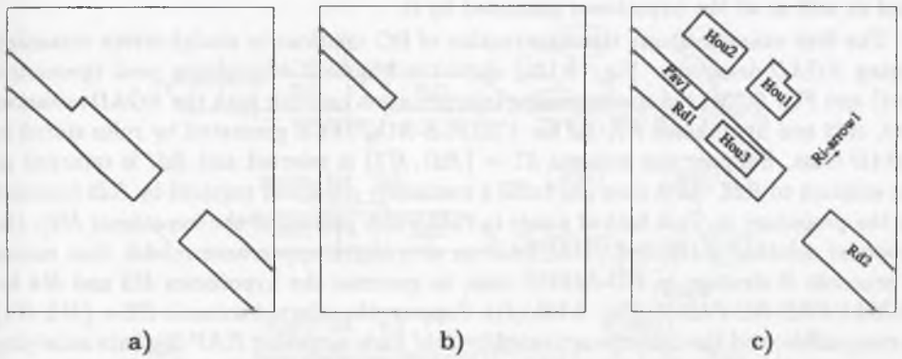


Fig. 8.9. Polygon classification during initial recognition of map shown in Fig. 8.7: a) ROEPP polygons, b) TEPP polygon detected by the polygon detector, c) the geometric part of evidence pool after initial recognition

Rys. 8.9. Wynik klasyfikacji wielokątów podczas wstępnego rozpoznawania fragmentu mapy z Rys. 8.7: a) wielokąty ROEPP, b) wielokąty TEPP, c) część geometryczna bazy faktów po wstępnym rozpoznawaniu

by finding minimum length cycles, and classifies them according to the 3-rd level of the map model shown in Fig. 7.1, based on such criteria like S/L^2 (S, L - polygon area and perimeter), l/d (l, d polygon axis length and breath), parallelity of opposite polygon sides. Fig. 8.8c) illustrates 3 extracted *HRAP* instances *Hrap1, Hrap2, Hrap3*, Fig. 8.9a) - 2 instances *Roepp1, Roepp2* of *ROEPP*, and Fig. 8.9b) - 1 instance *Tepp1* of *TEPP*. Based on the 3 detected *HRAP* instances and 1 *TEPP* instance, *DM* generates 3 *HPOL* instances *Hpol1, Hpol2, Hpol3*, 1 instance *Pav1* of *PAVEMENT*, and inserts them into evidence pool. Based on 2 *ROEPP* instances *Visible-ro-piece1, Visible-ro-piece2* instances of *VISIBLE-RO-PIECE* are generated, and the instantiation rules are applied at higher level in G/S hierarchy resulting in 2 instances *Ro-piece1, Ro-piece2* of *RO-PIECE* which are inserted into evidence pool. The *Ro-piece1, Ro-piece2, Hpol1, Hpol2, Hpol3* instances in evidence pool immediately generate their parent instances (according to the model shown in Fig. 7.1: *Rd1, Rd2* (*ROAD* class) and *Hou1, Hou2, Hou3* (*HOUSE* class), which results in establishment of PO relations between the instantiated parents objects and their parts. Fig. 8.9c) shows the detected *Rd1, Rd2, Hou1, Hou2, Hou3, Ri-arrow1* instances.

This is the end of initial recognition, the scenario tree is constructed, and the iteration of the interpretation cycle is performed until no more hypotheses can be generated, and every hypothesis has been verified, or refuted.

As a result of compatibility analysis of the scenario $S1 = \{Rd1, Hou3\}$ (Fig. 8.9c)), the 2 instances *Hou3* and *Rd1* are regarded as conflicting objects as their object classes are not compatible and the RV relation *INSIDE* between these two instances exist. As a result, operator assistance is invoked, and the *Hou3* instance is removed from evidence

pool as well as all the hypotheses generated by it.

The first example shows the construction of PO relations in model-driven reasoning during *ROAD* detection. Fig. 8.10a) show the fragment of evidence pool (geometric part) and Fig. 8.10e) the corresponding interpretation network with the *ROAD* instances *Rd1, Rd2* and hypotheses *H1, H2* for *VISIBLE-RO-PIECE* generated by rules stored in *ROAD* class. Suppose the scenario $S1 = \{Rd1, H2\}$ is selected and *Rd1* is returned as the solution to *Rd2*. As it does not fulfill a continuity condition required by *Rd2* (checked by the procedure in *Task* field of a rule in *Rd2* which generated the hypothesis $H2$), the proposed solution is rejected. *DM* finds an alternative appearance model, that means it proceeds *S* strategy in *RO-PIECE* class, to generate the hypotheses $H3$ and $H4$ for *OCCLUDED-RO-PIECE* (Fig. 8.10b),f)). Suppose the selected scenario $S2 = \{H3, H4\}$ is compatible, and the detector activated by *DM* finds a missing *RAP* instance satisfying a composite hypothesis constructed from *S2* (after closing the open *RAP* polygon between *Roepp1* and *Roepp2*, which is performed by the *ReSe* module). Based on *S*-type relations *DM* instantiates *OCCLUDED-RO-PIECE*, and then *RO-PIECE* into instance *RoP3* (Fig. 8.10b),f)). Since *RoP3* is connected to *Rd1*, the PO relation between them is constructed, and the verified hypothesis $H3$ is removed. In similar way the PO relation between *Rd2* and *RoP3* is constructed. Since *RoP3* has now two parents, *Rd1* and *Rd2*, these are merged into a new instance *Rd3* of *ROAD* (Fig. 8.10g)). A rule in *ROAD* is applied in *Rd3* to generate new hypotheses $H5$ and $H6$ for neighboring road pieces (Fig. 8.10c),g)). An instance of *ROEPP*, as well as of *RAP* cannot be extracted as the hypotheses are out of the picture boundary, and they are refuted. In the next interpretation cycle, *Rd3* generates a hypotheses about *RO-TERM*, which are satisfied (*RoT1, RoT2* instances). Fig. 8.10d) shows the extracted *ROAD* instance *Rd3*, while Fig. 8.10h) the constructed PO relations.

The next example ilustrates a data-driven construction of GE relations. Consider the fragment of evidence pool whose geometric description is shown in Fig. 8.11a). It consists of *ROAD* instance *Rd3* and *ROAD* hypotheses *H1, H2* generated by *Hou1*. Suppose the scenario $S1 = \{Rd3, H1\}$ has been selected. It passes successfully the compatibility analysis (regions representing $H1$ and *Rd3* intersect, their classes are compatible, and all the constraints: orientation, width associated with both *Rd3* and $H1$ are satisfied). Consequently, *Rd3* is returned as a solution to *Hou1*, which establish the relation *ALONG* between *HOUSE* instance *Hou1* and *ROAD* instance *Rd3* (Fig. 8.11c)).

RIVER instance *Ri3* is constructed in model-driven analysis from its 3 parts,where *RiP1, RiP2* are the *VISIBLE-RI-PIECE* instances (generated from 2 instances of *GEP*), and *RiP3* is the instance of *OCCLUDED-RI-PIECE* (shown in Fig. 8.12a)).

Fig. 8.12 illustrates the model-driven reasoning based on GE relations for detection the missing instance of *BRIDGE*. In the model shown in Fig. 7.2, *ROAD* and *RIVER* instances can generate *BRIDGE* hypothesis only after its corresponding *OCCLUDED-RO-PIECE* and *OCCLUDED-RI-PIECE* instances are detected. When the scenario $S2 = \{H1, H2\}$ is selected (Fig. 8.12b)), the composite hypothesis is constructed from $S2$, and the detector of line objects ([108]) is activated to extract *BRIDGE-SYMBOL* instance. The bridge intersects with a portion of a scarp, but due to special post-vectorization used

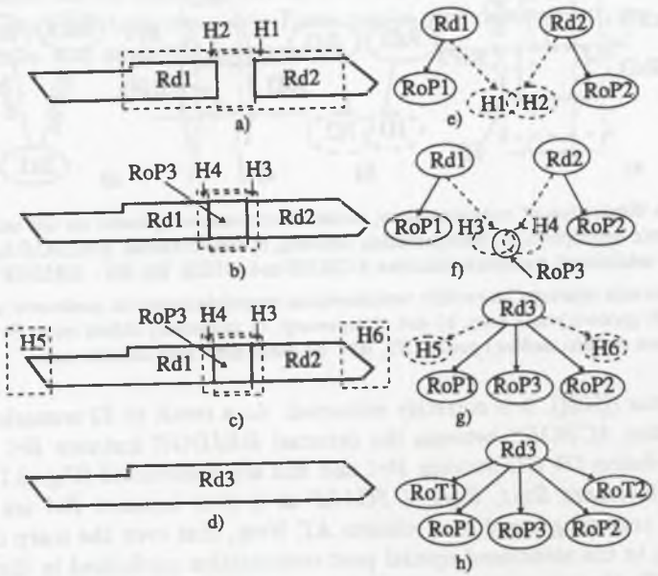


Fig. 8.10. The illustration of complementary, model-driven reasoning based on GE relations during extraction of *ROAD* instance. a), b), c) - geometric descriptions, e), f), g), h) corresponding interpretation networks, d) the extracted road instance

Rys. 8.10. Ilustracja wnioskowania uzupełniającego opartego na modelu na podstawie związku GE podczas ekstrakcji obiektu *droga*: a), b), c) - części geometryczne bazy, e), f), g) h) odpowiadające im sieci interpretacji, d) znaleziony obiekt *droga*

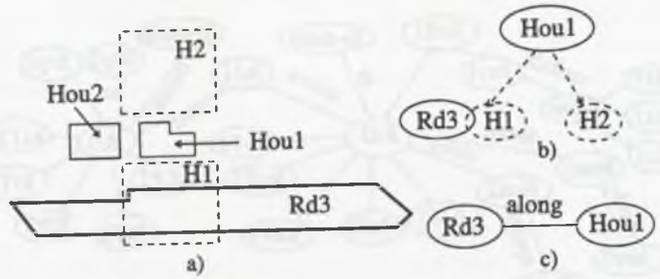


Fig. 8.11. The illustration of complementary, data-driven reasoning based on GE relation: a) geometric description, b) interpretation network, c) the established geometric relation *ALONG*

Rys. 8.11. Ilustracja opartego na danych wnioskowania uzupełniającego na podstawie związku GE: a) część geometryczna bazy, b) sieć interpretacji, c) skonstruowany związek geometryczny *wzdłuż*

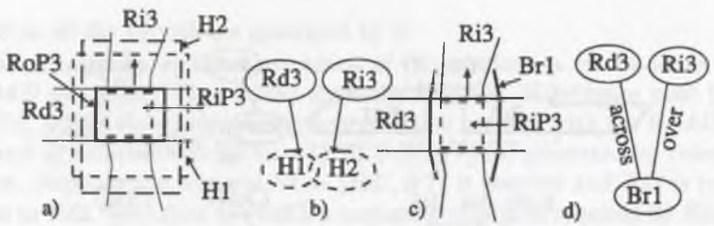


Fig. 8.12. The illustration of complementary, model-driven reasoning based on GE relations: a) geometric description, b) interpretation network, c) the extracted *BRIDGE* instance *Br1*, d) the established geometric relations *ACROSS* and *OVER*. H1, H2 - *BRIDGE* hypotheses

Rys. 8.12. Ilustracja opartego na modelu wnioskowania uzupełniającego na podstawie związku GE: a) część geometryczna bazy, b) sieć interpretacji, c) znaleziony obiekt most *Br*1, d) skonstruowane związki *wzdłuż* i *ponad*. H1, H2 - hipotezy dotyczące obiektu most

by this detector ([108]), it is correctly extracted. As a result of $S2$ scenario verification, the GE relation *ACROSS* between the detected *BRIDGE* instance *Br1* and *Rd*3, as well as GE relation *OVER* between *Br*1 and *Ri*3 are constructed (Fig. 8.12d)).

The two instances *Scp1*, *Scp2* of *SCARP* at a river instance *Ri3* are detected by model-driven reasoning based on a relation *AT*. Note, that even the scarp overlaps with a bridge, due to the mentioned special post-vectorization performed in the line objects detector ([108]), the two scarps are correctly extracted. As a result, the two relations *AT* between *Ri3* and *Scp1* and *Scp2* are constructed.

The 3 instances *Tr1*, *Tr2*, *Tr3* of *TREE* are extracted by model-driven reasoning based on a GE relation *INCLUDE* between *PAVEMENT* and *GTREE*.

RO-NAME, *RI-NAME* and *HNR* instances are extracted by model-driven reasoning which results in construction of appropriate GE relations (Fig. 8.13).

The analysis terminates when all the hypotheses created have been verified or refuted. Fig. 8.13 shows the final interpretation network for the map fragment shown in Fig. 8.13.

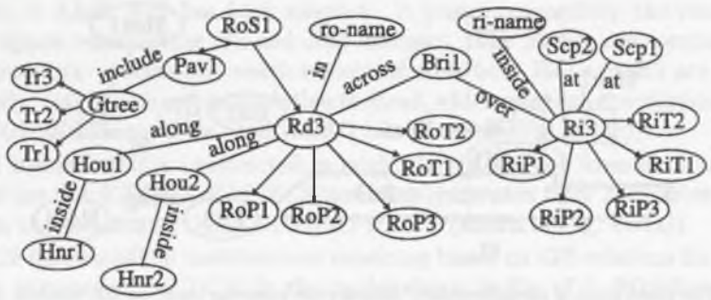


Fig. 8.13. Final interpretation network of Fig. 8.7

Rys. 8.13. Ostateczna sieć interpretacji dla Rys. 8.7

Summing up, all *TREE* as well as *SCARPS* and *BRIDGE* instances were detected by model-driven reasoning. One polygon (*OCCLUDED-RO-PIECE* instance) has been

detected in model-driven reasoning during the analysis process. One conflicting object (*HOUSE* instance) has been removed. These results well demonstrate the effectiveness of combined data- and model-driven reasoning.

Chapter 9

Classifier based on relational matching

1. The overview of the method

Cartographers use predefined symbols, as well as text to convey meanings associated with logical structures represented in maps. Recognition of *geographic symbols* and *text* is, therefore, an important aspect of any map analysis system ([320]). Fig. 9.1 shows some examples of such geographic symbols from 4 layers of Polish Fundamental Land Map (PFLM) ([124]). They are used to symbolize many different objects in a domain.

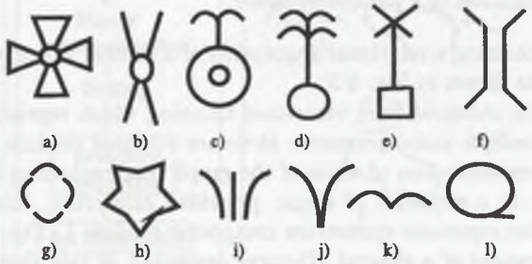


Fig. 9.1. Some examples of geographic symbols from PFLM: a) cross, b) light, c) well, d) fountain, e) windmill, f) bridge, g) leafy tree, h) coniferous tree, i) grass, j) gate, k) green fence, l) bush

Rys. 9.1. Przykłady symboli geograficznych z Zasadniczej Mapy Kraju: a) krzyż, b) latarnia, c) studnia, d) fontanna, e) młyn, f) most, g) drzewo liściaste, h) drzewo iglaste, i) trawa, j) brama, k) żywopłot, l) krzew

This chapter presents a structural method for recognition of *geographic symbols* and *text* (published in [314, 317]), which is based on the *relational model of a general 2D-curve.*

The symbols in the image are extracted by the following sequence of steps (as described in chapter 6): (3,4)-DT thinning ([23]), line following, segmentation into short and long lines, and vectorized thinning ([315]) of short elements which results in a vectorized skeleton of a symbol. The vectorized thinning algorithm decomposes image into regular parts *(lines)*, and singular parts (*multiple points*) representing branchings, crossings and end points on a 2D-curve. The output from the vectorized thinning algorithm

is a piecewise linear curve expressed in terms of the undirected graph $G = (J, F)$, where J is a set of nodes, and F is the set of edges. Nodes of a graph represent multiple points, while edges correspond to simple or closed curves.

The proposed, **relational model** of geographic symbols and text is a combination of two types of *discrete, global, qualitative features*:

- *Quasi-topological features* (convexity, concavity, loop) represented in a model by: *primitive sequences* and a list of their *connections*. These global features have been derived using systematic rules from local features: *primitives* and their binary *concatenations*.
- *A geometric structure* of each multiple point (branch, crossing).

Such qualitative and global structure is robust against many deformations. The above qualitative features composing a relational model are regarded as the dominant information, whereas *quantitative* and *continuous features* (geometric parameters such as size and position) are used as secondary information attached to structural components.

Finding a match between a model and an unknown 2D-curve is performed through the devised *hybrid procedure* consisting of:

- *Structural matching* algorithm
- *Distance calculation* in a parameter space.

A scheme for obtaining a relational description of a general 2D-curve according to the presented method is shown in Fig. 9.2.

Nodes of a graph obtained from vectorized thinning which represent multiple points are processed in *multiple point geometric structure analysis* module (module 2), while quasi-topological representation of edges of the graph (corresponding to simple or closed curves) is created in a sequence of steps: *primitive extraction, binary concatenation, linking* and *primitive sequences connection* composing module 1 (Fig. 9.2).

The relational model of a general 2D-curve presented in this chapter, is based on a modified, algebraic structure of a curve from [251], proposed for representation of Japan characters. We have modified procedure for decomposition of a multiple point on a curve, which resulted in a different to [251] geometric representation of a multiple point, more suitable for geographic symbols. The notion of a *stroke* from [251] has been replaced by a *set of pairs of primitive sequences* converging at a given multiple point. We have modified the matching criteria from ([251]) according to the introduced modifications in a structural description.

The presented method is translation and rotation invariant. Rotation invariance is achieved by a preliminary computation of the shape's orientation and obtaining symbol or string of characters in a reference position. The computation of the orientation can be easily done, because symbols as well as text are drawn parallel or perpendicular to the axis of the geographic object they symbolized, and these axis are known before recognition due to the analysis strategy described in chapter 8.

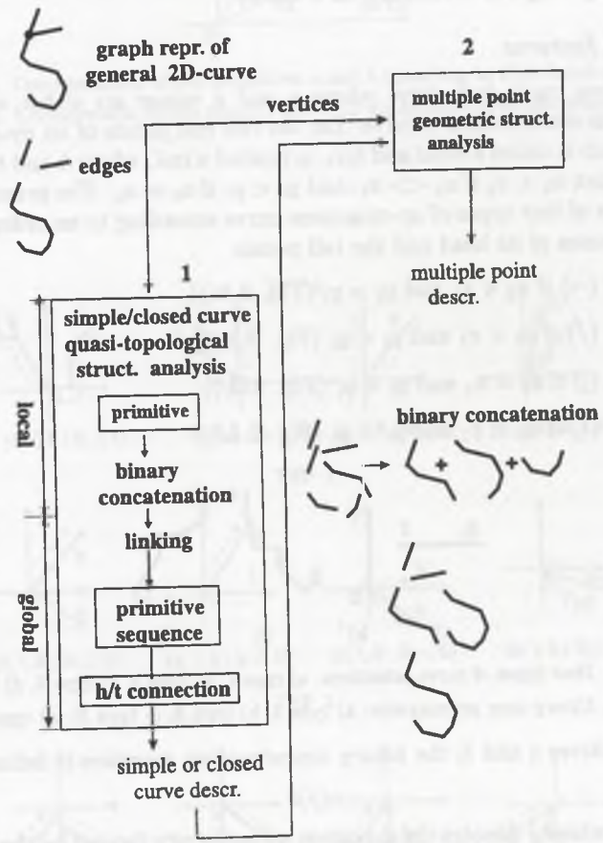


Fig. 9.2. Scheme of the analysis of a structure of a general 2D-curve in obtaining its relational description

Rys. 9.2. Schemat analizy struktury krzywej 2D podczas tworzenia jej opisu relacyjnego

2. Relational representation of a general 2D-curve

A structure of each general 2D-curve, is described by quasi-topological structure of each simple or closed curve, and a geometric structure of each multiple point, as mentioned in first section.

2.1. Quasi-topological structure of a curve

2.1.1. Local features

The *xy-monotone* curve is a curve whose x and y values are either nonincreasing or nondecreasing as one traverses a curve. Let the two end points of an xy-monotone curve be $h(x_0, y_0)$ which is called a *head* and $t(x_1, y_1)$ called a *tail*, where h and t are determined in such a way that $x_0 < x_1$ if $x_0 <> x_1$, and $y_0 < y_1$ if $x_0 = x_1$. The *primitive* of a curve is defined as one of four types of xy-monotone curve according to an order relation of the x and y coordinates of its head and the tail points:

(1) denoted by $(-)$ if $x_0 < x_1$ and $y_0 = y_1$ (Fig. 9.3a)),

(2) denoted by $(/)$ if $x_0 < x_1$ and $y_0 < y_1$ (Fig. 9.3c)),

(3) denoted by $(|)$ if $x_0 = x_1$ and $y_0 < y_1$ (Fig. 9.3b)),

(4) denoted by $(\backslash)$ if $x_0 < x_1$ and $y_0 > y_1$ (Fig. 9.3d)).

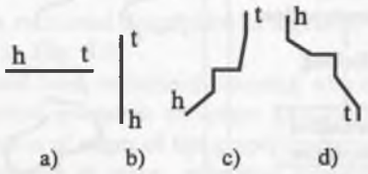


Fig. 9.3. Four types of curve primitives. a) type 1, b) type 3, c) type 2, d) type 4

Rys. 9.3. Cztery typy prymitywów: a) typu 1, b) typu 2, c) typu 3), d) typu 4

For two primitives a and b, the *binary concatenation operation* is defined as:

$$a(j)b$$

for $j = 0, 1, 2, 3$, where j denotes the *direction of convexity* formed by the concatenation of two primitives which could be *downward*, *leftward*, *upward* and *rightward* (e.g. see Fig. 9.5 for some examples). Two primitives a and b are concatenated in such a way that a pair of vectors: (PP_a, PP_b) constitute a right-hand system as shown in Fig. 9.4. Table 9.1 gives a definition of these four types of concatenation.

A symbol $[prm(t_1), p_1; prm(t_2), p_2]$ in Table 9.1 denotes the fact that two primitives t_1 and t_2, one of type $prm(t_1)$ and the second of type $prm(t_2)$ are concatenated at p_1 of the 1-st primitive and p_2 of the 2-nd primitive $prm(t_1), prm(t_2) \in \{-, |, \backslash, /\}$, $p_1, p_2 \in \{h, t\}$, where h and t denote a head and a tail of a primitive. Some examples of a concatenation operation are illustrated in Fig. 9.5.

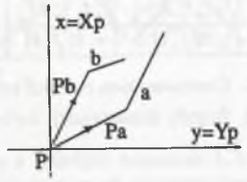


Fig. 9.4. Concatenation of two primitives a and b according to right-hand system

Rys. 9.4. Konkatenacja dwóch prymitywów a i b wg systemu prawoskrętnego

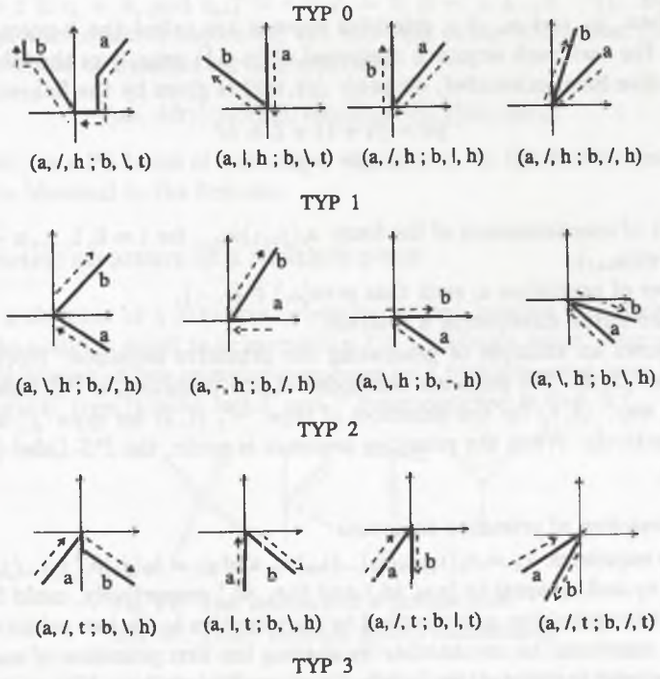


Fig. 9.5. Illustration of the four types of binary concatenation: a) downward, b) leftward, c) upward, d) rightward

Rys. 9.5. Ilustracja czterech typów binarnej konkatenacji: a) w dół, b) w lewo, c) do góry, d) w prawo

downward convex	0	$[/,h;\backslash,t]$	$[\vert,h;\backslash,t]$	$[/,h;\vert,h]$	$[/,h;/,h]$
leftward convex	1	$[\backslash,h;/h]$	$[-,h;/,h]$	$[\backslash,h;-,h]$	$[\backslash,h;\backslash,h]$
upward convex	2	$[/,t;\backslash,h]$	$[\vert,t;\backslash,h]$	$[/,t;\vert,t]$	$[/,t;/,t]$
rightward convex	3	$[\backslash,t;/,t]$	$[-,t;/,t]$	$[\backslash,t;-,t]$	$[\backslash,t;\backslash,t]$

Fig. Tablica 9.1. Concatenation rules of two primitives

Rys. Tablica 9.1. Reguły konkatenacji dwóch prymitywów

2.1.2. *Global features*

2.1.2.1. *Primitive sequences of curves*

A simple 2D-curve is covered with the defined primitives. Next, a defined binary concatenation operation is applied for each pair of the detected primitives. By linking the binary operations on primitives the following sequence, called *primitive sequence* is generated:

$$a_0(j1)a_1(j2)...a_{n-1}(j_n)a_n.$$

The end points: a_0 and a_n of a primitive seqence are called the h-point and t-point, respectively. For each such sequence composed of $(n+1)$ primitives the label called *PS-Label* (Primitive Sequence-Label) denoted: $\langle ps, idr\rangle$ is given by the following formulae:

$$ps = (n+1) + L + M$$
$$idr = j_1$$

where:

L is a number of concatenations of the form: $a_i(j_{i+1})a_{i+1}$, for $i = 0, 1, ..., n-1$, such that $prm(a_i) = prm(a_{i+1})$,

M is a number of primitives a_i such that $prm(a_i) \in \{|, -\}$,

idr denotes the initial direction of a rotation.

Fig. 9.6a) shows an example of generating the primitive sequence: 1(0)2(1)3(2)4(3)5 with *PS-Label* $\angle 5, 0\rangle$. For sequence composed of one primitive, a *PS-Label* is given in the following way: $\langle 1,1\rangle$ for the primitive of type '–', $\langle 1,2\rangle$ for type '/', and $\langle 1,3\rangle$ for type '\', respectively. When the primitive sequence is cyclic, the *PS-Label* $\langle 0,0\rangle$ is given to such curve.

2.1.2.2. *Connection of primitive sequences*

Two primitive sequences: $e_0 = a_0(i_1)a_1(i_2)...(i_m)a_m$ and $e_1 = b_0(j_1)b_1(j_2)...(j_n)b_n$ with the *PS-Labels* of e_0 and e_1 equal to $\langle ps_0, id_0\rangle$ and $\langle ps_1, id_1\rangle$ respectively, could be connected in two ways. If the primitive a_0 is identical to the primitive b_0 the two primitive sequences e_0 and e_1 are connected to one another by sharing the first primitive of each one. It is a *h-connection* and is denoted as: $e_0(h)e_1$ (e.g. see Fig. 9.6b)). If the primitive a_m is identical to the primitive b_n the two primitive sequences e_0 and e_1 are connected to one another by sharing the last primitive of each one. It is a *t-connection* and is denoted as $e_0(t)e_1$ (e.g. see Fig. 9.6c)).

Suppose that a curve consists of n primitive sequences e_i, $i = 1, 2, ..., n$, and e_i is connected to e_{i+1} , for $i = 1, 2, ..., n-1$, in such a way that:

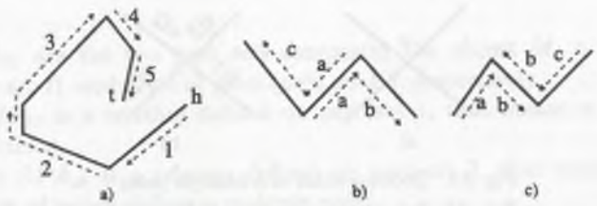


Fig. 9.6. a) Illustration of generating a primitive sequence: h, t denote the h-point and t-point of the primitive sequence, b), c) Illustration of concatenation of primitive sequences: h-connection (b) and t-connection (c)

Rys. 9.6. a) Ilustracja generacji sekwencji prymitywów: h, t oznaczają h-punt i t-punkt sekwencji prymitywów, b), c) ilustracja konkatenacji sekwencji prymitywów: połączenie typu h (b), oraz połączenie typu t (c)

$$e_i(c_{i+1})e_{i+1} \ (c_i \in h, t)$$

where $c_{i+1} = t$ if $c_i = h$, and $c_{i+1} = h$ if $c_i = t$; $(i = 2, 3, .., n-1)$. By linking the connections of the primitive sequences, the following string which describes the *quasi-topological structure* of a simple curve is generated:

$$\langle ps_1, idr_1\rangle(c_2)\langle ps_2, idr_2\rangle(c_3)...(c_n)\langle ps_n, idr_n\rangle$$

where $\langle ps_i, idr_i\rangle$ is a PS-Label of a primitive sequence e_i. In the case of closed curve, the last element is identical to the first one.

2.2. Geometric structure of a multiple point

The node in a skeleton of a 2D-curve, whose order (the number of primitive sequences incident to the node) is equal to or more than 3 is a *multiple point*. Four possible types of adjacent structures of two primitive sequences on a multiple point have been used: X type (crossing), K type (touch) and T and Y types depicted in Fig. 9.7.

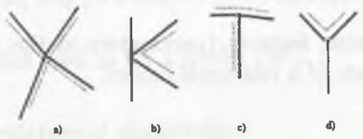
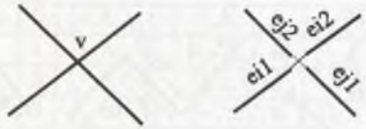


Fig. 9.7. Four structures of a multiple point

Rys. 9.7. Cztery struktury punktu wielokrotnego

The structure of a given multiple point v is described by a binary relation of primitive sequences (or their connections) incident to v, and is obtained by the following *decomposition procedure* of v. This procedure tries to connect in *pairs* the incident to v primitive sequences, so as the conditions (based on *local features* around v, *angles* in the presented method), for the four types of structures of a multiple point are satisfied. Thus, a general 2D-curve is decomposed into unicursal components by taking account of the smoothness



a) b)

Fig. 9.8. Decomposition of a multiple point

Rys. 9.8. Dekompozycja punktu wielokrotnego

of directional changes at a multiple point, that means local rules for adjacent edges. When local feature around multiple point is complex, decision should be made in terms of global features of adjacent curves rather than the local rules for adjacent edges. As a result, a multiple point of n degree is decomposed into $r = \lceil (n/2) \rceil$ curves incident to v. Fig. 9.8 ilustrates the decomposition of the multiple point with n=4 resulting in 2 pairs of primitive sequences: $p_k = (e_{i1}, e_{i2})$ and $p_l = (e_{j1}, e_{j2})$.

For X and K types, the resulting structure of a multiple point v is depicted in the following: $[p_k](T)[p_l] = [(e_{i1}, e_{i2})](T)[(e_{j1}, e_{j2})]$, where $T = K, X$ and $e_{i1}, e_{i2}, e_{j1}, e_{j2}$ are the incident to v primitive sequences or their connections.

For Y and T types, the resulting structure of a multiple point v is depicted in the following: $[p_k]T[p_l] = [(e_{i1}, e_{i2})](T)[(e_{j1}, 0)]$, where $T = K, X$ and other notations used, are the same as above.

3. The hybrid description of a general 2D-curve

The proposed, *hybrid description* of a general 2D-curve is composed of:

- Relational description, which captures the global, qualitative, and discrete features (quasi-topological structure of a curve and geometric structure of multiple point),
- Quantitative, continuous features (parameters of the global features) which are attached to components of a relational model.

3.1. The relational description

The proposed relational description is based on the extended notion of relational structure proposed in [292, 293], and described in chapter 1. As opposed to [292, 293], we allow an object to be composed of many different part sets.

The *relational description* of a model M of a geographic symbol is given in the following:

$$D(I) = \{SP_M, SPA_M, R_1^I, R_2^I, NPA_M, NP_M, NM_M, C_M, L_M, P_M, MP_M, op_M, om_M\}$$

where:
SP_M and SPA_M are the two part sets composing the object M, a set of primitive sequences (*part set 1*), and a set of pairs (*part set 2*), respectively,
$R_1^I \subset SP_M \times SP_M$ is a relation defined on part set 1, that means a set of primitive sequence connections,
$R_2^I \subset SPA_M \times SPA_M$ is a relation defined on part set 2, that means a set of pair connections, each of which defines a multiple point,
NPA_M, NP_M, NM_M are the numbers of pairs, primitive sequences and multiple points in a model M, respectively,
C_M: $\{1, ..., NP_M\} \times \{1, ..., NP_M\} \rightarrow Z \subseteq \{i(c)j : 1 \leq i, j \leq NP_M, c \in \{h, t\}\}$ is a primitive sequence connection labeling function,
$L_M : \{1, ..., NP_M\} \rightarrow 2^{PSL}$, where $PSL = \{\langle ps, idr \rangle \mid ps \in N; idr \in \{0, 1, 2, 3, 4\}\}$ is a primitive sequence labeling function,
$P_M :\subseteq \{1, ..., NP_M\} \rightarrow 2^{MPA}$, where $MPA = \{1, ..., NPA_M\}$ is a primitive sequence indexing function,
$MP_M : \{1, ..., NM_M\} \rightarrow 2^{MPS}$, where $MPS = \{[p_i](t)[p_j]; 0 \leq p_i, p_j \leq NPA_M; t = X, K, T, Y\}$ is a multiple point labeling function,
$op_M : P \rightarrow \{0, 1\}$; $om_M : M \rightarrow \{0, 1\}$ are optionality functions having the following meaning: if $op_M(i) = 0$ $(om_M(i) = 1)$ then the i-th primitive sequence (multiple point) is optional, if $op_M(i) = 1$ $(om_M(i) = 1)$ then i-th primitive sequence (multiple point) is mandatory.

According to the above stated labeling and indexing functions, $L_M(i)$ denotes the set of eligible labels of the i-th primitive sequence, $P_M(i)$ the set of pair's indexes to which i-th primitive sequence belongs, $C_M((i, j))$ the label of a connection of i and j primitive sequences, $MP_M(i)$ denotes the set of eligible structures of a multiple point i.

The description scheme of an object O is given formally as:

$$D(O) = \{SP_O, SPA_O, R_1^O, R_2^O, NPA_O, NP_O, NM_O, C_O, L_O, P_O, MP_O, op_O, om_O\}$$

where the same notations are used as for a model (M).

3.1.1. *Example of a relational description*

Figure 9.10 shows the example of a relational description of a curve, whose vectorized skeleton is depicted in Fig. 9.9a. Its skeleton has been covered with primitives: $w0$, $w1$, $w2$, $w3$, $w4$, and $w5$. Next, a binary concatenation operation has been applied, and such connected primitives have been linked to form the following primitive sequences: *F1, F2, F3*, and *F4*. As a result of the connection of two primitive sequences $F3$ and $F4$, a connection $F5$: $\langle 2, 1 \rangle (h) \langle 3, 3 \rangle$ has been generated. An existing multiple point $v1$ has been decomposed according to the described procedure, which reveals its structure: *[F1,F5](T)[F2,0]*.

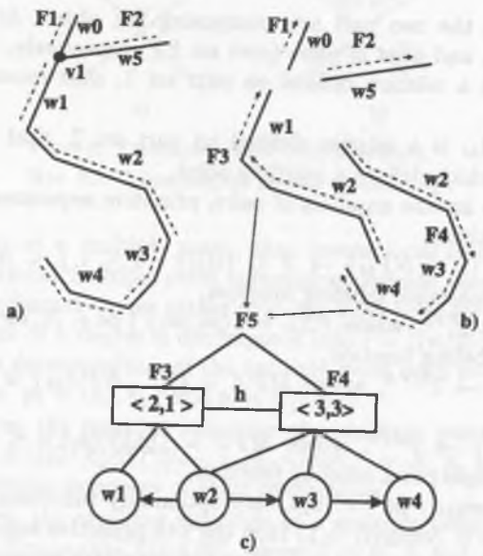


Fig. 9.9. a) A general 2D-curve, b) decomposition of 2D-curve into primitive sequences and multiple point, c) quasi-topological structure of the curve $F5 : F3(h)F4$

Rys. 9.9. a) Krzywa 2D, b) dekompozycja krzywej na prymitywne sekwencje oraz punkt wielokrotny, c) quasi-topologiczna struktura krzywej $F5 : F3(h)F4$

SP_I	SPA_I	R_1^I	R_2^I
F1 : w0	P1 = [F1, F5]	F5 = (F3, F4)	v1 = (P1, P2)
F3 : w2 (1) w1	P2 = [F2, 0]		
F4 : w2 (3) w3 (0) w4			
F2 : w5			

C_I	L_I	P_I	MP_I
C_I (F5) = F3 (h) F4	F1 : <1,1>	P_I (F1) = P1	MP_I (v1) = P1 (T) P2
	F3 : <2,1>	P_I (F3) = P1	
	F4 : <3,3>	P_I (F4) = P1	
	F2 : <1,1>	P_I (F2) = P2	

Fig. 9.10. Relational description of a curve from Fig. 6.9a). $op_M(F1) = 0, om_M(v1) = 0$, the remaining primitive sequences are mandatory. For simplification, pairs are identified with their indexes. Only one eligible label for primitive sequence, and for multiple point are depicted

Rys. 9.10. Opis relacyjny krzywej z Rys. 9a). $op_M(F1) = 0, om_M(v1) = 0$, pozostałe sekwencje prymitywów są obowiązkowe. Pary identyfikowane są ze swoimi indeksami. Pokazano tylko jedną dozwoloną etykietę dla sekwencji prymitywów oraz punktu wielokrotnego

3.2. Parameterization of a shape

Since the description method ignores metric information such as length, curvature, angle, or position, providing that object representing different symbols have the identical structure, parameterization of shapes, namely primitive, primitive sequence and multiple point has been introduced.

The normalization of a shape has been performed with respect to the bounding box (the upright rectangle just enclosing the shape) of the symbol, by scaling the longest side of the box to unity and preserving the aspect ratio. The center of the bounding box is set to be the origin (0,0).

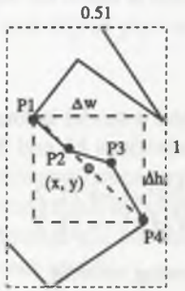


Fig. 9.11. Illustration of parameters describing primitive $P1P4$

Rys. 9.11. Ilustracja parametrów opisujących sekwencję prymitywów

The following parameters describes the three shapes:

1. $(x, y, \Delta w, \Delta h, x_h, y_h, x_t, y_t)$ for primitive sequence with PS-Label $\langle ps, idr \rangle$, and $ps > 0$; (x, y) is the center coordinates of the bounding box of the primitive sequence, Δw, Δh are width and height of the bounding box, (x_h, y_h), (x_t, y_t) are the locations of the h and t-points.

2. $(x, y, R_T, \Delta w, \Delta h)$ for primitive sequence with PS-Label $\langle 0, 0 \rangle$; (x, y) is the center coordinates of the loop, Δw, Δh are the width and height of the bounding box, R_T is the thinness ratio defined by: $R_T = 4\pi A/P^2$, where A is an area, and P is the perimeter of the loop.

3. $(x, y, conv, \Delta w, \Delta h)$ for a primitive; (x, y) is the center coordinates of the chord, Δw, Δh are the length of projections of the arc to the x and y axes, $conv$ represents convexity defined as: $conv = 2S/P_1P_n$, were S is the area of the polygon $(P_1, ..., P_n, P_1)$ where $P_1, ..., P_n$, is a point sequence forming a primitive, with P_1 and P_n being its head and tail, respectively.

4. (x, y) for multiple point, being its coordinates.

4. Hybrid matching

Finding a match between an unknown object O and a model M is accomplished in a devised, two-stage procedure, consisting of:

- Structural matching
- Distance calculation in parameter space.

If the object is matched with a relational model, the normalized Euclidean distance is computed for parameters on structural components (primitive sequences, primitives, multiple points), and the final decision is made on the basis of the distance.

4.1. Relational matching

Relational matching of an object and a model is formalized in the following. An object O is matched with a model M if surjections $h1$ and $h2$:

$$h1 : \{1, 2, ..., NP_M\} \rightarrow \{1, 2, ..., NP_O\} \cup \{\emptyset\}$$
$$h2 : \{1, 2, ..., NM_M\} \rightarrow \{1, 2, ..., NM_O\} \cup \{\emptyset\}$$

are found such that satisfy the following criteria:

1. If $h1(i) = h1(j) \neq \emptyset$ for $i, j \in \{1, 2, ..., NP_M\}$ then $i = j$
2. If $op_M(i) = 1$ for $i, j \in \{1, 2, ..., NP_M\}$ then there exists $j \in \{1, 2, ..., NP_O\}$ such that $h1(i) = j$
3. For each primitive sequence $j = h1(i)$ in the object its PS-Label $L_O(j)$ must be a member of a set $L_M(i)$ of a corresponding primitive sequence i in the model: $L_O(j) = L_O(h1(i)) \in L_M(i)$.
4. Connections of primitive sequence defined in the model are also preserved in the object under the correspondence $h1$: $C_O = \{h1(i)(c)h1(j) : i(c)j = C_M((i, j))\}$
5. If $h2(i) = h2(j) \neq \emptyset$ for $i, j \in \{1, 2, ..., NM_M\}$ then $i = j$.
6. If multiple point i in the model is mandatory, and for some primitive sequences forming a multiple point i, corresponding elements are found in the object, then a corresponding multiple point must exist in the object: If $om_M(i) = 1$ for $i, j \in \{1, 2, ..., NM_M\}$ and $h1(m) \neq \emptyset$ and $h1(n) \neq \emptyset$ for some $[m, p_m]t[n, p_n] \in MPI(i)$ then there exists $j \in \{1, 2, ..., NM_O\}$ such that $h2(i) = j$.
7. For each multiple point $j = h2(i)$ in the object O, its structure must be a member of a set $MP_M(i)$ of a corresponding multiple point i in the model: $MP_O(h2(i)) \in MP_M(i)$, that means there exists some $[p_q]t[p_r] \in MP_M, p_q, p_r \in P_M$ such that $[h1(p_q)]t[f(p_r)] \in MP_O(h2(i))$.

4.1.1. *Search procedure for relational matching*

Finding the two mappings $h1$ ang $h2$ described in the previous section, being the relational homomorphisms, is accomplished in the following way. The mapping $h1$, being the relational homomorphism (defined in chapter 1) from a relation R_1^I to a relation R_1^O is found with constraints 1 through 4 from the previous section. If homomorphism $h1$ is found out successfuly, then the homomorphism $h2$, from a relation R_2^I to a relation R_2^O is found with constraints 5 through 7 under the mapping $h1$.

```
procedure TreeSearch(A, B, H, R1^M, R1^O)
    A, B: part sets in a model and object, respectively
    H: homomorphism
    R1^I, R1^O: relation over model and object part sets, respectively
  a := first(A)
  for each b ∈ B do
    H' = H ∪ {(a, b)}
    ok := True
    for each N-tuple r ∈ R1^M containing component a
      and whose other components are all in domain (H) do
      if r ∘ H' not in R1^O or constraints are not satisfied
      then begin ok := False; break endif
    end for
    if ok then
      begin
        A' := remainder(A)
        if isempty(A')
        then output(H')
        else TreeSearch(A', B, H', R1^M, R1^O)
      end
  end for
```

Fig. 9.12. Backtracking tree search procedure for finding all relational homomorphisms from R_1^M to R_1^O

Rys. 9.12. Procedura przeszukiwania drzewa z powrotami dla znalezienia wszystkich relacyjnych homomorfizmów z relacji R_1^M do R_1^O

Fig. 9.12 shows the algorithm finding all relational homomorphisms (all mappings) $H : A \rightarrow B$ from a relation R_1^M, defined over part sets A in a model, to a relation R_1^O defined over part set B in an object. The backtracking tree search begins with the first unit of A. This unit can potentially match each label in set B. Each of these potential assignments is a node at level 1 of the tree. The algorithm selects one of these nodes, makes the assignment, selects the second unit of A, and begins to construct the children of the first node, which are nodes that map the second unit of A to each possible label of B. At this level some of the nodes may be ruled out because they violate the constraints.

The process continues to level $|A|$ of the tree. The paths from the root node to any successful nodes at level $|A|$ are the relational homomorphisms.

The backtracking tree search has exponential time complexity. As the shape is described by the presented method in terms of global features, the number of elements is not large in the presented description technique, and therefore computational complexity is not serious problem.

Each of the defined in the previous section homomorphisms $h1$ and $h2$, is found by a backtracking tree search procedure depicted in Fig. 9.12, during which primitive sequence and multiple points labels are checked. Since the symbol shape is described in terms of global features, the description is not sensitive to local deformations. Therefore, we employed neither cost functions, nor editing operations for finding optimal correspondence between two strings representing labels of primitive sequences, their connections and labels of multiple points. The exact string matching method is applied, in which two strings are matched against one another only if they are identical.

4.1.2. *Structural error of matching*

The *total error* $E(h1, h2)$ of $h1$ and $h2$ with respect to D_M and D_O is, as described in chapter 2, the sum of the *structural errors* for each of the two pairs of corresponding relations (R_1^M, R_1^O), and (R_2^M, R_2^O):

$$E(h1, h2) = E_s^1(h1)E_s^2(h2)$$

The *structural error* of $h1$ ($h2$) for the pair of corresponding relations in D_M and D_O $((R_1^M, R_1^O), ((R_2^M, R_2^O))$ is given by:

$$E_s^1(h1) = |R_1^M \circ h1 - R_1^O| + |R_1^O \circ h1^{-1} - R_1^M|$$
$$E_s^2(h2) = |R_2^M \circ h2 - R_2^O| + |R_2^O \circ h2^{-1} - R_2^M|$$

The structural error indicates how many tuples in R_1^M (R_2^M) are not mapped by $h1$ ($h2$) to tuples in R_1^O (R_2^O) and how many tuples in R_1^O (R_2^O) are not mapped by $h1^{-1}$ ($h2^{-1}$) to tuples in R_1^M (R_2^M).

5. Experimental results

Relational descriptions of all geographic symbols from Polish Geodesic Standardization ([124]) were prepared according to the method presented in this chapter. An example of a relational description (with one PS-Label for each primitive sequence and one structure for each multiple point), for a fountain geographic symbol is shown in Fig. 9.15. All primitive sequences and multiple points are mandatory.

In a learning phase performed on a set of samples from scanned, real, geodetic maps (scale 1:500), sets of eligible PS-labels for primitive sequences, and eligible structures for multiple points in each of the models of geographic symbols are incremented.

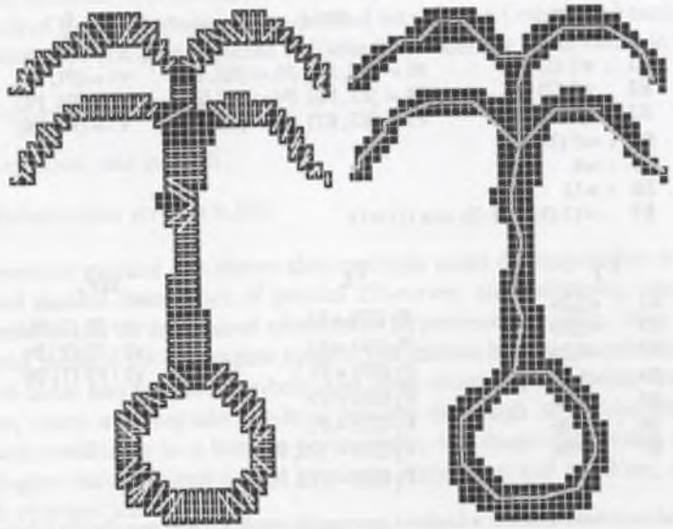


Fig. 9.13. The example of the proposed method for a fountain geographic symbol: a) simple region extraction, b) vectorized skeleton

Rys. 9.13. Przykład zastosowania zaproponowanej metody dla symbolu fontanny: a) ekstrakcja regionów prostych, b) szkielet wektorowy

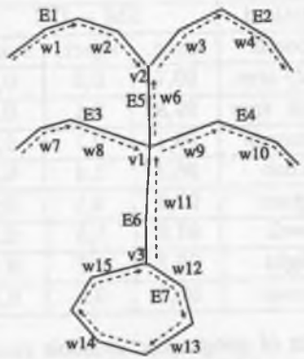


Fig. 9.14. Decomposition of a fountain geographic symbol into primitive sequences and multiple points

Rys. 9.14. Dekompozycja symbolu fontanny na sekwencje prymitywów i punkty wielokrotne

SP_I	SPA_I	R_2^I
E1 : w1 (2) w2	P1 = [E3, E4] P2 = [E5, E6]	v1 = (P1, P2)
E2 : w3 (2) w4	P3 = [E1, E2] P4 = [E5,0]	v2 = (P3, P4)
E3 : w7 (2) w8	P5 = [E7, E7] P6 = [E6,0]	v3 = (P5, P6)
E4 : w9 (2) w10		
E5 : w6		
E6 : w11		
E7 : w12 (3) w13 (0) w14 (1) w15		

L_I	P_I	MP_I
E1 : <2,2>	P_I (E3) = P1	v1 : P1 (X) P2
E2 : <2,2>	P_I (E4) = P1	v2 : P3 (Y) P4
E3 : <2,2>	P_I (E7) = P5	v3 : P5 (T) P6
E4 : <2,2>	P_I (E2) = P3	
E5 : <1,0>	P_I (E1) = P3	
E6 : <1,0>	P_I (E5) = {P2, P4}	
E7 : <0,0>	P_I (E6) = {P2, P6}	

Fig. 9.15. Relational description of a fountain geographic symbol. Only one eligible labels for primitive sequence and for multiple point are depicted. For simplification, pairs are identified with their indexes

Rys. 9.15. Opis relacyjny dla symbolu fontanny. Pokazano tylko jedną dozwoloną etykietę dla sekwencji prymitywów oraz punktu wielokrotnego. Pary identyfikowane są ze swoimi indeksami

Method	SM + DC		
Result	recog.	reject.	subst.
leafy tree	99,7	0,2	0,1
conif. tree	99,8	0,1	0,1
fountain	99,1	0,7	0,2
bush	98,3	1,4	0,3
grass	97,2	2,1	0,7
well	97,8	1,5	0,7
light	99,2	0,7	0,1
cross	99,3	0,5	0,2

Table 9.2. Experimental results of geographic symbols recognition using the presented method; SM - Structural Matching, DC - Distance Calculation

Tests were made on samples of the scanned, geodetic maps from which representations of geographic symbols were prepared according to the presented method.

The result of both two stages of the devised recognition (structural matching (SM) + distance calculation (DC)) is depicted in Table 9.2, while the mean result is the following:

1. The recognition rate is 98, 8%,
2. The rejection rate is 0, 9%,
3. The substitution error is 0, 3%.

The presented method has shown that multiple point decomposition is a clue to the compact and flexible description of general 2D-curves, and deformed, handprinted geographic symbols can be recognized effectively. Experimental results show that the proposed, hybrid method for geographic symbol recognition is very powerful in representing the shape of these handprinted symbols, and gives satisfying recognition rate.

However, since we integrate low-level features into high level ones with some simple geometric conditions in a bottom-up sequence, the shape description is sensitive to some topological deformations caused by stroke connection and breaking, and the global structure is changed accordingly.

Chapter 10

Classifier based on inexact graph matching

1. Introduction

Recognition of *geographic symbols* as well as *text* which convey meanings associated with logical structures represented in maps, is an important aspect of any map analysis system ([320]). Some examples of such geographic symbols from 4 layers of Polish Fundamental Land Map (PFLM) are shown in Fig. 9.1 in the previous chapter. This chapter presents another structural classifier for recognition of *geographic symbols* (published in [323]) which relies on an *error-tolerant (inexact) graph matching* procedure between the *attributed graphs* representing the unknown object, and the prototype object.

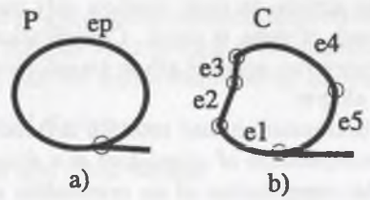


Fig. 10.1. a) The bush geographic symbol, b) its distorted version

Rys. 10.1. a) Symbol krzewu, b) jego zniekształcona wersja

Fig. 10.2. The examples of the complex structural deformations of a bush geographic symbol

Rys. 10.2. Przykłady złożonych deformacji strukturalnych symbolu krzew

The existing, traditional error-tolerant graph matching algorithms (for example [49, 283, 291, 55, 211, 230, 316]) assume that the input shape is identified better if the edges

and nodes of one graph are matched *one to one* to the edges and nodes of the second graph. However, this asumption is not necessarily true. Sometimes, for example, the nodes are artificially added by a presence of a spurious, concave angle, like that between the features $e2$ and $e3$ in Fig. 10.1b. As a result, an arc or a stroke in a prototype may need to be matched into a path of arcs and (or) strokes. For example the arc ep in Fig. 10.1a need to be matched into the path $\{e1, e2, e3, e4, e5\}$ of strokes and arcs.

Therefore, the traditional computational versions of inexact graph matching, where every effort is made to find *one-to-one* mapping, are **not useful** for the hand-printed geographic symbol recognition. For the above reasons, the inexact graph matching algorithm used in the proposed recognition method is based on the idea of *multiple-to-one mapping*. This idea has been previously used in the graph matching algorithm of the character recognition method proposed in [270]. However, the algorithm presented in [270] results in the incorrect matching when more complex structural deformations occur (i.e. additional branch points, loops, branches, etc. - see Fig. 10.2 for some examples), as well as it cannot cope with the topological deformations caused by stroke connection. We modified the algorithm [270] to to assure the correct behaviour in more complex situations, and to enrich the set of the structural deformations to which the algorithm [270] is tolerant. Moreover, as many geographic symbols are multicomponent, we introduced the new structure, the component graph, as well as the appropriate graph matching procedure.

The proposed recognition method diverges from the existing statistical and structural classifiers which require learning on a large set of samples, or a wide number of prototypes to capture the variability of a shape. In the presented method the variability of the actual symbol shapes, as well as noise, are modeled as a series of *graph transformations* from groups of features in the data, to the features in each prototype during recognition process. Symbol shape definitions are simple: in most symbols only one prototype per class, and the number of features in each of them is small. Classification of symbols is defined as a minimization problem among the possible graph transformations that map the input shape into the prototypical shapes.

The presented method is translation and rotation invariant. Rotation invariance is achieved by a preliminary computation of orientation of a shape and obtaining a symbol in a reference position. The computation of an orientation of a symbol can be easily done because the symbols are drawn parallel or perpendicular to the axis of a geographic object which they symbolize, and due to the intelligent map analysis strategy described in chapter 8 these axis are known before recognition.

2. Geographic symbol representation

Each (multicomponent) symbol is represented by a set of attributed graphs: the *component graph* denoted as K_C for a candidate symbol or K_P for a prototype symbol, describing the relative locations of the components in a given symbol (used only in a

case of the multicomponent symbols), and a set C (P) of the graphs c_i (p_i) (where $i = 1, ..., NrOfComponents$), each of which represents an individual component of a given candidate (prototype) symbol. Th edges of each graph in the above set are the *features* of a symbol (*strokes* or *convex arcs* (the last only in a case of a graph of component)), and the nodes of a graph are junctions among the strokes and arcs and represent the *singular points* on a shape (a branch point, end point, concave node, sharp corner). The singular points convey the spatial relationships among the structural features.

A *component graph* of a multicomponent symbol is created in the following way. For each component, as well as for a whole symbol the minimum bounding rectangles with the edges parallel to the axis of a coordinate system are computed. The nodes of a component graph are the center points of the computed rectangles, while each of its edges is defined by the center points of the two minimum bounding rectangles: one of a whole symbol and the second of a component. Fig. 10.3 shows the construction of the component graph of a multi-component leafy tree symbol.

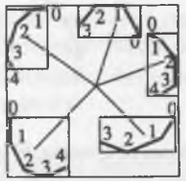


Fig. 10.3. The example of the component graph of a multicomponent leafy tree symbol

Rys. 10.3. Przykład grafu struktury dla symbolu drzewa liściastego

A graph of each component is obtained as a result of the vectorized thinning algorithm ([315]). Figures 10.5 and 10.6 show the examples of the prototype graphs of components of a fountain and a leafy tree geographic symbols, respectively, while Figures 10.8 and 10.9 show the examples of the candidate graphs of components of these symbols.

Each feature in a graph has two nodes identified as follows:

- *Left, right* for horizontal strokes.
- *Down, top* for vertical and tilted strokes.
- *First, second* for arcs, according to an order in which the node is found when the arc is traversed counterclockwise.

Each stroke has two *attributes*: an *angle* α and a *length* l (normalized with respect to a height of a symbol) (Fig. 10.4a), while each arc has three *attributes*: a *beginning* φ and an *ending* ϕ *angles* in the counterclockwise direction, and a *length* (normalized with respect to a height of a symbol) (Fig. 10.4b).

A full description of a feature comprises the following items: a *label*, *type* (| for strokes, $\smile$ for arcs), *index of a 1-st node*, *index of a 2-nd node*, 1 if a feature is optional (or a *bridge*), otherwise 0, *length*. Further items, depending on a type consist of an *angle* for a stroke, or two angles: a *beginning* and *ending*, for arcs.

For example, the line:

0 | 1 0 0 0.5 1.57

describes the feature labeled 0, being the *stroke*, whose left and right nodes have 1 and 0 indices, the angle having value 1.57.

The line:

3 ⌣ 2 7 1 1.5 0.78 3.14

describes the feature labeled 3, being the optional *arc*, whose first and second nodes have 2 and 7 indices, respectively, with the normalized length 1.5 whose first and second angles have 0.17 and 3.14 values, respectively.

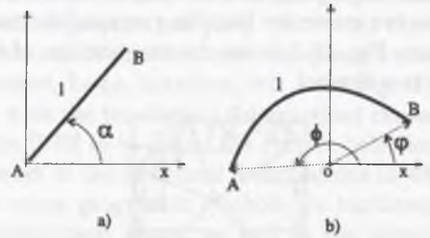
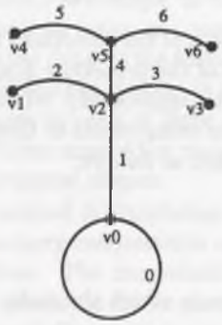


Fig. 10.4. a) Attributes of a stroke, b) attributes of an arc

Rys. 10.4. a) Atrybuty odcinka, b) atrybuty łuku

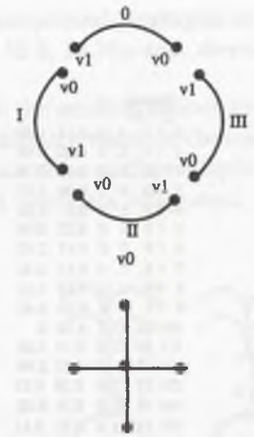


fountain 1
1.0 1.0 2.0 0.8 1.0
7 7
0 ^ 0 0 0 0.9 1.57 1.57
1 | 2 0 0 0.5 1.57
2 ^ 2 1 0 0.28 0.79 2.36
3 ^ 3 2 0 0.28 0.79 2.36
4 | 5 2 0 0.16 1.57
5 ^ 5 4 0 0.28 0.79 2.36
6 ^ 6 5 0 0.28 0.79 2.36

Fig. 10.5. The graph of component of the prototype fountain geographic symbol (left) and its corresponding structural description (right)

Rys. 10.5. Graf komponentu wzorcowego symbolu fontanny (na lewo) i jego opis stukturalny (na prawo)

Fig. 10.7 shows a general format of a symbol description. It consists of a class name, number of components followed by *weights* (described later), followed by descriptions of each component, and description of a component graph. A description of each component (part 2 in Fig. 10.7) is composed of the number of features and the nodes in a component, and the descriptions of each feature as described above. A component graph description (part 3 in Fig. 10.7) is filled only for the multicomponent symbols, and it consists of the



leafy tree 4
1.0 0.7 1.5 0.8 1.0
1 2
0 ^ 0 1 0 0.7 0.39 2.74
1 2
0 ^ 0 1 0 0.7 1.96 4.32
1 2
0 ^ 0 1 0 0.7 3.53 5.89
1 2
0 ^ 0 1 0 0.7 5.10 1.18

component graph
0 0.4 1.57
1 0.4 3.14
2 0.4 -1.57
3 0.4 0

Fig. 10.6. The graphs of components and the component graph of the prototype leafy tree geographic symbol (left) and their corresponding structural descriptions (right)

Rys. 10.6. Grafy komponentów oraz graf struktury wzorcowego symbolu drzewa liściastego (na lewo) oraz jego opis strukturalny (na prawo)

lines describing the attributes of the edges (features) of a component graph (length and angle only). Figures 10.5, 10.6, 10.8, 10.9 (right side) show the examples of the symbol descriptions of the fountain and the leafy tree geographic symbols.

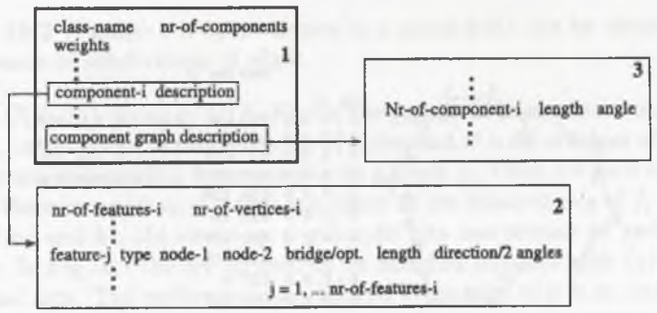
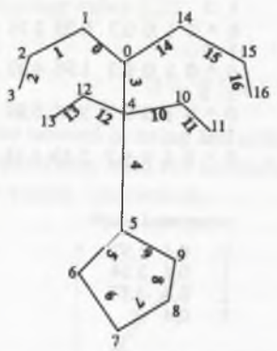


Fig. 10.7. A general format of a symbol description

Rys. 10.7. Format opisu symbolu

3. Homomorphism between graphs

Exact matching of structural descriptions is enough to identify candidates with the prototypes if a noise, or alteration of features are not present. However, any real structural

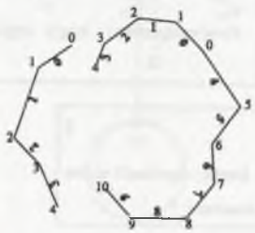


fountain 1

17	17				
0	1	0	0	0.15	2.52
1	1	2	0	0.20	0.46
2	2	3	0	0.02	0.78
3	0	4	0	0.09	1.57
4	4	5	0	0.5	1.57
5	5	6	0	0.22	0.96
6	6	7	0	0.17	2.43
7	8	7	0	0.11	0.32
8	9	8	0	0.13	1.29
9	5	9	0	0.19	2.42
10	10	4	0	0.18	0
11	10	11	0	0.14	2.26
12	12	4	0	0.20	2.96
13	12	13	0	0.19	0.85
14	14	0	0	0.16	0.62
15	14	15	0	0.11	2.81
16	15	16	0	0.09	2.21

Fig. 10.8. The graph of the broken component of the candidate fountain geographic symbol (left) and its corresponding structural description (right). Feature labels depicted in bold

Rys. 10.8. Graf komponentu poprzerywanego wzorcowego symbolu fontanny - kandydata (na lewo) i jego opis stukturalny (na prawo). Etykiety cech zaznaczono poprzez wytłuszczenie



leafy tree 2

10	11				
0	1	0	0	0.2	2.35
1	2	1	0	0.2	3.05
2	2	3	0	0.2	0.78
3	3	4	0	0.07	1.32
4	0	5	0	0.34	2.24
5	5	6	0	0.23	1.0
6	6	7	0	0.12	1.57
7	7	8	0	0.17	0.78
8	8	9	0	0.27	0
9	10	9	0	0.06	2.15

4	5				
0	0	1	0	0.15	0.78
1	1	2	0	0.29	1.44
2	2	3	0	0.23	2.24
3	3	4	0	0.19	2.15

component graph

0	0.12	0
1	0.37	-2.94

Fig. 10.9. The graphs of components and the component graph of the candidate leafy tree geographic symbol (left) and their corresponding structural description (right). Some components are glued. Feature labels depicted in bold

Rys. 10.9. Grafy komponentów oraz graf struktury wzorcowego symbolu drzewa liściastego - kandydata (na lewo) i jego opis stukturalny (na prawo). Niektóre komponenty są sklejone

description of hand-printed geographic symbol has random structural deformations (e.g. see Figures 10.8, 10.9, 10.10), and, therefore, inexact identification is necessary for real-world systems.

As opposed to the existing inexact graph matching algorithms (i.e. [49, 283, 291, 55, 211, 230, 316]) which are based on the one-to-one matchings, the presented inexact graph matching algorithm relies on the *multiple-to-one* matchings from the features of the graph c of the unknown candidate component, to the features of the graph p of the prototype component.

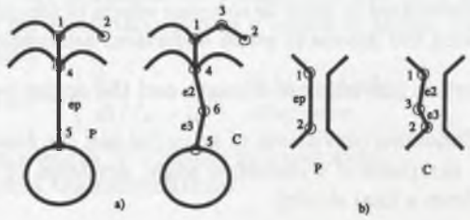


Fig. 10.10. a), b) Graphs P (prototype) and C (candidate) are homomorphic

Rys. 10.10. a), b) Grafy P oraz C są homomorficzne

Definition 10.1 An edge e of a graph is *subdivided* by replacing it by a node and two edges such that the new edges are adjacent to the new node and to the nodes adjacent to e.

Definition 10.2 A graph c is *homomorphic* to a graph p if c can be obtained from p by a sequence of subdivisions of edges.

Let $c = G(J_c, F_c)$ and $p = G(J_p, F_p)$ be the graphs of a candidate and a prototype components, respectively (J denotes a set of nodes and F a set of edges of a graph). Let c' be a subgraph of a graph c, homomorphic to a graph p. Then, for each edge $f_j \in p$, for $j = 1, ..., n$, there is a *path* $\alpha_j \in c$ that is a result of the subdivisions of f_j. For example, in Fig. 10.10a) and b), the stroke $ep \in p$ is split into two strokes $e2$ and $e3$ to get the graph the c. In Fig.10.1 the arc ep need to be matched into the path $\{e1, e2, e3, e4, e5\}$ of strokes and arcs. The paths are node disjoint, every edge of p is on one and only one of the paths $\{\alpha_j\}$. However, there may be edges of c that are not covered by the paths $\{\alpha_j\}$.

Definition 10.3 A homomorphism H between graph p and subgraph c' of c is determined by a set of the following tuples, each of which is of the form *(candidate path, prototype feature)*:

$$H(p, c, c') = \{(\alpha_j, f_j)\}, \text{ for } j = 1, ..., n$$

Definition 10.4 A *multiple-to-one matching* is a set of tuples that define a homomorphism.

4. Graph transformations and their costs

A *transformation* eliminates a deformation in an input graph which is identified when there is a singularity (concave node, sharp angle, etc.) in a candidate that is not present in a particular prototype. Given an input graph, it is very likely that the transformations allow it to *match* to the different prototypes. However, we expect that a candidate graph must be tranformed less when matched to its correct prototype, than to any other prototype. As a result, an *amount of a deformation* (i.e. a *cost*) imparted by the transformations should be accumulated in order to compare effects of the different transformation sequences. The following five groups of *graph distortions* are considered.

1. Geometric distortion (deviation of a length and the angles between the strokes).
2. Morphological distortion (deviations of an initial and the final angles between arcs, concave angles, sharpness of a smoothed angle, deviation of straightened arcs and lists of strokes from a final stroke).
3. Nonmatched features in a candidate.
4. Nonmatched features in a prototype.
5. Non-topological distortions (broken or glued symbols).

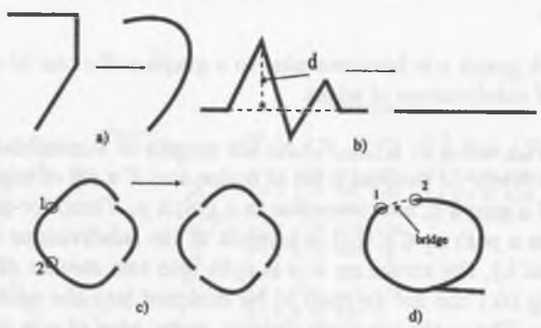


Fig. 10.11. Graph transformations: a) the path of strokes rewritten into the arc, b) straightening of the path of strokes, c) the glued multicomponent tree symbol - 2 divisions are required to get the correct tree symbol, d) the use of bridges to cope with the breakings

Rys. 10.11. Transformacje grafu: a) łańcuch odcinków zamieniony na luk, b) wyprostowanie łańcucha odcinków, c) sklejony symbol drzewa liściastego - 2 operacje podziału są konieczne dla uzyskania poprawnego symbolu drzewa, d) użycie mostków

To correct the above mentioned local distortions in a candidate graph, the following six groups of *graph transformations*, which corresponds to the mentioned above five groups of graph distortions have been proposed. The last group of the two non-topological transformations has been divided into two separate groups.

- **1. Geometric transformations**:
  - *Stroke attribute transformations*
    The cost of matching the two strokes f and g represents the distance between two of their nodes when the other nodes end_f, end_g are superimposed:
    $$diff_{st}(f,g,end_f,end_g) = \begin{cases} \infty \ \ if \ \ end_f,\ end_g \ \ incompatible \\ (l_f{}^2 + l_g{}^2 - 2l_f l_g cos(\alpha_f - \alpha_g)^{1/2} & otherwise \end{cases}$$
    Symbols l_f (l_g), α_f (α_g) represent a length and an angle of a stroke f (g), while $(\alpha_f - \alpha_g)$ represents the smallest positive angle between those strokes. To neglect the small variations of a length or angles, the above cost function is thresholded:
    $$cost_{st}(diff_{st})) = \begin{cases} 0 & diff_{st} < l_g/4 \\ diff_{st} - l_g/4 & otherwise \end{cases}$$
    where l_g is a length of a feature in a prototype.
- **2. Morphological transformations**:
  - *Arc attribute transformations*
    - *Matching two arcs*
      A cost of matching the two arcs is a sum of the three thresholded functions:
      $$cost(f,g,end_f,end_g) = \begin{cases} \infty \ \ if \ \ end_f \neq end_g \\ \frac{1}{2}diff_{arc}(\mid \phi_f - \phi_g \mid) + \\ \frac{1}{2}diff_{arc}(\mid \varphi_f - \varphi_g \mid) + cost_{st}(\mid l_f - l_g \mid) \\ otherwise \end{cases}$$
      where:
      $$diff_{arc}(x)) = \begin{cases} 0 & x < 1 \\ x - 1 & x \geq 1 \end{cases}$$
      φ and ϕ are a starting and an ending angles of an arc, $cost_{st}$ and other variables are defined as for the geometric transformations.
    - *Matching an arc into a stroke*
      An arc is transformed into its secant, and an associated cost of a transformation is a matching cost of a secant with a prototype stroke, plus a maximum distance from an arc to its secant. An arc in a prototype is never matched to a single stroke in a candidate.
  - *Straightening of strokes*
    A path of strokes can be "straighten"by creating a segment between a begining and an ending of a path, and an associated cost of a transformation is calculated as a cost of matching a new segment plus a maximum distance d from a path to a segment (e.g. see Fig. 10.11b)).
  - *Rewriting of strokes into arcs*
    A path of strokes can be rewritten into an arc. An angle between two strokes is smoothed into an arc with an associated cost proportional to its cosine (Fig. 10.11a)).

- **3. Leaving candidate features**
  If a feature appears in a candidate graph and it is not present in a prototype, it is deleted, and an associated cost is equal to a length of a feature.

- **4. Leaving prototype features**
  If a feature appears in the prototype but not in a candidate, and if it is mandatory, a matching possibility is not considered any further, unless a feature is optional in which case a cost is equal to a length of a feature .

- **5. Filling breakings with bridges**
  Gaps caused by feature breakings are identified on a candidate graph at a low level using a notion of relative neighborhood. Two points in a set are *relative neighbors* if they are as close to each other as they are not to any other point in a set. If two nodes in a candidate graph are relative neighbors, and one of them is an ending point, a marked segment called *bridge* is introduced into a graph (e.g. see Fig. 10.11d for an example of a bridge 1-2). The bridges identify the gaps but do not fill them, they only facilitate an association of the features for recognition. However, the use of bridges is a subject to penalty, there is a cost associated with their use which is equal to their length.

- **6. Division into separate parts** of glued multicomponent symbols
  Often, due to a noise, or not following rules by draftsmen, certain components of the multicomponents symbols are connected. If a sum of the nonmatched edges of a candidate graph is above a certain threshold, a candidate graph is divided (e.g. see Fig. 10.11c). A penalty of a division, is a constant, empirically set.

5. Graph distance measure

Let $H(p,c,c\prime) = \{(\alpha_j, f_j)\}$ for $j = 1,...,n$ be the homomorphism between graph p and subgraph c' of c.

Definition 10.5 *Transformation measure trans*(α, f) is a real function defined on all edge paths α on a candidate graph, and edges f on a prototype graph, based on the attributes of the features and the nodes.

A function *trans* is a quantitative measure of an amount of deformation that the homomorphism implies. Depending on a type of a second argument f, it measures a degree of straigtness of a path in a first argument, or how well it fits a simple arc. A value of a function *trans* is calculated in the following two stages.

1. A cost of transforming a path into a prototype feature (a stroke or an arc).
2. A cost resulting from a difference of attribute values between an obtained stroke or an arc and an appropriate prototype feature (as defined in the previous section).

Definition 10.6 The *cost* C_H of the homomorphism $H(p,c,c\prime) = \{(\alpha_j, f_j)\}$, for $j = 1,...,N$ between graph p and subgraph c' of c consists of the following three components:

$$C_H = \sum_{j=1}^{n} trans(\alpha_j, g_j) + \sum_{g \notin c' \cup B} length(g) + \sum_{j=1}^{n} \sum_{g \in \alpha_j \cap B} length(g).$$

The first component is a cost for a transformation of paths, the second is a cost for the features nonmatched in a candidate and the third is a cost for the use of bridges.

Definition 10.7 The *optimal homomorphism* $H^*(p,c)$ between graphs p and c is the one that minimizes a cost among subgraphs of c that are homomorphic to p:

$$H^*(p,c) = min_H\{cost(H(p,c,c')\}$$

Thus, the effect of comparing transformation costs when matching a graph of a particular candidate component c_i to a graph of a prototype component p_i is a minimization of the transformations required for an identification with a prototype p_i.

Definition 10.8 A *distance* of a candidate graph C composed of a set of graphs c_i of its individual components to a prototype graph $P(l)$ (representing a given class l) is defined as:

$$D(l) = \sum_{j=1}^{5} w_{l,j} c_{l,j} + r \times w_D$$

where $c_{l,j}$ represents a value of a cost type j, for $j = 1,...,5$ in the optimal homomorphisms of graphs of all candidate components c_i with respect to the prototype components $p_i(l)$, r is a number of division transformations, w_D is a constant equal to a cost of a single division operation (empirically set). Each cost type j describes a cost in one of the five groups of graph transformation described above (i.e. geometric, morhological, leaving candidate and prototype features, introducing bridges), $w_{l,j}$ is a weight constant which represents the relative importance of a transformation type j in a class l. The weights were set manually. The order of importance of the cost types for all the classes is the same as the order in which they were presented above. The least important is the geometrical cost and the most important one is the size of the nonmatched features in the prototype.

If N is a number of prototypes, an unknown graph C of a candidate is *classified* as belonging to a class k represeted by a prototype $P(k)$ if its distance $D(k)$ to a prototype graph $P(k)$ is a minimum among a set $\{D(l), l = 1,...,N\}$ of distances to all prototypes $P(l)$.

6. Inexact graph matching algorithm

A procedure for recognition of an unknown candidate symbol is composed of:

- Matching component graph of an unknown candidate symbol with component graphs of the prototype symbols.

- Matching graph of each candidate component with graphs of the prototype components, i.e. searching for optimal many-to-one homomorphisms between the appropriate pairs of graphs.

6.1. Matching component graphs

The procedure *MatchSymbol* (shown in Fig. 10.12) realizes the inexact matching between the component graphs K_P and K_C of the prototype and the candidate symbols, respectively, as well as between the graph P of the prototype and graph C of the candidate symbols. Graph P (C) is composed of graphs p_i (c_i) of individual prototype (candidate) components. Each graph $p_i \in P$ of the prototype component is matched with those unmatched graphs $c_i \in C$ of the candidate components for which the corresponding features in component graphs can be matched (i.e. a diffrence between their attributes is below certain threshold T), and then the candidate component with the lowest cost is chosen. Matching between the graphs p_i and c_i is calculated using the procedure *MatchGraphOfComponent* (described in the next subsection). If the length of the unmatched candidate component features is greater than the total component length times the certain threshold P (10%), then it is suspected that the candidate component is glued with another one, and it is tried to reconnect them (in the procedure *ComponentDivision*). The procedure *ComponentDivision* searches for all unmatched candidate features and creates new graphs for all sets of the connected unmatched candidate features.

6.2. Matching graphs of components

The algoritm *MatchGraphOfComponent* shown in Fig. 10.13 calculates the optimal homomorphism between two graphs: c_i of the candidate component and p_i of the prototype component. The search for this optimal many-to-one homomorphism can be formalized as a minimization of the possible graph transformations from a perfect matching. The candidate features are grouped in a very *dynamic* way, i.e. they are virtually transformed in a context of each prototype.

All *states* in the heuristic search for the optimal homomorphism except the goal state are the partial matchings between the graphs c_i and p_i. A state is considered a *goal state* if its corresponding matching cannot be expanded. The heuristic search is *initialized* with the matching of all possible features from the graph c_i of the candidate to the predefined and required feature from the graph p_i of the prototype, and as a result an *initial state* corresponding to the lowest cost matching is chosen. During initialization features are matched without taking into consideration specific endings, as it is impossible to determine nodes correspondence at the beginning. Normally, feature is always matched with respect to one of its nodes.

Generation of new states from the given state is performed by considering paths of different length (the possible extensions) in a procedure *FindExtensions* (shown in Fig. 10.14), starting with the feature and the node suggested by the *extension rule*, and


```
MatchSymbol(C, K_C, P, K_P).
Input: Candidate (prototype) graphs C (P)
    candidate (prototype) component graphs K_C (K_P)
Output: Matchings: K_C-K_P and C-P.
for all component p_i ∈ P do
  for all unmatched component c_i ∈ C do
  (*component graphs matching*)
    if the difference between attributes of the corresponding
    features f_{c_i} ∈ K_C and g_{p_i} ∈ K_P < T
  (* graphs of component matching*)
      MatchGraphOfComponent(c_i, p_i)
      Cost_i = TotalCost(c_i, p_i)
      if there are unmatched prototype components and
      length(unmatched c_i features) > P× length(c_i features)
        ComponentDivision(c_i)
      end if
    end if
  end for
  assign p_i a candidate component c_i with the lowest cost Cost_i
end for
```

Fig. 10.12. The procedure for matching component graphs of the candidate and the prototype symbols and graphs of their components

Rys. 10.12. Procedura porównywania grafów kandydata oraz wzorca

matching the node of each path and its adjacent features with the corresponding node and its adjacent features in the prototype, in a procedure *MatchNode* (shown in Fig. 10.17). The function *WriteState* writes them to the list of all considered states. The heuristic function *NextState* chooses the cheapest state (partial matching) from the set of all the calculated (in the procedure *FindExtensions*) possible next states , thus the algorithm uses a *cheapest-first strategy* that selects for processing the state with the lowest partial matching cost.

6.2.1. *Extension of a partial solution*

The procedure *FindExtensions* depicted in Fig. 10.14 calculates the list of possible extensions (i.e. the possible next states) of the partial matching between the graph c_i of the candidate component and the graph p_i of the prototype component. The function *Trans(path, feature)* in the procedure *FindExtensions* calculates a cost of transforming a list of features in a path into the prototype feature, each of which has an associated cost. The maximum length of a path is bounded by a predefined threshold times the expected size of the matched feature.

To extend a partial matching of nodes and features with a new pair of matched nodes,

MatchGraphOfComponent (c_i, p_i)
Input: Two graphs of components: candidate c_i and prototype p_i.
Output: The optimal homomorphism between the graphs c_i and p_i.
/* Initialization */
Let $g \in F_{p_i}$ be a non-optional feature.
for each feature $f \in F_{c_i}$
  $TotalCost = trans(\{f\}, g)$
  let $a \in J_{c_i}$ be one of the nodes adjacent to f
  let $b, c \in J_{p_i}$ be the nodes adjacent to g
  $H = MatchNode(a, b, f, g)$
  $WriteState(H)$
  $H = MatchNode(a, c, f, g)$
  $WriteState(H)$
end for
/* End of initialization */
repeat
  $H^* = NextState()$
  $FindExtensions(H^*)$
until *goal state* is found
return(H^*)

Fig. 10.13. The calculation of the optimal homomorphism H^* between the graphs of the candidate and the prototype components

Rys. 10.13. Obliczenie optymalnego homomorfizmu H^* pomiędzy grafami kandydata oraz wzorca

a set of already matched candidate features, but whose one node is not matched yet is considered. The algorithm considers paths of different lengths starting from 1-element path. In general, a next feature to expand a path is found using the *extension rule* when a branch point creates more than one alternative to continue a path. An extension is done in a context of a feature in a prototype, so there is always a knowledge of a direction that a path should follow.

Extension rule. If it is known that a path $f_1, ..., f_l$ in the candidate that ends in a node B_l (see Fig. 10.15 for an illustration) with the degree greater than one must be extended with a feature f_{l+1} adjacent to B_l, to improve the matching with feature g in the prototype, than it is possible to find optimally the next feature f_{l+1} adjacent to B_l that is also being matched to g, using attributes of the features of a *whole path* $f_1, ..., f_l, f_{l+1}$, and the attributes of g.

6.2.2. Node matching

The procedure $MatchNode(n_{c_i}, n_{p_i}, f_{c_i}, f_{p_i})$ shown in Fig. 10.17 calculates the best inexact matching between the features adjacent to two nodes $n_{c_i} \in J_{c_i}, n_{p_i} \in J_{p_i}$ belonging to

$FindExtensions(H^*)$.
Input: A partial matching H^*.
Output: A list of possible extensions of H^*.
let F'_{c_i} be a set of candidate component features with one not matched node.
if $F'_{c_i} = \emptyset$
  then return (*solution is found*)
let $f_1 \in F'_{c_i}$ be the feature already matched with $g \in F_{p_i}$
let $B_1 \in J_{c_i}$ is a non-matched node of f_1.
let $D \in J_{p_i}$ be the corresponding node to B_1 adjacent to g.
$TotalCost = trans(\{f_1\}, g)$
$m = MatchNode(B_1, D, f_1, g)$
$WriteState(H^* \cup m \cup \{(\{f_1\}, g)\}, TotalCost)$
$l = 0$
repeat
  $l = l + 1$
  $TotalCost = 0$
  for all continuation path h^k_{l+1} in B_l do (* $1 \le k \le degree(B_l)$*)
    $CostPath_k = trans(\{f_1, ..., f_l, h^k_{l+1}\}, g)$
  $CostPath_{k^*} = min_k\ (CostPath_k)$ (* k^* is the minimum cost path*)
  $f_{l+1} = h^{k^*}_{l+1}$
  $TotalCost = TotalCost + CostPath_{k^*}$
  $m = MatchNode(B_{l+1}, D, f_{l+1}, g)$
  $WriteState(H^* \cup m \cup \{(\{f_1, ..., f_{l+1}\}, g)\}, TotalCost)$
until $length(\{f_1, ..., f_{l+1}\}) > K * length(g)$

Fig. 10.14. Calculation of the list of the possible extensions of the partial matching

Rys. 10.14. Obliczenie listy możliwych rozszerzeń częściowego dopasowania

a graph c_i of a candidate component and graph p_i of a prototype component, respectively, given that two of those features $f_{c_i} \in F_{c_i}, f_{p_i} \in F_{p_i}$ are already matched. This procedure relies on the fact, that a matching should preserve the relative order of orientation of the features around two matched nodes, and the overall orientation of the features, since rotation near 90 degree are very unlikely. After sorting both list of features adjacent to nodes n_{c_i} and n_{p_i}, counterclockwise after f_{c_i} and f_{p_i}, respectively (by the procedure *sort*), the algorithm uses the procedure *MatchStrings* (shown in Fig. 10.18), which is the inexact string matching procedure, to find the best matching between these two sorted lists of features, l_{c_i} and l_{p_i}. The function $Cost(l_{c_i}, l_{p_i})$ calculates the cost of matching the features, and is based on the difference of feature directions in matched nodes. To calculate this direction we use *tangent* to a feature in a node being matched instead of a secant as in [270]. The functions *LeaveProt*, *LeaveCand* calculate the penalty for leaving the prototype and candidate feature, respectively.

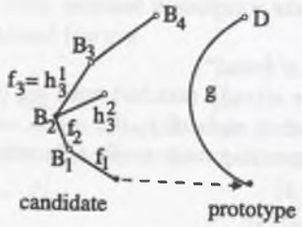


Fig. 10.15. The illustration of the proposed extension rule used in the branch point B_3 (the matched nodes are filled black). The correct feature h_3^1 is chosen as the possible continuation during matching to the prototype feature g. The local condition from ref. 270 fails choosing feature h_3^2

Rys. 10.15. Ilustracja zaproponowanej reguły rozszerzania w punkcie wielokrotnym $B3$, w wyniku której zostaje prawidłowo wybrana cecha h_3^1 jako możliwe rozszerzenie ścieżki dopasowanej do cechy prototypu g. Uzgodnione węzły są przedstawione na czarno. Zastosowanie lokalnego kryterium z ref. 270 powoduje nieprawidłowy wybór cechy h_3^2

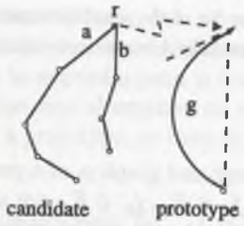


Fig. 10.16. The illustration of the proposed cost function used during nodes matching resulting in the correct matching of the candidate feature a into the prototype feature g. Using the cost function from ref. 270 results in the incorrect matching to the candidate feature b (the matched nodes are filled black)

Rys. 10.16. Ilustracja nowej funkcji kosztów podczas uzgadniania węzłów, w wyniku której zostają prawidłowo dopasowane cecha grafu prototypu g do cechy grafu kandydata a. Uzgodnione węzły są przedstawione na czarno. Zastosowanie funkcji z ref. 270 powoduje nieprawidłowe dopasowanie do cechy grafu kandydata b

$MatchNode(n_{c_i}, n_{p_i}, f_{c_i}, f_{p_i})$.
Input: nodes $n_{c_i} \in J_{c_i}$ and $n_{p_i} \in J_{p_i}$, features $f_{c_i} \in F_{c_i}$ and $f_{p_i} \in F_{p_i}$.
Output: Matching of the features adjacent to n_{c_i} and n_{p_i}, such that f_{c_i} is matched to f_{p_i}.
begin
  $l_{c_i} = sort(n_{c_i}, f_{c_i})$
  $l_{p_i} = sort(n_{p_i}, f_{p_i})$
  $MatchStrings(l_{c_i}, l_{p_i})$
end

Fig. 10.17. The calculation of the best inexact matching between the two nodes

Rys. 10.17. Obliczenie najlepszego niedokładnego dopasowania między dwoma węzłami

6.3. The complexity

To analyze the complexity of inexact graph matching algorithm we have to consider the complexity of algorithm $MatchNode$, and the number of times it is called by the search procedure. Assume that m is the maximum between the number of features in the prototype and in the candidate. Let d_G be the largest degree of a node in a graph G. A dynamic programming string matching algorithm finds a matching of the adjacent features of the two nodes in at most $d_P \times d_G$ steps. However, the search procedure can call this routine 2^m times in the worst case. Therefore, the algorithm is exponential in the number of features in a graph.

It is very important to note that the number of features or nodes in a symbol graph is usually small. Therefore, the real time of execution is relatively small.

7. Experimental results

Graph representations of prototypes of all geographic symbols from Polish Fundamental Land Map ([124]) were prepared as described in chapter 6. About 400 samples of geographic symbols were used to select the prototypes, and were processed repeatedly to revise the transformation rules and their associated costs. Tests were made on 500 samples of the scanned maps from which representations of geographic symbols were prepared in the similar way as for the prototypes. In the case of multicomponent symbols, their further segmentation was performed by interacting with recognition algorithm. Starting from the recognized candidate component, the region of the remaining potential components of a given symbol was created from which the component graph was constructed.

The classification results for the selected geographic symbols performed on 500 samples of maps are shown in Table 10.1. The minimum recognition rate obtained when varying the weights was 97.5%, and the maximum 98.9%. A sensitivity analysis showed that weights can be varied greatly without affecting significantly the recognition rate of symbols. The results are very encouraging, bearing in mind the wide variability of shapes

MatchStrings (l_{c_i}, l_{p_i}).
Input: sorted lists of features around node in candidate and prototype
Output: m - the best inexact matching between l_{c_i} and l_{p_i}

```
D[0,0] := 0
for l := 1 to length(l_{p_i}) do
   D[l,0] := LeaveProt(l_{p_i}(l)) + D[l-1,0];
for l := 1 to length(l_{c_i}) do
   D[0,l] := LeaveCand(l_{c_i}(l)) + D[0,l-1];
for l := 1 to length(l_{p_i}) do
begin
   m2 := LeaveProt(l_{p_i}(l))
   for j := 1 to length(l_{c_i}) do
      begin
      m1 := Cost(l_{c_i}(j), l_{p_i}(l))
      m3 := LeaveCand(l_{c_i}(j))
      D[l,j] := min(m1, m2, m3)
      if D[l,j] = m1 then set pointer from (l,j) to (l-1, j-1)
      if D[l,j] = m2 then set pointer from (l,j) to (l-1, j)
      if D[l,j] = m3 then set pointer from (l,j) to (l, j-1)
      end
end
m = D[length(l_{c_i}), length(l_{p_i})]
```

Fig. 10.18. The inexact matching of the two lists of features adjacent to the two nodes
Rys. 10.18. Niedokładne dopasowanie dwóch list cech przyległych do dwóch wierzchołków

and their different deformations like broken or touching symbols and complex structural deformations.

7.1. The example of 1-component symbol matching

Table 10.2 shows the detailed results of the inexact matching between the graphs of components of the candidate and the prototype of geographic symbol fountain, i.e. the first five categories of the transformation costs (geometric, morphological, bridges, leaving candidate and prototype features) of the optimal homomorphism between those graphs. The graph of component of the broken candidate fountain symbol is shown in Figure 10.8, while its prototype in Fig. 10.5. Elements in each row of the first two columns are the pairs *(prototype feature, candidate path)* forming the calculated optimal homomorphism. The path in the candidate (second column) is specified using feature labels according to Fig. 10.8, and their sequence corresponds to the sequence of their matching by the algorithm.

The computed distance (i.e. a weighted cost) between the graphs of components of the candidate and the prototype fountain symbols is equal to 0.571 and is the lowest

symbol	recog.
leafy tree	98,5%
conif. tree	97,8%
fountain	98,9%
bush	97,5%
grass	97,4%
well	98,1%
light	97,6%
cross	98,6%

Table 10.1. Experimental recognition results using the presented method for the selected geographic symbols

among the distances to all other geographic symbols. The main part of costs is the cost of introducing the bridges (0.2) and then the morphological cost which is connected with a transformation of a path into an arc, and with a difference between the attributes of the arcs. Also, not used parts of a candidate have some influence on the overall cost.

7.2. The example of multi-component symbol matching

Table 10.3 shows the detailed results of the inexact matching between the graphs of components of the candidate and the prototype of geographic symbol leavy tree, i.e. the first five categories of the transformation costs of the optimal homomorphism between those graphs. The meaning of columns and rows is the same as in the previous example. The component graphs of the glued candidate symbol are shown in Figure 10.9, while graphs of the prototype in Fig. 10.6. The computed distance (i.e. a weighted cost) between the graphs of components of the candidate and the prototype leavy tree symbols is equal to 2.320 and is the lowest among the distances to all other geographic symbols. The cost of division of the graph of component of the glued candidate symbol into the separate components is equal to 0.08. The most significant is the cost of unused (nonmatched) elements in the candidate. The morphological cost is composed mainly of the cost of transformation of paths into arcs. Because of the fact that strokes are not present the associated geometrical cost is 0.

prototype feature	candidate path	Cost categories				
		geom.	morpho.	leave-cand.	leave-prot.	bridge
1	4	0.0	0.0	2.433	0.0	0.0
0	5 6 7 8 9	0.0	0.033	1.588	0.0	0.2
2	12 13	0.0	0.081	1.184	0.0	0.0
4	3	0.027	0.0	1.091	0.0	0.0
3	10 11	0.0	0.009	0.762	0.0	0.0
5	0 1	0.0	0.045	0.395	0.0	0.0
6	14 15 16	0.0	0.009	0.119	0.0	0.0
Overall cost		0.027	0.177	0.20	0.119	0.2
Weights		1.0	1.0	2.0	0.8	1.0
Weighted cost		0.571				

Table 10.2. The results of matching between the graphs of components of the candidate fountain geographic symbol shown in Fig. 10.8 and its prototype in Fig. 10.5. The second column shows the sequence of matching the candidate features by the algorithm

prototype feature	candidate path	Cost categories				
		geom.	morpho.	leave-cand.	leave-prot.	bridge
prototype component: 0						
0	0 1 2	0.0	0.018	1.333	0.0	0.0
prototype component:1						
0	1 2	0.0	0.009	0.351	0.0	0.0
prototype component:2						
0	9 8 7	0.0	0.018	0.739	0.0	0.0
prototype component:3						
0	5 4	0.0	0.008	0.153	0.0	0.0
Overall cost		0.0	0.143	2.576	0.0	0.0
Weights		1.0	0.7	0.8	1.0	2.0
Weighted cost		2.320				

Table 10.3. The results of matching between the graphs of components of the candidate leafy tree geographic symbol shown in Fig. 10.9 and its prototype in Fig. 10.6. The second column shows the sequence of matching the candidate features by the algorithm. The cost of the division of the glued components is equal 0.08

Chapter 11

Syntactic classifier

1. Introduction

Among many groups of geographic objects are objects with a periodic structure, so-called *line objects*, or *1-dimensional texture objects*. Some examples of them are shown in Fig. 11.1. These line objects could be classified into different types, depending on their form, structure, thickness and other parameters. There may be connected and unconnected line objects. The unconnected ones are formed by repetition of one primitive element that could be a line segment, circle, or any other small primitive pattern. The simplest one is a dashed-line (Fig. 11.1a)), consisting of line segments separated by gaps which are used to symbolize for example land-use borders, edges of roads. More difficult examples are scarps (Fig. 11.1g-i)) and dams (Fig. 11.1j)). To recognize line objects which are characterized by a stable structure, the syntactic approach is a more suitable one.

This chapter presents a syntactic method for the recognition of line objects in map drawings (previously published in [318]). The presented method is also capable of recognition of other geographic objects like geographic symbols, and objects which have a characteristic, stable pattern, for example some sort of buildings.

The proposed recognition method is based on two extensions of a string grammar. The first extension is the usage of a *programmed grammar*, which offers a facility of explicitly programming the application order of productions. A class of context-free grammars (CFG), being the strongest class that we can construct efficient syntax analyzers for is too weak in practical pattern recognition applications. On the other hand, a class of context-sensitive grammars (CSG), which is stronger than CFG is hard to treat. Therefore, some enhanced models of context-free grammars have been proposed in the literature. One of the best known methodologies used for this purpose is the *programming* approach ([278]). A context-free *programmed grammar* ([278]) allows one to generate much wider class of languages than a 'pure' context-free grammar. However, the parsers for programmed grammar are based on a backtracking method, which means that their computational complexity is exponential.

The second proposed extension is the generalization of a string grammar into two dimensions by introducing special *relational symbols* for describing two-dimensionality.

There is also a completely different approach based on the usage of a representational formalism that is inherently more dimensional, for example *graph grammar*. But there are two problems which make a construction of a syntax analyzer for a graph grammar difficult. The first is in defining an unambiguous graph representation. The second is connected with the specific form of production of a graph grammar, especially with the embedding transformation. Some suggestions for solving this problem concerns graphs in a particular form ([298, 108]). For these reasons, we developed the higher-dimensional extension of a string grammar.

In the presented method, a grammar is used as an *extraction tool*, that means the grammar productions for object form a natural description of a strategy for its extraction.

The proposed parsing algorithm is a modification of that from [232] so as to avoid backtracking which results in polynomial complexity ($O(N) = N^3$). To handle specific kinds of errors in noisy and distorted objects in hand-drawn maps, the *error correcting mechanism* has been encoded in the parsing algorithm. The considered corrections are prolongation or shortening of primitives, change of their directions, or location. Due to proposed, special post-vectorization technique, the *linking neighboring primitives*, the developed parsing algorithm enable to recognize correctly *overlapped objects* in most situations.

The proposed recognition method is translation, scale and rotation invariant.

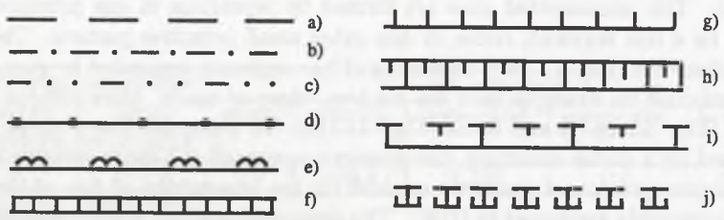


Fig. 11.1. Some examples of line geographic objects from instruction K-1: a)-c) different sort of borders, d),e) fences, f) stairs, g)-i) different sort of scarps, j) dam

Rys. 11.1. Przykłady obiektów liniowych z Zasadniczej Mapy Kraju: a)-c) różne typy granic, d),e) - ogrodzenia, f) schody, g)-i) różne rodzaje skarp, j) tama

2. Recognition of overlapped objects

The situation in which one object is superimposed on another one, occurs very often in maps. An idealized situation of overlapped objects is shown in Fig. 11.2 in which object $O1$ intersects object $O2$ at edge AB. This results in a division of an edge AB by two points $P1$ and $P2$ after vectorization procedure. Because primitive AB does not exist in such situation, subsequent parsing according to a grammar describing shape $O2$ fails, because of missing the required primitives.

To enable the correct recognition of overlapped objects we proposed the special post-vectorization procedure, the *linking neighboring line primitives* performed before recognition procedure starts. This post-vectorization is applied to those line primitives which are nearly collinear (with predefined tolerance), and their appropriate end-points are distant from each other not more than a predefined value. For example, in Fig. 11.2, linking primitives $AP1, P1P2, P2B$ into one primitive AB enables correct recognition of both objects.

In this way, in most situations with overlapped objects, the correct recognition by the proposed method is enabled. The linking neighboring line primitives into one primitive enables also to repair some classical distortions caused by thinning.

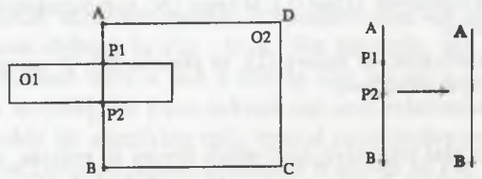


Fig. 11.2. Ilustration of linking neighboring line primitives

Rys. 11.2. Ilustracja zasady łączenia sąsiednich prymitywów

3. The grammar

Each shape is described by its *grammar*, and additionally distinguished one of its terminal symbols, called a *starting primitive*, which must be a *line* type primitive. All parameters characterizing the terminal symbols of a given grammar are then measured with respect to that starting primitive.

3.1. Primitives

Currently in the presented method three types of primitives are used: 1) *line* primitive (symbolized as L), 2) *break* primitive (B), 3) *included-in-line* primitive (I_L). It is assumed that each primitive, as well as each shape, has exactly two points called *head* (symbolized as H), and *tail* (symbolized as T) where it can be linked to other primitives. Thus, each primitive can be represented by a directed and labelled edge of a graph as shown in Fig. 11.3. Each primitive is characterized by a set of parameters.

The *line* primitive is denoted as $L(len_r, angle_r, tol_l, tol_a)$. The parameter len_r (e.g. see Fig. 11.3a)) denotes a length of a primitive which is always measured *relative* to a length of a starting primitive of a given shape. The $angle_r$ parameter denotes an angle between an initial primitive and a given one. This is a directed angle, with '+' values defined in the following way. A plus half-axis y of a local coordinate system with its

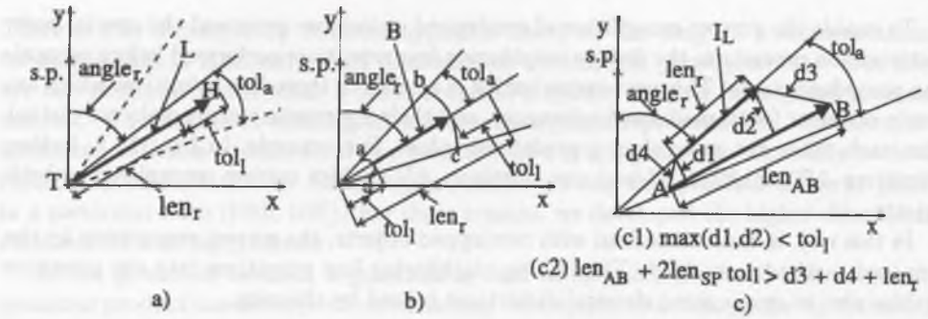


Fig. 11.3. Three types of primitives: a) line (L), b) break (B), c) included-in-line (I_L). (s.p. denotes a starting primitive)

Rys. 11.3. Trzy typy prymitywów: a) liniowy (L), b) przerwa (B), c) zawarty w liniowym (I). s.p. oznacza prymityw początkowy

origin at a tail of an initial primitive, and which covers it, rotates, so as to cover a given primitive. If the angle which it traces is less than 180° than the resulted angle has '+' sign, otherwise '-'. Two parameters specifying tolerances have been introduced: tol_l for a length tolerance, and tol_a for an angle tolerance.

The *break* primitive is denoted as $B(len_r, angle_r, tol_l, tol_a)$. The meaning of its parameters is shown in Fig. 11.3b). The existence of the break primitive means that in the area *abcd* in Fig. 11.3b) there is no head or tail of any other line primitive. A search for a break primitive is realized by searching the list of all existing in the image line primitives.

The *included-in-line* primitive is denoted as $I_L(len_r, angle_r, tol_l, tol_a)$ and illustrated in Fig. 11.3c). This type of primitive is a consequence of applying a procedure for linking neighboring line primitives, which is used in the presented method. An extraction of this primitive is realized by searching the list of all existing in the image line primitives. We say that included-in-line primitive I_L exists if: 1) there exists a line primitive in the image (for example AB in Fig. 11.3c)) with the same value of an $angle_r$ parameter (with up to tolerance tol_a), and 2) end-points of I_L primitive are not more than tol_l distant from that existing line primitive (condition c1) in Fig. 11.3c)). Additionally, to assure that a primitive I_L lies in a region occupied by an existing line primitive (AB in Fig. 11.3c)), the geometric constraint has been introduced (condition c2) in Fig. 11.3c)).

Because of using the relative length and angle of the primitives, such description will be invariant by scaling and rotation.

3.2. Relational symbols

Subshapes are assembled into more complex shapes by using the special type of relational symbols, the joining operators. These operators specify how two of the specific primitive points, its head or tail, have to be put in coincidence. In the presented picture description language, there are four ways of joining a pair of primitives. The three

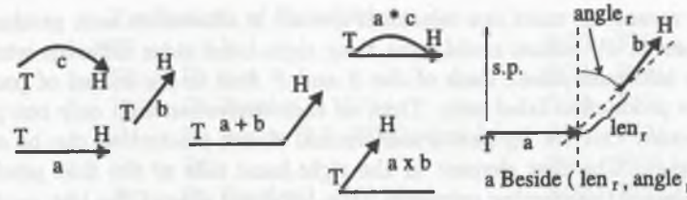


Fig. 11.4. Four type of relational symbols: +, x, *, Beside. s.p. denotes a starting primitive

Rys. 11.4. Cztery typy symboli relacyjnych: +, x, *, Beside. s.p. oznacza prymityw początkowy

binary relational symbols which are based on PDL ([297]), are denoted as: '+', '*', 'x'. These relational symbols, and their geometric interpretation are shown in Fig. 11.4. Let a, b, c be the primitives defined in Fig. 11.4. For example, the expression $a + b$ designates the shape obtained with a and b having the tail of b in coincidence with the head of a. Similarly to [232], we have defined our own relational symbol which allows more imprecise assembly by specifying only spatial relationship on the subshapes. This is operator $Beside(len_r, angle_r)$ whose semantics is shown in Fig. 11.4. The expression $a\ Beside(len_r, angle_r)\ b$ denotes the shape obtained from the primitives a and b in the following way. If we translate a starting primitive to a head of a primitive a, as shown in Fig. 11.4, then a tail of a primitive b lies at angle $angle_r$ (relative to the starting primitive), at a distance len_r. The tail of a primitive b can be located in a circle of radius specified by tol_l parameter of a primitive b.

Each shape resulting from linking the two primitives by means of the defined relational symbols, has a tail and a head, too as shown in Fig. 11.4. A tail of a resulting shape is always a tail of a second operand, and a head of a first. Thus, we can extend the four operations to higher level structures, i.e. we can use more complex expressions as arguments of the operators.

It is assumed that a tolerance for a location of a head or tail of a primitive during its extraction from an input image, is equal to tol_l. That means, a head or tail of a primitive (dependent of a sort of relational symbol used), is searched in a circle of a radius specified by tol_l parameter of a given primitive.

3.3. Production rules

The set P of all productions defining a given grammar G (2D-shape) consists of a union of two subsets: P_B and P_T (e.g. see Table 11.2 for an example of a grammar of geographic object scarp). The productions in P_B subset define (together with other components N, T, S, J) a programmed grammar. The productions in the P_T subset define the possible subshapes S_i of a shape S. Each such subshape S_i has its associated subset P_{S_i} of productions in P_T subset, and together with other components ($N' \subset N, T' \subset T, S_i$) defines its LL(0)-type grammar.

We put several restrictions on admissible form of productions. Each production has the following form: $A \rightarrow B\ op\ C$, where $A \in N$, $op \in \{+, \times, *, Besides()\}$, $B, C \in$

$N \cup T$. That means, at most one relational symbol is allowed in each production. No two productions in P_T subset could have their right-hand sides different while having identical their left-hand sides. Each of the S and F field in P_B subset of productions, consists of one production label only. Thus, at each derivation step only one possibility in derivation exist. Only in P_B part a nonterminal A in a production can be either left- or right-recursive. The first element in the right-hand side of the first production in P_B subset is always the starting primitive. The left-hand side of the last production in P_B subset is the starting symbol of the grammar, and represents the final goal of the recognition.

The set P of productions of a grammar of a geographic symbol scarp is given in Table 11.2, while the set of its terminals in Table 11.1. Nonterminal symbols are marked with big letters, while terminals with small ones. The presented values of parameters describing terminals, agree with K-1 instruction ([124]) for a map scale 1:500.

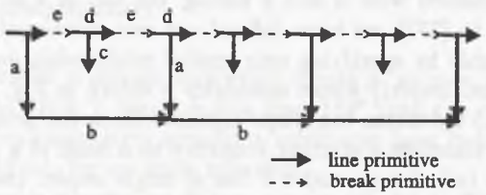


Fig. 11.5. The example of the prototype of scarp geographic symbol (a - e are the terminals)

Rys. 11.5. Przykład wzorcowego symbolu skarpy (a-e oznaczają symbole terminalne

a	$L(1, 0, 0.4, 20)$
b	$I_L(1.8, -90, 0.4, 20)$
c	$L(0.4, 0, 0.4, 20)$
d	$I_L(0.4, -90, 0.4, 20)$
e	$B(0.4, -90, 0.4, 20)$

Table 11.1. The set of terminals of a scarp grammar

4. The recognition algorithm

4.1. The parsing algorithm

The developed parsing algorithm consists of three cooperating parsing procedures: *Bottom-up, Head-to-tail* and *Tail-to-head.*

The *Bottom-up* procedure which is the main procedure, proceeds bottom-up, taking productions from the P_B subset, as specified by labels in the S and F fields, and as

	Productions of P_B subset:		
1.	P1 → a x b	S(2)	F(0)
2.	P2 → P1 + a	S(3)	F(0)
3.	P3 → P2 Beside (1, 110) c	S(4)	F(0)
4.	P4 → P3 Beside (0.9, 90) Dashed-Line	S(5)	F(0)
5.	P5 → P4	S(6)	F(0)
6.	P5 → P5 Beside (1, 180) Mesh	S(6)	F(7)
7.	Scarp → P5	S(0)	F(0)
	Productions of P_T subset:		
	Mesh → b + P6		
	P6 → a Beside (0.9, 110) P7		
	P7 → c Beside (1, 90) Dashed-Line		
	Dashed-Line → d + L1		
	L1 → e + L2		
	L2 → d + L3		
	L3 → e + d		

Table 11.2. The set P of productions of a scarp grammar

a result it returns a recognized shape with its tail and head. This procedure which is detailed in Fig. 11.7, has five input parameters: 1) the goal G of the recognition, for example *Scarp*, 2) the subpattern A already recognized, 3) T, 4) H, the location of the tail and head of A, respectively, 5) the list R of the primitives in a subimage which has to be considered for this parsing.

The *Tail-to-head* procedure, detailed in Fig. 11.8, proceeds in a top-down manner. It starts knowing the tail of the pattern to be analyzed, and returns its head as a result. It has three input parameters: 1) the pattern A to be parsed, 2) the location T of its tail, 3) the current list R of primitives on which the analysis is to be performed, and one output parameter, beeing the location H of the recognized shape A. The *Head-to-tail* procedure is "symmetric"to the procedure *Tail-to-head*, that means starts with the head and returns the tail of the analyzed pattern.

In the case of scarp recognition (e.g. see Table 11.2 for production set of a scarp grammar), the parsing starts with a calling to the *Bottom-up* procedure with the *Scarp* as a value of G parameter, a as a value of A, T and H being the head and tail of a, respectively, and R beeing the list of all primitives extracted from an input image. Parsing then proceeds alternatively in a bottom-up and top-down manner. For example, when a $P3$ subshape is recognized by successful application of the production 3, the parsing algorithm chooses production label 4 from S field of the 3-rd production (the only one). The parsing algorithm proceeds then in a top-down manner in order to recognize the missing part *Dashed-Line*, in the right-hand side of 4-th production, and so complete the subpattern $P4$. This is accomplished by calling to the procedure *Tail-to-head* with the *Dashed-Line* as the value of A parameter, the value resulting from parameters of relational

symbol Beside as a value of T parameter, and R being the current list of primitives to be analyzed. As a result, the head of the recognized subshape *Dashed-Line* is returned. During this top-down parsing, only productions from the P_T subset of production set P are applied.

4.2. Error-correction

Objects in real hand-drawn maps are often distorted as shown in Fig. 11.6. To be able to explicitly manage different kind of errors introduced for handwritten map, the error-correcting parsing mechanism has been encoded in the parsing algorithm, but in a more pragmatic way than the systematic techniques described in chapter 1.

The corrections we are considering, the following types of corrections:

- Prolongation or shortening of primitives in order to have exact coincidence.
- Change of direction.
- Change of the location of a primitive head or tail.

These corrections are accomplished by means of the appropriate values of *tolerance parameters* of primitives. These tolerance values determine what an admissible corrected input is.

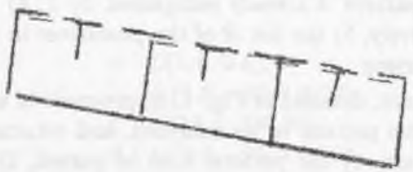


Fig. 11.6. The result of correct recognition of a distorted scarp. Raster scarp shown in gray, scarp generated by a grammar in black

Rys. 11.6. Wynik prawidłowego rozpoznania zdeformowanej skarpy. Skarpa raster pokazana na szaro, wygenerowana przez gramatykę na czarno

4.3. Computational complexity

We now assess the time complexity of the parsing algorithm. The input to the parsing algorithm consists of N primitives which have been extracted from the input image. The total time complexity of analysis of an input consisting of N primitives is equal to $n_1(N) \times O_1(N) \times N$, where $n_1(N)$ is the number of iterations of *while*-loop in *Bottom-up* procedure, $O_1(N)$ is the time complexity of *Tail-to-head* (*Head-to-tail*) procedure. The multiplication by N results from the fact that in the worst case, all the N primitives must be checked to find the initial primitive.

The number $n_1(N)$ of iterations of *while*-loop in *Bottom-up* procedure is proportional to N ($n_1(N) \sim N$). This holds because after one successful application of a production, at least one primitive has to be used (it is then throw out from the list of primitives). After unsuccessful applying of a production, the algorithm ends.

The computational complexity $O_1(N)$ of the recursive *Tail-to-head* procedure, is equal to the number of nodes in a tree constructed from its calls. Each call to the *Tail-to-head* procedure makes one derivation step as it applies one production from P_T subset. This number is fixed for a given grammar. Time complexity of application of one production $\sim N$. Thus, $O_1(N) \sim N$.

The total time complexity of the parsing procedure $O(N) = N \times N \times N = N^3$.

The proposed analysis strategy descrined in chapter 8 allows for a computation of a region of interest in which the parsing will be performed, which drastically reduces the number of candidate primitives which must be checked to make a derivation. Without such a method the system would waste its time trying to match what is actually intersections between lines, too large areas of the map, or even subgraphs containing only a part of an object. The parameter R, denoting the list of primitives to be parsed in a procedure *Bottom-up*, is a list of primitives in a computed region of interest. Thus, the number N, being the number of primitives is not very high.

It is obvious, that the key point is teh right choice of a starting primitive, especially in case of a rich library of prototype objects. To speed up a recognition rate, a method for an "intelligent"search of a starting primitive should be provided.

```
Bottom-up (
    G cc goal of the recognition,
    A cc subpattern already recognized,
    T cc tail of A;
    H cc head of A;
    R cc list of primitives to be parsed);
begin
  T1 = T
  H1 = H
  while A <> G do cc goal is not reached
  choice of a next production B → m
  if production is recursive
  then set IsRec = 1 else IsRec = 0
  case
    the production is B → A + C then
        Tail-to-Head (C, H1, P, R)
        H1 = P;
    the production is B → A × C then
        Tail-to-Head (C, T1, P, R)
        H1 = P;
    the production is B → A * C then
        Tail-to-Head (C, T1, P, R)
        if P <> H1 then Error = 1 ;
        cc coincidence of heads must me verified;
    the production is B → A Beside (length, angle) C then
        T2 = value resulting from parameters of Beside
        Tail-to-head (C, T2, P, R)
        H1 = P;
  end case
  if Error = 1 then
    if IsRec = 0 then break
  T = T1
  H = H1
  end while;
end;
```

Fig. 11.7. The bottom-up parsing procedure

Rys. 11.7. Procedura mieszanej analizy syntaktycznej typu z dołu do góry


```
Tail-to-Head (
    A cc subpattern to be parsed from tail to head,
    T cc a tail of A;
    H cc a head of A;
    R cc list of primitives to be parsed);
begin
  if A is a primitive then
    if there is an occurrence a of A in R whose tail coincides with T
    then H = head(a) else error = 1
  choice of a production A → m
  case
    the production is A → B + C then
        Tail-to-Head (B, T, H1, R)
        T1 = H1
        Tail-to-Head (C, T1, H2, R)
        H = H2;
    the production is A → B × C then
        Tail-to-Head (B, T, H1, R)
        Tail-to-Head (C, T, H2, R)
        H = H2;
    the production is A → B * C then
        Tail-to-Head (B, T, H1, R)
        Tail-to-Head (C, T, H2, R)
        if H1 <> H2 then error = 1
          else H = H1;
    the production is A → B Beside (length, angle) C then
        Tail-to-Head (B, T, H1, R)
        T2 = value resulting from parameters of Beside
        Tail-to-Head (C, T2, H2, R)
        H = H2;
    the production is A → B then
        Tail-to-Head (B, T, H1, R)
        H = H1;
  end case
end;
```

Fig. 11.8. The top-down parsing procedure

Rys. 11.8. Procedura mieszanej analizy syntaktycznej typu z góry do dołu

Chapter 12

Discussion and conclusions

The *first goal* of the research described in this thesis was the development of a flexible, and generally applicable reasoning framework for geographic map image analysis which allows to solve all the mentioned problems in map image analysis. The *second goal* was to augment the proposed framework with structural classifiers of graphics and text components of a map, and to overcome the border problems of structural pattern recognition.

From the results described in this thesis we conclude that the goals have been reached. The proposed, general reasoning framework for knowledge-based geographic map image analysis, together with the developed, and properly integrated within the framework structural classifiers, responds to all the problems mentioned in Chapter 1. The developed classifiers, being the structural pattern recognition modules, are integrated into a whole analysis process by the suitable intelligent analysis strategy described by the image analysis flow scheme. The proposed approach to image analysis enables for increased flexibilty: the image analysis flow scheme is independent of the application, the map model, dependent on the application, can be used for another ones only after small adjustments. The framework proposed in this thesis is a promising and flexible approach to map image analysis in particular, and graphics documents in general.

From practical experience it follows that the map conversion module *MAPIN*, being the implementation of the proposed framework, which replaced the manual input into the Integrated Spatial Organization System SIT-y ([260]), correctly converts about 70% of the information in the maps (situational and land register layers) into ArcInfo/ArcView, leaving the remaining 30% to the operator. The system capable of converting the entire map but with an error rate of 10% is not practical as all results have to be verified and corrected manually. Thus, *MAPIN* enables to achieve a considerable speed-up of the overall conversion process. The objects are recognized reliably but at the cost of a significant rejection rate. The fully automatic analysis of maps is therefore not yet feasible, however, integration within a semi-automatic environment should be possible.

In what follows, we summarize the characteristics of these proposed methods, solutions to the mentioned problems, and discuss the remaining problems.

1. Image analysis flow scheme

The proposed image analysis flow scheme enables to overcome the following problems:

- *Reasoning with incomplete information* (Problem 1)
  The proposed, non-monotonic reasoning method, the *complementary reasoning* ([321]), can be initiated at any time based on the incomplete information thus far obtained, and the description of the map can be constructed even, if some information is missing. This is due to the process of active hypothesis generation which compensates for insufficient input information.

  The logical framework of the complementary reasoning is based on a modified Poole's approach to non-monotonic reasoning ([258]). The complementary reasoning does not monotonically deduces theorems from a fixed logical theory, instead, it attempts to use deduction to construct consistent theories (i.e. compatible explanations) for which the relations are logical consequences.

  Although, complementary reasoning proved to be successful, there are still some limitations. This reasoning is based on a concept that each object has one or more relations with other objects. In the *geodesic base* layer of PFLM (contour lines and different altitude points), this concept may be only partly valid, as the altitude points are distributed more or less randomly over the map, and do not share a clear spatial relation with other object types.

- *Control strategy* (Problem 2)
  The proposed, mixed control strategy ([321]) is the flexible integration of bottom-up and top-down strategies. It does not require ad hoc control knowledge about when to activate one analysis or the other, instead, the top-down analysis is activated dynamically, depending on the contents of the current scenario under analysis. Combination of the domain-specific knowledge with the derived results renders the possibility for selecting the most promising goals to control and guide the analysis process. Moreover, the properly chosen weight values assigned to links enable to control quantitatively the analysis process. Reasoning based on cost values improves the robustness of the analysis. The analysis can be focused on the most reliable partial interpretation, leaving the others unanalyzed.

  The performance of complementary reasoning will be improved by introducing numerical reliability computation instead of the very simple scenario cost computation scheme described in chapter 8. However, this requires an augmentation of ordinary probability models (for example Dempster and Shaffer ([70, 289]), as they assume a fixed set of entities for probability computation. There is also need to represent the reliability of pieces of evidence by numerical value. Most systems have used ad hoc reliability assignment rules. We need well defined foundations to transform property value of a piece of evidence into a numerical reliability value.

- *Contextual layer separation, interaction of segmentation with recognition, selection of region of interest* (Problems 3, 8, 9).

  To cope with the above problems, we proposed a *context-based* segmentation and vectorization, based on the *intelligent map analysis strategy* ([321]) which uses *a priori knowledge* (the map model) about the meaning of the extracted components, that means algorithms have conceptual model of what they are looking for. This strategy allows the feature extraction process to be continuously refer to the image data, and to be dynamically dependent upon the current context (the analyzed scenario). This is important, as introduction of context (determined by the current scenario being analyzed) improves drastically the quality of segmentation and vectorization. By interpreting the pixels in terms of the possible hypotheses, the intelligent analysis strategy can compensate for the imperfection of segmentation: a description of a partially overlapped object can be completed by introducing hypotheses for the overlapped parts, and hypotheses are used as goals to guide the top-down search (or segmentation) for missing objects. Recognition techniques that depend on having the object of interest isolated from the image, may fail because an error in the isolation will preclude success in the subsequent processing steps.

  To cope with noise and structural deformations, all the proposed and implemented pattern recognition modules are error-correcting ones. Each of the proposed pattern recognition module is not treated as a standalone tool, or a "black box", instead, the recognition module is properly inserted in its context, and interacts correctly and efficiently with its surrounding modules, that means, both with the low-level image segmentation (Vector Graph construction module), and with higher level analysis module (Reasoning Engine). Thus, the developed within the framework image analysis flow scheme, based on intelligent analysis strategy, enables for proper integration of the proposed pattern recognition modules.

  The intelligent analysis strategy allows also for precise selection a region of interest while searching for potential candidate structures (graphs, trees), or primitive pattern constituents to be matched with the models, or used in construction of a derivation by a parser. Parameters of such a region are determined based on the geometry, and location of the object which generated a hypothesis.

- *Reliable analysis* (Problem 4)
  The accumulation of partial evidence, the idea taken from [221], increases the reliability of the analysis. We extended this idea by the usage of relations from a domain model to control quantitatively the accumulation process, as well as by introducing compatibility examination in each interpretation cycle to detect contextual inconsistences as early as possible.

In the proposed framework, many hypotheses are generated to complement insufficiency of input information and increase the reliability. From a computational point of view, however, generation of such multiple descriptions need large memory space and much computation time. We used several mechanisms to reduce the number of pieces of

evidence in the database, for example hypothesis generation by hypothesis is prohibited, verified/refuted hypothesis are immediately removed. The global selection method in interpretation cycle also helps to increase the efficiency. Thus, the computational complexity itself is not great. But, as reasoning scheme in the framework includes much potential paralelism, parallel processing like parallel applicaton of rules, or compatibility examination, will be useful to improve the speed of the analysis.

2. Map model

As a map model, we proposed hybrid knowledge representation scheme, the hybrid semantic network ([321]), which is a combination of different knowledge representation formalisms: the object-oriented computation paradigm, pure semantic network, frame system and rules. The proposed hybrid semantic network, together with the developed image analysis flow scheme provides a general reasoning framework which allows the *integration* of a variety of pieces of knowledge, and reasoning methods into one consistent framework. Object instances are not static data structures to record properties of recognized objects, but active reasoning agents which perform reasoning about their surrounding environments. The knowledge required for an object instance to perform reasoning is represented by rules which are stored in corresponding object class: all instances of the same object class share the same set of rules. In this sense, the knowlege representation can be considered as a mixture of frame and rules. Moreover, in order to realize flexible reasoning, rules are augumented by introducing *Hypothesis* and *Function* fields which play a crucial role in coordinating reasoning activities by individual object instances. Execution of tasks is synchronized through their corresponding hypotheses. The object-oriented approach, through encapsulation and modularity enables to improve the consistency, understandability, and maintainability of a evidence pool.

3. Object detection methods

For the different graphics/text components of map drawing we have developed within the framework *detectors* (*classifiers*), each of which is based on the appropriate structural model of a shape of *graphics/text* components. Various models have been used to describe objects, those based on local features like attributed relational graphs, a string grammar together with special operators for describing two-dimensionality, as well as global, qualitative, quasi-topological features describing the structure of a general 2D-curve.

We have developed the following *classifiers*:

1. Polygon classifier.

2. Classifier based on relational matching of 2D-curve structural descriptions ([314, 317]).
3. Classifier based on inexact graph matching ([323]).
4. Syntactic detector that relies on error-correcting parsing ([318]).

The proposed structural models of graphics and text elements in a map drawing enable to overcome the following problems, intrinsic to structural pattern recognition.

- *Limited descriptive power of a grammar* (Problem 5)
  The developed *syntactic detector* ([318]) based on *programmed, higher-dimensional extension of a string grammar* with special operators for describing two dimensionality, allows to expand the generative power of a pure CFG grammar. The parsing algorithm for the proposed grammar has polynomial complexity ($O(n^3)$).

- *Errorneous patterns* (Problem 6)
  The usage of a *programmed grammar* in the proposed syntactic detector ([318]), by providing a facility for explicitly programming the application order of productions enables to overcome the problem of deriving structures that do not belong to the pattern class (errorneous ones).

- *High number of prototypes per class and diversity of document's structures* (Problem 7)
  Two solutions to the above problems have been proposed within the framework. First, we proposed the *incorporation* to a structural prototype, a *graph*, a *procedural information* which includes *algorithmic*, and *rule-based mechanism* in the nodes and edges of a graph, and extend it with ISA and PO relations. The resulted, augmented prototype graph structure, is the *hybrid semantic network* ([321]). The nodes in a prototype graph are interpreted as concepts of a semantic network, with node attributes and labels corresponding to particular slots of a frame in a semantic network. In a structural prototype there are problem-dependent relations between nodes. They correspond exactly with problem dependent relations between the concepts in a semantic network. The proposed augmentation of a graph allows for a great number of different reasoning procedures which are principally useful for semantic networks, including matching (being the only feasible type of inference in graphs), as a special case.

  For example, the variety of prototypes which are required to describe different possible variations of *ROAD* and correlated objects from a map model shown in Fig. 7.1 in chapter 7, has been reduced to one, by incorporating suitable algorithms and rule components (with applicability predicates) in nodes and edges of a graph.

  Secondly, we proposed a method which reduces a number of required prototypes per class, by the proposed, inexact, many-to-one graph matching, that manages dynamically the relative distortion between a candidate shape and its prototype,

and enables flexible matching between components and a flexible grouping of the individual comonents to be matched. This has been implemented in a structural classifier for hand-printed geographic symbols, as well as characters ([323]).

As the geographic objects are handprinted, different shape deformations are inevitable, the developed structural recognition methods are error-correcting ones.

We have proposed the strategy for recognition of **overlapping shapes**. Due to special post-vectorization technique, linking neighboring primitives, the syntactic detector ([318]) together with the intelligent map analysis strategy enables correct recognition of overlapping objects in map drawings.

4. Model-based vectorization

We have proposed the efficient vector representation of an image in the form of the Vector Graph ([319]). This vector representation is achieved by two different model-based vectorization schemes, each of which is appropriate for vectorization of different lines:

- For short lines, representing symbols and text we proposed vectorized thinning ([315]). In this approach features are extracted based on detection of regular and singular features which allows to avoid the major disadvantage of all thinning based vectorization methods: distortion at junctions. Finding correct position of junction is accomplished by fitting to the *model of ideal junction*.

- For long lines, representing geographic object edges, we proposed 2 variants of mixed vectorization. The full variant (a modification of [166, 168]) is based on locating precisely all dominant points first, then fitting the detected potential curve segments to conic curves (here, being a *model* for curve lines), and polygonal approximation of the remaining portion of a curve. Such a combination of both feature detection methods, first dominant points detection, then polygonal approximation, allows to avoid the main disadvantage of polygonal approximation, that some feature points may be missed or shift. As the full variant is computationaly expensive, it should be used only when the results need to be of high quality. In the remaining cases, the limited variant, without fitting to a conic curve will be sufficient. We suggest to use the post-vectorization based on a *model of ideal result*, to improve the quality of the constructed line objects.

Appendix A

Layers of Polish Fundamental Land Map

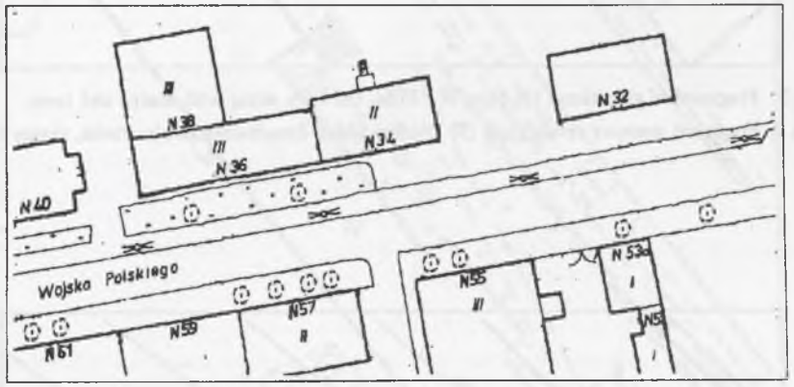


Fig. A.1. Fragment of situational (S) layer of PFLM: the road system (pavements with trees, green belts) along with buildings

Rys. A.1. Fragment warstwy sytuacyjnej (S) Polskiej Mapy Zasadniczej Kraju: system dróg wraz z budynkami (chodniki z drzewami, pasy zieleni)

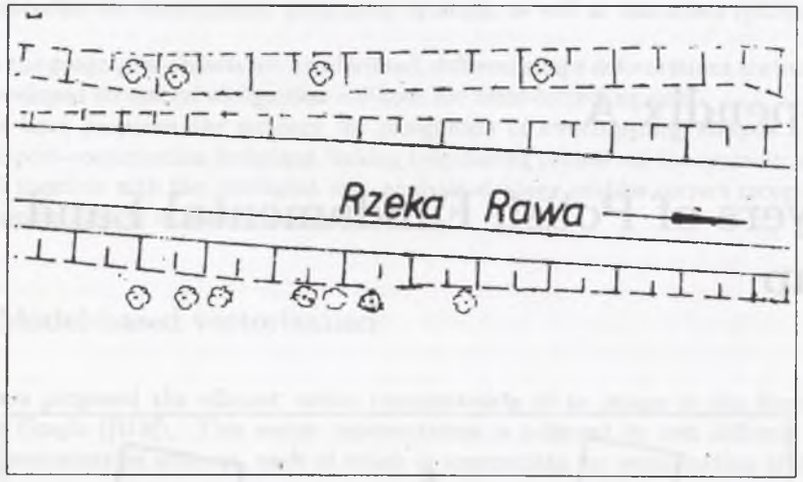


Fig. A.2. Fragment of situational (S) layer of PFLM: the river along with scarps and trees

Rys. A.2. Fragment warstwy sytuacyjnej (S) Polskiej Mapy Zasadniczej Kraju: rzeka, skarpy i drzewa

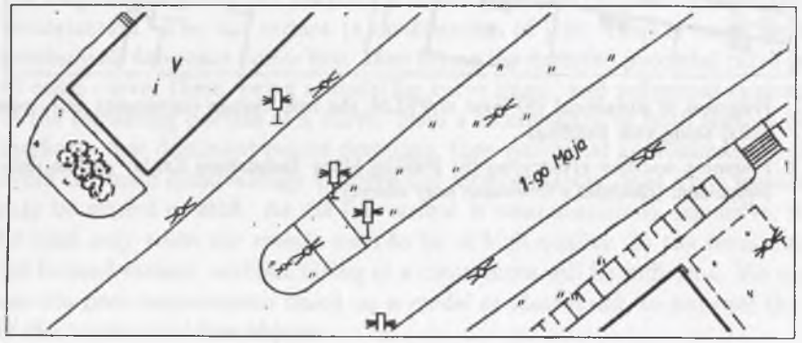


Fig. A.3. Fragment of situational (S) layer of PFLM: buildings, the road system composed of road, pavements, green squares and island, Scarp is also visible

Rys. A.3. Fragment warstwy sytuacyjnej (S) Polskiej Mapy Zasadniczej Kraju: budynki, system dróg złożony z drogi, chodników, zieleńców oraz wysepki, widoczna skarpa

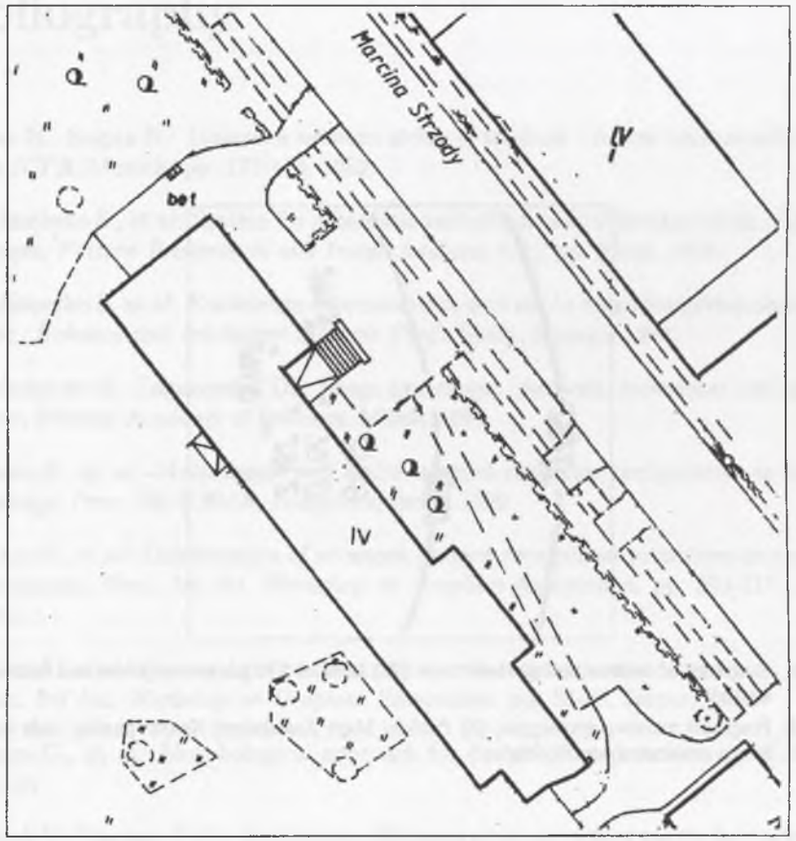


Fig. A.4. Fragment of situational (S) layer of PFLM: the road system, buildings with terrace, stairs, green squares. Green fence along pavement also visible

Rys. A.4. Fragment warstwy sytuacyjnej (S) Polskiej Mapy Zasadniczej Kraju: system dróg, budynki, pokazane tarasy, schody, trawniki. Widoczny żywopłot wzdłuż chodnika

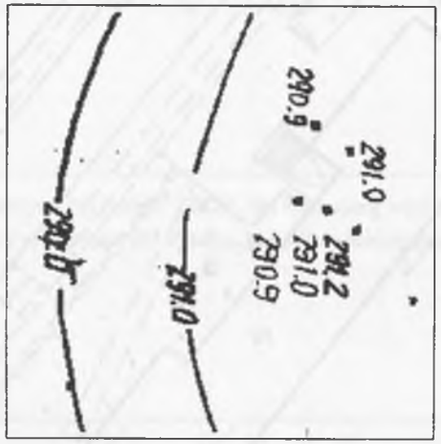


Fig. A.5. Fragment of isolines and geodesic base (W) layer of PFLM: stable points and isolines with altitudes

Rys. A.5. Fragment warstwy sytuacyjnej (S) Polskiej Mapy Zasadniczej Kraju: punkty stałe oraz izolinie z oznaczoną wysokością

Bibliography

[1] Abel K., Sugita N.: Distances between string of symbols - review and remarks, *Proc. 6th ICPR*, Munich, pp. 172-174, 1982.

[2] Ablameyko S., et al: System for automatic vectorization and interpretation of graphic images, *Pattern Recognition and Image Analysis*, 3(1), pp. 39-52, 1993.

[3] Ablameyko S. et al: Knowledge representation and use in map interpretation system, *Proc. Robotics and Intelligent Systems Conf.*, Spain, Malaga, 1994.

[4] Ablameyko S., Lagunowski D.: *Image processing: methods, techniques and applications*, Belarus Academy of Sciences, Minsk, 1999.

[5] Adam S., et al: Multi-scaled and multi-oriented character recognition: an original strategy, *Proc. 5th ICDAR*, Bangalore, India, 1999.

[6] Adam S., et al: Combination of invariant pattern recognition primitives on technical documents, *Proc. 3rd Int. Workshop on Graphics Recognition*, pp. 203-210, Jaipur, India.

[7] Adam S., et al: Processing of connected shapes in raster-to-vector conversion process, *Proc. 3rd Int. Workshop on Graphics Recognition*, pp. 38-45, Jaipur, India.

[8] Agam G., et al: Morphological approach for dashed lines detection, in: [180], pp. 92-104.

[9] Aho A.V. Peterson T.G.: A minimum-distance, error-correcting parser for contect-free languages, *SIAM J. Comput*, 1, pp. 305-312, 1972.

[10] Aho A.V.: Algorithms for finding patterns in strings, in: [202], pp. 255-300.

[11] Aho A.V., Ullman J.D.: *The theory of parsing, translation and compiling, v.1: Parsing*, Prentice-Hall, Englewood Cliffs, N.J., 1972.

[12] Almohamad H.A., Duffuaa S.O.: A linear programming approach for weighted graph matching problem, *IEEE Trans. PAMI*, 14(5), pp. 522-525, 1993.

[13] Andenberg M.R.: *Cluster analysis for applications*, Academic Press, N.Y., 1973.

[14] Anderson R.H.: Syntax-directed recognition of handprinted two-dimensional mathematics, in: M. Klerer, J. Reinfelds, *Interactive systems for experimental applied mathematics*, Academic Press, N.Y., 1968.

[15] Anderson M., Bezdek J.: Curvature and tangential deflection of discrete arcs, *IEEE Trans. PAMI*, 5(6), pp. 27-40, 1984.

[16] Anegawa M., et al: A system for recognizing numeric strings from topographical maps, *Proc. 3rd ICDAR*, Montreal, pp. 944-947, 1995.

[17] Antoine D.: CIPLAN: A model-based system with original features for understanding french plats, *Proc. 1st ICDAR*, St. Malo, France, pp. 647-655, 1991.

[18] Antoine D., et al: Analysis of technical documents: the REDRAW system, in: [28], pp. 385-402.

[19] Arabie P., Hubert L.: *Clustering and classification*, World Scientific, Singapore, 1992.

[20] Arai H.S. et al: Intelligent interactive map recognition using neural network, *Proc. 2nd ICDAR*, Tsukuba, Japan, 922-925, 1993.

[21] Arcelli C.: Pattern thinning by contour tracing, *CGIP*, 17(1), pp. 130-144, 1981.

[22] Arcelli C., Sanniti di Baja G.: A width independent fast thinning algorithm, *IEEE Trans. PAMI*, 7(4), pp. 463-474, 1985.

[23] Arcelli C., Sanniti di Baja G.: A one-pass two-operations process to detect the skeletal pixel on a 4-distance transform, *IEEE Trans. PAMI*, 11(4), pp. 411-414, 1989.

[24] Arias J., Kasturi R.: Recognition of graphical objects for intelligent interpretation of line drawings, in: Arcelli C. et al, *Aspects of visual form processing*, World Scientific, Singapore, pp. 11-31, 1994.

[25] Arias J.F., et al: Interpretation of telephone system manhole drawings, *Pattern Recognition Letters*, 16(3), pp. 355-369, 1995.

[26] Asada H., Brady M.: The curvature primal sketch, *IEEE Trans. PAMI*, 8(1), pp.1-14, 1986.

[27] Attneave F.: Some informational aspects of visual perception, *Psychol. Rev.*, 61(3), pp. 183-193, 1954.

[28] Baird H., Bunke H., Yamamoto K.: *Structured Document Image Analysis*, Springer Verlag, 1992.

[29] di Baja G.S.: Well-shaped, stable and reversible skeletons from the (3,4)-distance transform, *Journal of Visual Communication and Image Representation*, 5(1), pp. 107-115, 1994.

[30] Baker B.S.: Tree transducers and tree languages, *Inform. Control*, 37(1), pp. 241-266, 1978.

[31] Barrero A.: Inference of tree grammars using negative samples, *Pattern Recognition*, 24(1), pp.1-8, 1991.

[32] Baumann S.: A simplified attributed graph grammar for high level music recognition, *Proc. 3rd ICDAR*, Montreal, pp. 1080-1083, 1995.

[33] Belaid A., Haton, J.P.: A syntactic approach for handwritten mathematical formula recognition, *IEEE Trans. PAMI*, 6(1), pp. 105-111, 1984.

[34] Belkasim S.O., et al: Pattern recognition with moment invariants: a comparative study and new results, *Pattern Recognition*, 24(10), pp. 1117-1138, 1991.

[35] Bhattacharjee S., Monagan G.: Recognition of cartographic symbols, Proc. Workshop on Machine Vision App., pp. 226-229, 1994.

[36] Binford T.O: Survey on model-based image analysis systems, *Int. J. Robotics Res.*, 1(1), pp. 18-64, 1982.

[37] Blake R.E.: Partitioning graph matching with constraints, *Pattern Recognition*, 27(3), pp. 439-446, 1994.

[38] Bley H.: Segmentation and preprocessing of electrical schematics using picture graphs, *CVGIP*, 28(2), pp. 271-288, 1984.

[39] Blum H.: A transformation for extracting new descriptors of shape, in: Wathen-Dunn W.: *Models for the perception of speech and visual form*, MIT, Press, Cambridge Mass.

[40] Blum M., Hewitt C.: Automata on a two-dimensional tape, *Proc. 8th IEEE Conf. Switching Automaton Theory*, pp. 155-160, 1968.

[41] Blum H:, Nagel R.: Shape description using weighted symmetric axis features, *Pattern Recognition*, 10(2), pp. 167-180, 1978.

[42] Boatto L., et. al.: An interpretation system for land register maps, *IEEE Computer*, 25(7), pp. 25-33, 1992.

[43] Borgefors G.: Distance transformations in arbitrary dimensions, *CVGIP*, 27(2), pp. 321-345, 1984.

[44] Brady M.: Criteria for representation of shapes, in: Beck J., et al: *Human and Machine Vision*, Academic Press, N.Y., pp. 39-84, 1983.

[45] Brayer J.M.: Parsing of web grammars, *IEEE Workshop Data Descr. Manag.*, Long Beach, CA, 1977.

[46] Brown D.E., et al: A parallel genetic heuristic for the quadratic assignment problem, in: [288], pp. 406-415.

[47] Bunke H., Allermann G.: Probabilistic relaxation for the interpretation of electrical schematics, *Proc. 2nd ICPR*, pp. 438-440, 1981.

[48] Bunke H.: Attributed graph grammar and their application to schematic diagram interpretation, *IEEE. Trans. PAMI*, 4(5), pp. 574-582, 1982.

[49] Bunke H., Allerman G.: Inexact graph matching for structural pattern recognition, *Pattern Recognition Letters*, 1(4), pp. 245-253, 1983.

[50] Bunke H., Haller B.: A parser for context-free plex grammars, LNCS 411, pp. 136-150, 1990.

[51] Bunke H., Sanfelieu A.: *Syntactic and structural pattern recognition. Theory and applications*, World Scientific, Singapore, 1990.

[52] Bunke H. (ed): *Advances in structural and syntactic pattern recognition*, World Scientific, Singapore, 1992.

[53] Bunke H., Csirik J.: Inference of edit costs using parametric string matching, *Proc. 11th ICPR*, Hague, v.2, pp. 549-552, 1992.

[54] Bunke H., Messmer B.T: Similarity measures for structured representations, in: Richter M.M., et al: Topics in case-based reasoning, *LNCS 951*, Springer Verlag, pp. 106-118, 1994.

[55] Bunke H., Messmer B.T.: Efficient attributed graph matching and its application to image analysis, *LNCS 974*, pp. 45-55, Springer Verlag 1995.

[56] Bunke H.: String grammar for syntactic pattern recognition, in: [51], pp. 29-53.

[57] Bunke H.: Structural and syntactic pattern recognition, in: [60], pp. 163-209.

[58] Chai I., Dori D.: Orthogonal zig-zag: an efficient method for extracting lines from engineering drawings, in: Arcelli C. et al, *Visual Form*, Plenum Press, N.Y., pp. 127-136, 1992.

[59] Chang C.L., Lee R.C.: *Symbolic Logic and mechanical theorem proving*, Academic Press, N.Y., 1973.

[60] Chen C.H., Pau L.F., Wang P.S.P: *Handbook of pattern recognition and computer vision*, World Scientific, Singapore, 1995.

[61] Chen Y., et al: Perfecting vectorized mechanical drawings, *CVIU*, 63(2), pp. 273-286, 1996.

[62] Chhabra A.: Graphic symbol recognition: an overview, *Proc. 2nd Int. Workshop Graphics Recognition*, Nancy, France, pp. 244-252, 1997.

[63] Christmas W.J., et al: Structural matching in computer vision using probabilistic relaxation, *IEEE Trans. PAMI*, 17(8), pp. 749-764, 1995.

[64] Collin S., Colnet D.: Syntactic anaysis of technical drawing dimensions, *Int. J. Pattern Recognition and Artificial Intelligence*, 8(5), pp. 1131-1148, 1994.

[65] Couasnon B., Camillerap J.: Using grammars to segment and recognize music scores, *Proc. IAPR Workshop on Document Analysis Systems*, Kaiserlautern, pp. 15-27, 1994.

[66] Csirik J., Bunke H.: Longest k-distance substrings of two strings, *Proc. 9th ICPR*, Rome, pp. 69-71, 1988.

[67] Dasarathy B.V. (ed): *Nearest Neighbor (NN) Norms: NN pattern classification techniques*, IEEE Computer Society Press, CA, 1991.

[68] Davies E.R., Plummer P.N.: Thinning algorithms: A critique and a new methodology, *Pattern Recognition*, 14(1), pp. 53-63, 1982.

[69] Dekking F.M., Van Otterloo P.J.: Fourier coding and reconstruction of complicated contours, *IEEE Trans. SMC*, 16(2), pp. 395-404, 1986.

[70] Dempster A.P.: A generalization of Bayesian inference, *J.Royal Statistical Society*, B-30, pp. 205-247, 1968.

[71] Deseilligny M.P., et al: Map understanding for GIS data capture: algorithms for road network graph reconstruction, *Proc. 2nd ICDAR*, Tsukuba, 1994.

[72] Deseilligny M.P. et al: Character string recognition on maps, a rotation-invariant method, *Pattern Recognition Letters*, 16(12), pp. 1297-1309, 1995.

[73] Deseilligny M.P., Mariani R.: A three year project on topographics map interpretation, *Proc. 2nd Int. Workshop Graphics Recognition*, Nancy, France, pp. 160-167, 1997.

[74] Devijer p.R., Kittler J.: *Pattern recognition: a statistical approach*, Prentice-Hall. Englewood Cliffs, N.J., 1982.

[75] Digricoli V.J.: Resolution by unification and equality, *Proc. 4th Workshop on Automated Deduction*, 1979.

[76] Donar J.E.: Tree acceptors and some of their applications, *J.Comput Sys. Sci.*, 4(50), pp. 207-233, 1970.

[77] Dori D., A. Pnueli: The grammar of dimensions in machine drawings, *CVGIP*, 42(1), pp. 1-18, 1988.

[78] Dori D.: A syntactic/geometric approach to recognition of dimensions in engineering drawings, *CVGIP*, 47(2), pp. 271-291, 1989.

[79] Dori D.: Dimensioning analysis: toward automatic understanding of engineering drawings, *Commun. ACM*, 35(10), pp. 92-103, 1992.

[80] Dori D., et al: Object-process based segmentation and recognition of ANSI and ISO Standard dimensioning texts, in: [180], pp. 212-232.

[81] Dori D. et al: Sparse pixel recognition of primitives in engineering drawings, *Machine Vision App.*, 6(1), pp. 69-82, 1993.

[82] Dori D., Bruckstein A.: *Syntactic and structural pattern recognition*, World Scientific, Singapore, 1995.

[83] Dori D.: Object process analysis: maintaining the balance between system structure and behaviour, *Journal of Logic and Computation*, 5(2), pp. 227-249, 1995.

[84] Dori D.: Vector-based arc segmentation in machine drawing understanding system environment, *IEEE Trans. PAMI*, 17(11), pp. 1057-1068, 1995.

[85] Dori D.: Orthogonal zig-zag: an algorithm for vectorizing engineering drawings compared with Hough transform, *Advances in Engineering Software*, 28(1), pp. 11-24, 1997.

[86] Dori D., et al: How to win a dashed line detection contest ?, in: [180], pp. 286-300.

[87] Dori D.: Sparse pixel vectorization: an algorithm and its performance evaluation, *IEEE Trans. PAMI*, 21(3), pp. 202-215, 1999.

[88] Dorst L., Verbeek P.: The constrained distance transform: a pseudo-Euclidean recursive implementation of the Lee algorithm, in: *Signal Processing III: theories and applications*, North Holland, 1986, pp. 917-920.

[89] Dorst L.: Pseudo-Euclidean skeletons, *Proc. 8th ICPR*, pp. 286-288, 1986.

[90] Duda R.O, Hart P.E.: *Pattern Classification and Scene Analysis*, John Wiley&Sons, New York, 1973.

[91] Duerr B., et al: A combination of statistical and syntactical pattern recognition applied to classification of unconstrained handwritten numerals, *Pattern Recognition*, 12(2), pp. 189-199, 1979.

[92] Dupont F. et al: Automatic interpretation of scanned maps: reconstruction of contour lines, *Proc. 2rd. Int. Workshop Graphics Recognition*, LNCS 1389, Springer-Verlag, pp. 194-207, 1997.

[93] Earley J.: An efficient context-free parsing algorithm, *Commun. ACM*, 13(1), pp. 94-102, 1970.

[94] Ebi N., et al: An image analysis system for automatic data acquisition from colored scanned maps, *Machine Vision App.*, 7, pp. 148-164, 1994.

[95] Ehrig H. Nagl M., Rozenberg G.: Graph grammars and their applications to computer science, *LNCS 153*, Springer Verlag, 1983.

[96] Ehrig H., Nagl M., Rozenberg G.: Graph grammars and their applications to computer science, *LNCS 291*, Springer Verlag, 1987.

[97] Ehrig H., Kreowski H.J., Rozenberg G.: Graph grammars and their applications to computer science, *LNCS 532*, Springer Verlag, 1991.

[98] Elliman D.G. Sen-Gupta M.: Automatic recognition of linear features symbols and textured areas within maps, *Proc. Workshop on Machine Vision App.*, pp. 239-242, 1994.

[99] Environmental Systems Research Institute Inc.: *Arc/Info data management*, Redlands, USA, 1994.

[100] Eshera M.A., Fu K.S.: A graph distance measure for image analysis, *IEEE Trans. SMC*, 14(3), pp. 398-408, 1984.

[101] Etherington D.W.: *Reasoning from incomplete information*, Pittman Publishing Ltd., London, 1987.

[102] Etherington D.W.: Formalizing nonmonotonic reasoning systems, *Artificial Intelligence*, 31(1), pp. 41-85, 1997.

[103] Fahn C.S., et al: A topology-based component extractor for understanding elecronic circuit diagrams, *CVGIP*, 44(1), pp. 119-138, 1988.

[104] Falkenheiner B., et al: The structure-mapping engine: algorithms and examples, *Artificial Intelligence*, 41(1), pp. 1-63, 1989.

[105] Feder T.: Plex languages, *Inf. Sci.*, 3(2), pp. 225-241, 1971.

[106] Feng J. et al: Inexact matching using neural networks, in: Gelsema E.S, Kanal L.N.: *Pattern recognition in practice: multiple paradigms, comparative studies and hybrid systems*, North-Holland, pp. 177-184, 1994.

[107] Ferrate H., Pavlidis T., Bunke H. Sanfelieu: *Syntactic and structural pattern recognition*, Springer Verlag, Berlin, 1988.

[108] Flasiński M.: Parsing of edNLC-graph grammars for scene analysis, *Pattern Recognition*, 21(6), pp. 623-629, 1988.

[109] Flasiński M.: Distorted pattern analysis with the help of Node Label Controlled graph languages, *Pattern Recognition*, 23(8), pp. 765-774, 1990.

[110] Flasiński M.: On the parsing of deterministic graph languages for syntactic pattern recognition, *Pattern Recognition*, 26(1), pp. 1-16, 1993.

[111] Flasiński M.: Use of graph grammars for the descripion of mechanical parts, *CAD*, 27(6), pp. 403-433, 1995.

[112] Fletcher L.A., Kasturi K.: A robust algorithm for text string separation from mixed text/graphics image, *IEEE Trans. PAMI*, 10(6), pp. 910-918, 1988.

[113] Freeman H.: On the encoding of arbitrary geometric configurations, *IEE Trans. Electronic Computers*, 10(2), pp. 260-268, 1961.

[114] Freeman H., Davis L.: A corner finding algorithm for chain-coded curves, *IEEE Trans. Comput.*, 26(2), pp. 297-303, 1977.

[115] Friedman J.H., et al An algorithm for finding nearest neighbors, *IEEE Trans. Comput.*, 24(10), pp. 1000-1006, 1975.

[116] Frost R.: *Introduction to knowledge base systems*, McGraw-Hill, N.Y., 1988.

[117] Fu K.S., Booth T.L.: Grammatical inference: introduction and survey- part 1, *IEEE Trans. SMC*, 5(1), pp. 95-111, 1975.

[118] Fu K.S., Booth T.L.: Grammatical inference: introduction and survey- part 1, *IEEE Trans. SMC*, 5(4), pp. 409-423, 1975.

[119] Fu K.S.: A clustering procedure for syntactic patterns, *IEEE Trans. SMC*, 7, pp. 734-742, 1977.

[120] Fu K.S.: *Syntactic pattern recognition and applications*, Prentice-Hall, Engelwood Cliffs, N.J., 1982.

[121] Fu K.S.: A step towards unification of syntactic and statistical pattern recognition, *IEEE Trans. PAMI*, 5(1), pp. 200-205, 1983.

[122] Fukunaga K. Narendra P.M. A branch and bound algorithm for computing k-nearest neighbors, *IEEE Trans. Comput.*, 24(6), pp. 750-753, 1975.

[123] Fukunaga K.: *Introduction to statistical pattern recognition*, Academic Press, San Diego, CA, 1990.

[124] *Fundamental Land Map. Instruction K-1*, Chief Geodesist of the Country, 1995, Warsaw.

[125] Fung L.W., Fu K.S.: Maximum-likelihood syntactic decoding, *IEEE Trans. Inf. Theory*, 21(3), pp. 423-430, 1975.

[126] Gao Q., Wong A., Curve detection based on perceptual organization, *Pattern Recognition*, 26(10), 1039-1046, 1993.

[127] Garey M.R., Johnson D.S.: *Computers and intractability: a guide to the theory of NP-completeness*, Freeman and Company, 1979.

[128] Garvey T.D.: Evidential reasoning for geographic evaluation for helicopter route planning, *IEEE Trans. GE-25(3)*, pp. 294-304, 1987.

[129] Gatnar E., Stąpor K.: *Prolog: language of artificial intelligence*, PLJ, Warsaw, 1992.

[130] Genesereth M.R., Nilsson N.J.: *Logical foundations of artificial intelligence*, Morgan Kaufmann, 1987.

[131] Ginsburg S.: *The mathematical theory of context-free languages*, McGraw-Hill, N.Y., 1966.

[132] Gonzalez R.C., Thomason M.G.: *Syntactic pattern recognition*, Addison Wesley, N.Y., 1978.

[133] Gonzalez R.C., Woods R.E.: *Digital Image Processing*, Addison Wesley, N.Y., 1992.

[134] Goodson K.J., Lewis P.H.: A knowledge-based line recognition system, *Pattern Recognition Letters*, 11(4), pp. 295-304, 1990.

[135] O'Gorman L., R. Kasturi: *Document image analysis*, IEEE Society Press, 1996.

[136] Grabska E.: Theoretical concepts of graphical modelling. Part one: realization of CP graphs, *Machine Graphics & Vision*, 2(1), pp. 3-38, 1993.

[137] Grabska E.: Theoretical concepts of graphical modelling. Part two: CP graph grammars and languages, *Machine Graphics & Vision*, 2(2), pp. 149-178, 1993.

[138] Graham S.L., et al: An improved context-free recognizer, *ACM Trans. Program. Lang. Syst.*, (2), pp. 415-449, 1980.

[139] Granlund G.H.: Fourier preprocessing for handprinted character recognition, *IEEE Trans. Comput.*, 21(1), pp. 195-201, 1972.

[140] Green E.A., Krishnamoorthy M.S.: Model-based analysis of printed tables, in: [180], pp. 80-92.

[141] Groen F.C.A., et al: Symbol recognition in electrical diagrams using probabilistic graphs matching, *Pattern Recognition Letters*, 8(3), pp. 343-350, 1985.

[142] Guo Z., Hall R.W.: Parallel thinning with two-subiteration algorithms, *Commun. ACM*, 32(3), pp. 359-373, 1987.

[143] Hall P.A.V., Dowling G.R.: Approximate string matching, *ACM Comput. Surveys*, 12(2), pp. 381-402, 1980.

[144] Hallard D.H.: Generalizing the Hough Transform to detect arbitrary shapes, *Pattern Recognition*, 13(2), pp. 111-122, 1981.

[145] Haralick R.M., Shapiro L.G.: The consistent labeling problem. Part 1, *IEEE Trans. PAMI*, 1(2), pp. 173-184, 1979.

[146] Haralick R.M., Shapiro L.G.: The consistent labeling problem. Part 2, *IEEE Trans. PAMI*, 2(3), pp. 193-203, 1980.

[147] Haralick R.M., Elliot G.L.: Increasing tree search efficiency for constraint satisfaction problems, *Artificial Intelligence*, 14(1), pp. 263-313, Elsevier, 1980.

[148] Haray F.: *Graph Theory*, Addison Wesley, N.Y., 1972.

[149] Hartigan J.A.: *Clustering algorithms*, John Wiley&Sons, N.Y., 1975.

[150] den Hartog J.E. et al: An alternative to vectorization: decomposition of graphics into primitives, *Proc. 3rd Annual Symp. on Document Analysis and Information Retrieval*, Las Vegas, pp. 263-274, 1994.

[151] den Hartog J.E. et. al.: Knowledge-based interpretation of utility maps, *CVIU*, 63(1), pp. 105-107, 1996.

[152] Herault L. et al: Symbolic image matching by simulated annealing, *Proc. British Machine Vision Conf.*, pp. 319-324, Oxford, 1990.

[153] Hilditch C.J: Linear skeletons from square cupboards, in: Meltzer B. et al: *Machine Intelligence*, v.4, University Press, Edinburgh, 1969.

[154] Hildreth E.: Computations underlying the measurement of visual motion, *Artificial Intelligence*, 22(1), pp. 1-27, 1983.

[155] Hirschberg D.S.: A linear space algorithm for computing maximal common subsequences, *Commun. ACM*, 18(6), pp. 341-343, 1975.

[156] Hopcroft J.E., Ullman J.D: *Introduction to automata theory, languages and computation*, Addison-Wesley, Reading, MA, 1979.

[157] Hori O., Okazaki A.: High quality vectorization based on a general object model, in: [28], pp. 325-339.

[158] Hu M.K.: Visual pattern recognition by moment invariants, *IEEE Trans. Information Theory*, 8(1), pp. 179-187, 1962.

[159] Hung S., Kasvand T.: Critical points on a perfectly 8- or 6-connected thin binary line, *Pattern Recognition*, 16(2), 297-306, 1983.

[160] Hunt J.W., Szymanski T.G.: A fast algorithm for computing longest common subsequences, *Commun. ACM*, 20(5), pp. 350-353, 1977.

[161] Ilg M.: Knowledge-based understanding of road maps and other line images, *Proc. 10th ICPR*, Atlantic City, 1990.

[162] Jain A.K., Dubes R.C: *Algorithms for clustering data*, Prentice-Hall, Engelwood Cliffs, N.J., 1988.

[163] Jakubowski R.: Extraction of shape features for syntactic recognition of mechanical parts, *IEEE Trans. SMC*, 1985, 15(5), pp. 624-651.

[164] Jakubowski R.: Decomposition of complex shapes for their structural recognition, *Inf. Sci.*, 50(1), pp. 35-71, 1990.

[165] Jakubowski R.: Multiple sweeping in product modelling and analysis. *Proc. of ASME Winter Annual Meeting*, DE-vol. 66, New Orleans, 28-Dec. 3, 1993.

[166] Jakubowski R., K. Stąpor: Structural method in perceptual organization of 2D-curve detection, representation and analysis, *Bull. of the Polish Academy of Sciences*, 47(3), pp. 309-324, 1999.

[167] Jakubowski R., K. Stąpor: A structural approach to representation of 3D-objects, *Bull. of the Polish Academy of Sciences*, 47(2), pp. 175-189, 1999.

[168] Jakubowski R., K. Stąpor: Structural/syntactic methods in perceptual organization of shape modeling and analysis, *Archives of Theoretical and Applied Computer Science*, 11(2), pp. 133-155, 1999.

[169] James M.: *Classification algorithms*, John Wiley&Sons, N.Y., 1985.

[170] Jang B., Chin R.T.: One-pass parallel thinning: analysis, properties and quantitative evaluation, *IEEE Trans. PAMI*, 14(11), pp. 1129-1139,1992.

[171] Janssens D., Rozenberg G.: On the structure of node label controled graph languages, *Info. Sci.*, 20(2), pp. 191-216, 1980.

[172] Janssens D., Rozenberg G., Verraedt R.: On sequential and parallel node-rewritting graph grammars, *CGIP*, 18(2), pp. 279-304, 1982.

[173] Janssen R., Vossepoel A.: Compilation of mosaics from separately scanned line drawings, *Proc. 2nd IEEE Workshop on Appl. on Computer Vision*, Sarasota, pp. 36-43, 1994.

[174] Janssen R., Vossepoel A.M.: Adaptive vectorization of line drawing images. *CVIU*, 65(1), pp. 38-56, 1997.

[175] Janssen R. D.: Interpretation of maps: from bottom-up to model-based, in: Bunke H., Wang P.S.P: *Hanbook of Character recognition and document image analysis*, World Scientific, Singapore, 1997.

[176] Jardine N., Sibson R.: *Mathematical taxonomy*, John Wiley&Sons, N.Y., 1971.

[177] Jimenez J., Navalon J.L: Some experiments in image vectorization, *IBM J. Res. Develop.*, 26(6), pp. 724-734, 1982.

[178] De Jong K.A. Spears W.M.: Using genetic algorithms to solve NP-complete problems, in: [288], pp. 124-132.

[179] Joseph S.H., Pridmore T.H.: Knowledge-directed interpretation of mechanical engineering drawings, *IEEE Trans. PAMI*, 14(90, pp. 928-940, 1992.

[180] Kasturi R., Tombre K.: Graphics recognition - methods and aplications, *LNCS 1072*, Springer Verlag, 1996.

[181] Kasturi R., et al: A system for interpretation of line drawings. *IEEE Trans. PAMI*, 12(10), pp. 978-992, 1990.

[182] Kasturi R.: An overview of techniques for graphics recognition, in: Baird H.S., et al: *Structured Document Image Analysis*, Springer-Verlag, pp. 285-324, 1992.

[183] Khotanzad A., Hong Y.H: Rotation invariant image recognition using features selected via a systematic method, *Pattern Recognition*, 23(10), pp. 1089-1101, 1990.

[184] Kim W., Kak A.C.: 3-D object recognition using biparite matching embedded in discrete relaxation, *IEEE Trans. PAMI*, 13(2), pp. 224-251, 1991.

[185] Kittler J., et al: Probabilistic relaxation for matching of symbolic structures, in: [52], pp. 471-480.

[186] de Knecht J.P., Schutte K.: Finding correspondence of utility maps with a GIS, *Proc. 13th ICPR*, pp. 755-759, Los Alamitos, 1995.

[187] de Knecht J.P., et al: Utility map reconstruction, *IEEE Document Image Analysis Workshop*, Puerto Rico, June, 1997.

[188] de Knecht J.P.: Reconstruction of geometric primitives in a map using a similarity transform with tiepoints, *Proc., 3rd Annual Conf. of the Advanced School for Comp. and Imaging*, Heijen, pp. 180-184, 1997.

[189] Krishnaiah P.R., Kanal L.N.: *Handbook of statistics 2: classification, pattern recognition and reduction of dimensionality*, North Holland, Amsterdam, 1982.

[190] Kruskal J.R., Sankhoff D.: An anthology of algorithms and concepts for sequence comparison, in: [284], pp. 265-310.

[191] Kulikowski J.L.: Some problems in structural analysis of complex patterns, *Arch. Automation and Telemechanics*, 3, 1970.

[192] Kulikowski J.L.: Structural analysis of biomedical data, *Biocybernetics and biomedical engineering*, 7(1-4), 1987.

[193] Kulikowski J.L.: Relational approach to structural analysis of images, *Machine Graphics & Vision*, 1(1/2), pp. 299-309, 1992.

[194] Kurozumi Y., Davis W.: Polygonal approximation by minimax method, *CGIP*, 19(1), 248-264, 1982.

[195] Kurzyński M.: *Pattern Recognition: A statistical approach* (in Polish), Wyd. Polit. Wrocł., 1997.

[196] Kurzyński M.: On the multistage Bayes classifier, *Pattern Recognition*, 21(3), pp. 355-365, 1988.

[197] Kurzyński M.: On two-stage Bayes classifier, *Proc. 8th Int. Conf. on Pattern Recognition*, vol. 1, pp. 150-153, 1986.

[198] Kurzyński M.: The optimal strategy of a tree classifier, *Pattern Recognition*, 16(1), pp. 81-87, 1983.

[199] Lai C.P., Kasturi R.: Detection of dimension sets in engineering drawings. *IEEE Trans. PAMI*, 16(8), 1994.

[200] Lam L., Lee S., Suen C.Y.: Thinning methodologies - a comprehensive survey, *IEEE Trans. PAMI*, 14(9), pp. 869-885, 1992.

[201] Lee S.W.: Recognizing hand-drawn electrical circuit symbols with attributed graph matching, in: [28], pp. 340-358.

[202] Leeuwen J.: *Handbook of theoretical computer science*, Elsevier Science Publishers, B.V., 1990.

[203] Leu J., Chen L.: Polygonal approximation of 2-D shapes through boundary merging, *Pattern Recognition Letters*, 7(4), pp. 231-238, 1988.

[204] Leung Y: *Intelligent spatial support system*, Springer Verlag, Berlin, 1997.

[205] Levenshtein V.I.: Binary codes capable of correcting deletions, insertions and reversals, *Sov. Phys. Dokl.*, 10, 8, pp.707-710, 1966.

[206] Lin X. et al: Efficient diagram understanding with characteristic pattern detection, *CVGIP*, 30(1), pp. 84-106, 1985.

[207] Lin B.C., Shen J.: Fast computation of moment invariants, *Pattern Recognition*, 24(8), pp. 807-813, 1991.

[208] Liu W., Dori D.: Recognition of hatching lines in engineering drawings, *Proc. 13th Israeli Symp. of AI, CV, and NN*, Tel Aviv, 1997.

[209] Lloyd J.W.: *Foundations of logic programming*, Springer Verlag, Berlin, 1984.

[210] Lu S., Fu K.S.: Error-correcting tree automata for syntactic pattern recognition, *IEEE Trans. Comput.*, 27(11), pp. 1040-1052, 1978.

[211] Lu S., et al; Hierarchical attributed graph representation and recognition of handwritten Chinese characters, *Pattern Recognition*, 24(7), pp. 617-632, 1991.

[212] Lu Z.: Detection of text regions fom digital engineering drawings, *IEEE Trans. PAMI*, 20(4), pp. 431-439, 1998.

[213] Lukaszewicz W.: *Non-monotonic reasoning. Formalization of commonsense reasoning*, Ellis Horwood Series in Artificial Intelligence, Ellis Horwood, N.Y., 1990.

[214] Luo H., Dinstein H.: Automatic extraction of character strings from maps using directional mathematical morphology, *Proc. ICARCV*, pp. 1689-1693, 1994.

[215] Luo H., Agam G., Dinstein I.: Directional mathematical morphology approach for line thinning and extraction of character strings from maps and line drawings, *Proc. 3rd ICDAR*, pp. 257-260, 1995.

[216] Luo H., Kasturi R.: Improved morphological operations for separations of characters from maps/graphics, *Proc. 2nd Int. Workshop on Graphics Recognition*, Nancy, France, pp. 8-15, 1997.

[217] Lyon G.: Syntax-directed least-errors analysis for context-free languages: a practical approach, *CACM*, 17, pp. 3-14, 1974.

[218] MacLane S., Birkhoff G.: *Algebra*, MacMilland, Canada, 1967.

[219] Manaf A., et al: Automatic recognition of symbols in utility maps, *Proc. 10th Annual Int. Conf. Frontiers of Computer Technology*, Singapore, pp. 857-861, 1994.

[220] Masek W.J., Paterson M.S.: A faster algorithm for computing string-edit distances, *J. Computer and System Sciences*, 20(1), pp. 18-31, 1980.

[221] Matsuyama T., Hvang V.: *Sigma: knowledge-based aerial image understanding system*, N.Y., Plenum Press, 1990.

[222] Mayer H., et al: Knowledge-based interpretation of scanned large-scaled maps using multilevel modelling, Int. Archives of Photogrammetry and Remote Sensing, *Proc. 17th ISPRS Congress*, Washington, USA, 1992.

[223] McCarthy J.: Circumscription - a form of non-monotonic reasoning, *Arificial Intelligence*, 13(1/2), pp. 27-39, 1980.

[224] McCarthy J.: Applications of circumscription to formalizing commonsense knowledge, *Artificial Intelligence*, 28(1), pp. 89-116, 1986.

[225] McDermott D., Doyle J.: Non-monotonic logic I, *Artificial Intelligence*, 13(1/2), pp. 41-72, 1980.

[226] McDermott D.: Non-monotonic logic II, *Journal ACM*, 29(1), pp. 33-57, 1982.

[227] McKeown D.M., et al: Automating knowledge acquisition for aerial image interpretation, *CVGIP*, 46(1), pp. 37-81, 1989.

[228] Medioni G., Yasumoto Y.: Corner detection and curve representation using cubic B-splines, *CVGIP*, 29(1), pp. 267-278, 1987.

[229] Messmer B.T., Bunke H.: Efficient error-tolerant subgraph isomorphism detection, in: [82], pp. 231-240.

[230] Messmer B.T., Bunke H.: A network-based approach to exact and inexact graph matching, *Technical Report IAM-93-021*, Univ. Bern, 1993.

[231] Meyer B.: *Object-oriented software construction*, Prentice-Hall, Englewood Cliffs, N.J.

[232] Mohr R.: A general purpose line drawing analysis system, in: [51], pp. 479-497.

[233] Mohr R., Pavlidis T., Sanfelieu A.: *Structural pattern analysis*, World Scientific, Singapore, 1990.

[234] Mokhtarian F., Mackworth A.K.: A theory of multiscale, curvature-based shape representation for planar curves, *IEEE Trans. PAMI*, 14(8), pp. 789-805, 1992.

[235] Mokrzycki W.: *Discretization of algebraic curves in computer displaying information* (in Polish), Inst. Mathematical Machines, Warsaw, 1985.

[236] Monagan G., Roosly M.: Appropriate base representation using a run graph, *Proc. 2nd ICDAR*, Tsukuba, Japan, pp. 623-626, 1993.

[237] Morean O.A., Kasturi R.: Symbol identification in geographic maps, *Proc. 7th ICPR*, pp. 966-967, 1984.

[238] Mori S., et al: Historical review of OCR research and development, *Proc. IEEE*, 80(10), pp. 1029-1098, 1992.

[239] Morris J.B.: E-resolution: extension of resolution to include the equality relation, *Proc. of the IJCAI-69*, pp. 287-294, 1969.

[240] Mostowski A., Stark M.: *Higher Algebra*, PWN, 1954.

[241] Myaeng S.H., Lopez-Lopez A.: Conceptual graph matching: a flexible algorithm and experiments, J. of Experimental and Theoretical *Artificial Intelligence*, 4(2), pp. 107-126, April 1992.

[242] Myers G., et al: Verification-based approach for automated text and feature extraction, in: Kasturi R., Tombre K.: Graphics recognition, *LNCS 1072*, Springer Verlag, 1995.

[243] Naccache N., Shinghal R.: SPTA: A proposed algorithm for thinning binary patterns, *IEEE Trans. SMC*, 14(3), pp. 409-418,1984.

[244] Nagasamy V., Langrana N.A.: Engineering drawing processing and vectorization system, *CVGIP*, 49(2), pp. 379-397, 1990.

[245] Nagl M.: A tutorial and bibliographical survey on graph grammars, in: *Graph Grammars and their applications for computer science and biology*, Springer Verlag, New York, 1979.

[246] Nagy G., et al: A prototype document image analysis system for technical journals, *IEEE Computer Magazine*, 25(7), pp. 10-22, 1992.

[247] Nagy G.: At the frontiers of OCR, *Proc. IEEE*, 80(7), pp. 1093-1100, 1992.

[248] Nakamura A., et al: A method for recognizing character strings from maps using linguistic knowledge, *Proc. 2nd ICDAR*, pp. 561-564, 1993.

[249] Nakejima C., Yazawa T.: Automatic recognition of facility drawings and street maps utilizing the facility management database, *Proc. 3rd ICDAR*, Canada, pp. 516-519, 1995.

[250] Nilsson N.J.: *Principles of artificial intelligence*, Berlin, Springer Verlag, 1980.

[251] Nishida H., Mori S.: Algebraic description of curve structure, *IEEE Trans. PAMI*, 14(5), pp. 517-533, 1992.

[252] Ogier J.M.: Multilevel approach and distributed consistency for technical map interpretation: application to cadastral maps, *CVIU*, 70(3), pp. 438-451, 1998.

[253] Pavlidis T.: *Structural Pattern Recognition*, Springer Verlag, New York, 1977.

[254] Pavlidis T.: *Algorithms for graphics and image processing*, Rockville, MD.: Comput. Sci., pp. 119-214, 1982.

[255] Pearl J.: *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, San Mateo, 1988.

[256] Peng J., et al: A new parsing scheme for plex grammars, *Pattern Recognition*, 23(3), pp. 393-402, 1990.

[257] Persoon E., Fu K.S.: Shape discrimination using Fourier descriptors, *IEEE Trans. SMC*, 7(1), pp. 17-179, 1977.

[258] Poole D., et al: Theorist: a logical reasoning system for defaults and diagnosis, in: Cercone N.J., et al: *The knowledge frontier: essays in representations of knowledge*, Springer-Verlag, Berlin, pp. 331-352, 1987.

[259] Priestnall G., et al: Arroehead recognition during automated data capture, *Pattern Recognition Letters*, 17(2), pp. 277-286, 1996.

[260] Puchała W., Stąpor K., Wojciechowski K.: Integrated Spatial Organization System SIT-y v. 6.2, *Proc. 2nd ArcInfo User's Conf.*, Warsaw, 24-25 Sept., pp. 21-24, 1996.

[261] Quinmimg H., et al: Method to extract and recognize topographic symbols from contour maps, *Proc. of SPIE Conf. on automatic object recognition*, pp. 305-311, Orlando, 1992.

[262] Ramer U.: An iterative procedure for the polygonal approximation of plane curves, *CGIP*, 1(2), pp. 244-256, 1972.

[263] Ray B.K., Ray K.S.: Detection of significant points and polygonal approximation of digitized curves, *Pattern Recognition Letters*, 13(5), 443-452, 1992.

[264] Reiher E. et al: A system for efficient and robust map symbol recognition, *Proc. 12th ICPR*, Austria, pp. 783-787, 1996.

[265] Reingold E.M., Nievergelt J., Deo N.: *Combinatorial algorithms*, PWN, Warsaw, 1985.

[266] Reiter R.: A logic for default reasoning, *Artificial Intelligence*, 13(1), pp. 81-132, 1980.

[267] Reiter R.: On closed-world databases, in: Gallaire H., Minker J.: *Logic and databases*, Plenum Press, N.Y., pp. 55-76, 1978.

[268] Rich E.: *Artificial Intelligence*, McGraw-Hill, N.Y., 1988.

[269] Robinson G.A., Wos L.: Paramodulation and theorem proving in first order theories with equality, in: Meltzer B., et al: *Machine Intelligence*, v.4, American Elsevier, pp. 135-150, 1969.

[270] Rocha J., Pavlidis T.: A shape analysis model with applications to a character recognition system, *IEEE Trans. PAMI*, 16(4), pp. 393-404, 1994.

[271] Roosli M., Monagan G.: Adding geometric constarints to the vectorization of line drawings, in: [180], pp. 49-56.

[272] Roosli M.,Monagan G.: A high quality vectorization combining local quality measures and global constarints, *Proc. 3rd ICDAR*, pp. 243-248, 1993.

[273] Rosenfeld A. et al: Scene labeling by relaxation operations, *IEEE Trans. SMC*, pp. 420-433, 1976.

[274] Rosenfeld A.: Array, tree and graph grammars, in: [51], pp. 85-115.

[275] Rosenfeld A., Johnston E.: Angle detection on digital curves, *IEEE Trans. Comput.*, 22(8), 875-878, 1973.

[276] Rosenfeld A., Kak A.C.: *Digital picture processing*, Academic Press, N.Y., 1976.

[277] Rosenfeld A., Milgram D.L.: Web automata and web grmmars, *Machine Intelligence*, 7(3), pp. 307-324, 1972.

[278] Rosenkrantz D.J.: Programmed grammars and classes of formal languages, *J. ACM*, 15(1), pp. 107-131, 1968.

[279] Rosin P., West G.: Segmentation of edges into lines and arcs, *Image and Vision Computing*, 7(2), pp. 107-114, 1989.

[280] Samet H., Soffer A.: A legend-driven geographic symbol recognition system, *Proc. 2nd ICPR*, pp. 350-355, 1994.

[281] Sandewall E.: An approach to frame problem and its implementation, in: Meltzer B., et al: *Machine Intelligence*, 7, Edingburgh University Press, pp. 195-204, 1972.

[282] Sanfelieu A.: Matching methods, *Proc. 8th Scandinavian Conf. on Image Analysis*, Norway, pp. 353-370, 1993.

[283] Sanfelieu A., Fu K.S.: A distance measure between attributed relational graphs for pattern recognition, *IEEE Trans. SMC*, 13(3), pp. 353-362, 1983.

[284] Sankoff D., Kruskal J.R.: *Time warps, string edits and macromolecules: the theory and practice of sequence comparison*, Addison Wesley, Reading, MA, 1983.

[285] Schalkoff R.J.: *Pattern Recognition: statistical, structural and neural approaches*, John Wiley&Sons, N.Y., 1992.

[286] Seong D.S., et al: Incremental clustering of attributed graphs, *IEEE Trans. SMC*, 23(5), pp. 1399-1411, 1993.

[287] Serra J.: *Image analysis and mathematical morphology*, Academic Press, London, 1982.

[288] Shaffer J.D.: *Genetic algorithms*, Morgan Kaufmann, 1989.

[289] Shafer G.: *A mathematical theory of evidence*, Princeton University Press, 1976.

[290] Shapiro L.G., Haralick R.M.: A metric for comparing relational descriptions, *IEEE Trans. PAMI*, 7(1), pp. 90-94, 1985.

[291] Shapiro L.G., Haralick R.M.: Structural descriptions and inexact matching, *IEEE Trans. PAMI*, 3(5), pp. 504-519, 1981.

[292] Shapiro L.G.: Relational matching, in: [377], pp. 475-496.

[293] Shapiro L.G.: Matching relational structures using discrete relaxation, in: [51], pp. 179-185.

[294] Shapiro L.G., Haralick R.M.: Organization of relational models for scene analysis, *IEEE Trans. PAMI*, 4, 6, pp. 595-602, 1982.

[295] Shapiro L.G.: The use of numerical relational distance and symbolic differences for organizing models and for matching, in: Rosenfeld A.: *Techniques for 3D machine perception*, Elsevier Science Publ., pp. 255-270, 1986.

[296] Shapiro L.G., Haralick R.: Decomposition of two-dimensional shapes by graph-theoretoc clustering, *IEEE Trans. PAMI*, 1(1), pp. 10-20, 1979.

[297] Shaw A.C.: Parsing of graph representable pictures, *J.ACM*, 17(4), pp. 453-487, 1969.

[298] Shi Q.Y., Fu K.S.: Parsing and translation of attributed expansive graph languages for scene analysis, *IEEE Trans. PAMI*, 5(4), pp. 472-485, 1983.

[299] Shimada S. et al: Agent-based parallel recognition method of contour lines, *Proc. 3rd ICDAR*, pp. 154-157, 1995.

[300] Shimotsuji S. et al: A robust recognition system for drawing superimosed on a map, *Computer*, 25(7), pp. 56-59, 1992.

[301] Shoam Y.: *Reasoning about change: time and causation from the standpoint of artificial intelligence*, MIT Press, Cambridge, MA, 1988.

[302] Shridhar M., Badreldin A.: Context-directed segmentation algorithm for handwritten numeral strings, *Image and Vision Computing*, 5(1), pp. 3-9, 1987.

[303] Simon J.C., Baret O.: Regularities and singularities in line pictures, in: [28], pp. 261-285.

[304] Sinha R.M.: A width-independent algorithm for character skeleton estimation, *CGIP*, 40(2), pp. 388-397, 1987.

[305] Sklansky J., Gonzalez F.: Fast polygonal approximation of digitized curve, *Pattern Recognition*, 12(3), 327-331, 1980.

[306] Smeulders A.W.M, ten Kate T.K.: Software system design for paper map conversion, *Post-proceedings, Int. Workshop on Graphics Recognition, LNCS 1072*, pp. 204-212, Springer Verlag, 1996.

[307] Smith R.W.: Computer processing of line images: a survey, *Pattern Recognition*, 20(1), pp. 7-15, 1987.

[308] Stanat D.F., McAllister D.F.: *Discrete mathematics in computer science*, Prentice-Hall, Englewood Cliffs, N.J., 1977.

[309] Starovoitov V. et al: Binary texture border extraction on line drawings based on distance transform, *Pattern Recognition*, 26(8), pp. 1165-1176, 1993.

[310] Stąpor K.: Processing of spatial join using spatial index and methods of computational geometry, *Archives of Theoretical and Applied Computer Science*, 9(1-4), pp. 199-210, 1997.

[311] Stąpor K.: Optimization of the R-tree structure - the spatial access method for multidimensional objects (in Polish), *Scientific Letters*, Silesian Technical University, 35, pp. 113-128, 1998.

[312] Stąpor K.: The review of spatial access methods (in Polish), *Scientific Letters*, Silesian Technical University, 35, pp. 96-111, 1998.

[313] Stąpor K.: On some spatial indexing methods in geographical information systems (in Polish), *Scientific Letters*, Silesian Technical University, 34, pp. 407-421, 1998.

[314] Stąpor K.: Recognition of cartographic symbols based on a structural model of a shape, *Machine Graphics & Vision*, 8(1), pp. 129-143, 1999.

[315] Stąpor K.: A vectorized thinning algorithm for handwritten symbols recognition, *Machine Graphics & Vision*, 8(3), pp. 341-353, 1999.

[316] Stąpor K., Skabek K., Tomaka A.: Model-based recognition of polyhedral objects from single intensity image using aspect graph, *Machine Graphics & Vision*, 8(2), pp. 249-265, 1999.

[317] Stąpor K.: A hybrid method for recognition of cartographic symbols, *Proc. 1st Polish Conf. Comput. Recognition Systems*, KOSYR, Trzebieszowice, pp. 201-207, 1999.

[318] Stąpor K.: A description language and a parsing algorithm for recognition of distorted line objects in map drawings, *Archives of Theoretical and Applied Computer Science*, 11(3/4), 1999.

[319] Stąpor K.: A knowledge-based approach to map interpretation by means of a semantic network, *Archives of Theoretical and Applied Computer Science*, 11(3/4), 1999.

[320] Stąpor K.: Geographic map image interpretation - survey and problems, *Machine Graphics & Vision*, 9(1/2), pp. 497-519, 2000.

[321] Stąpor K.: A knowledge-based framework for geographic map image analysis, 9(3), pp. 647-671, *Machine Graphics & Vision*, 2000.

[322] Stąpor K.: Complementary reasoning for geographic map image understanding, to be published in *Archives of Theoretical and Applied Computer Science*.

[323] Stąpor K.: Geographic symbol recognition by many-to-one inexact graph matching, submitted to *Machine Graphics & Vision*.

[324] Stefanelli R., Rosenfeld A.: Some parallel thinning algorithms for digital pictures. *J. ACM*, 18(2), pp. 255-264, 1971.

[325] Stefik M.: *Introduction to knowledge systems*, Morgan Kaufmann, 1997.

[326] Suzuki S. et al: Automatic line drawing recognition of large-scale maps, *Optical Engineering*, 26(7), pp. 642-649, 1987.

[327] Suzuki S., Yamada T.: MARIS: map recognition input system, *Pattern Recognition*, 23(8), pp. 919-933, 1990.

[328] Suzuki T., Mori S.: Structural description of line images by the cross section sequence graph. *Int. Journal of Pattern Recognition and Computer Vision*, 7(5), pp. 1055-1076, 1993.

[329] Suzuki T., Mori S.: Structural description of line images by the cross section sequence graph, *Int. Journal of Pattern Recognition and Computer Vision*, 7(5), pp. 1055-1076.

[330] Swain P.H., Fu K.S.: Stochastic programmed grammars for syntactic pattern recognition, *Pattern Recognition*, 4(1), pp. 83-100, 1972.

[331] Tadeusiewicz R., Flasiński: *Pattern Recognition* (in Polish), PWN, 1991.

[332] Tai K.C: The tree-to-tree correction problem, *J.ACM*, 26(3), pp. 422-433, 1979.

[333] Tanaka E.: Parsing and error-correcting parsing for string grammars, in: [51], pp. 57-83.

[334] Tanaka E., Fu K.S.: Error-correcting parsers for formal languages, *IEEE Trans. Comput.*, 31(3), pp. 327-328, 1982.

[335] Tanaka E.: An improved error-correcting parser for context-free languages, *Trans. IECE*, 67(3), pp. 379-385, 1984.

[336] Tanaka E., et. al: Direct parsing, *Pattern Recognition*, 19(3), pp. 315-323, 1986.

[337] Tanaka E.: Theoretical aspects of syntactic pattern recognition, *Pattern Recognition*, 7(10), pp. 1053-1061, 1995.

[338] Tanaka E.: A string correction method based on the context-dependent similarity, in: [107]. pp. 3-17.

[339] Teh C.H., Chin R.C.: On digital approximation of moment invariants, *CVGIP*, 33(2), pp. 318-326, 1986.

[340] Teh C.H., Chin R.C.: On image analysis by the method of moments, *IEEE Trans. PAMI*, 10(4), pp. 496-513, 1988.

[341] Teh C.H., Chin R.C.: On the detection of dominant points on digital curves, *IEEE Trans. PAMI*, 8(8), pp. 859-872, 1989.

[342] Tombre K.: Graphics recognition - general context and challenges, *Pattern Recognition Letters*, 16(8), pp. 883-891, 1995.

[343] Tombre K.: Stable and robust vectorization: how to make the right choices, *Proc. 3rd IAPR Int. Workshop Graphics Recognition*, pp. 3-17, Jaipur, India.

[344] Trier O.D., Taxt T.: Improvement of integrated function algorithm for binarization of document images, *Pattern Recognition Letters*, 16(3), 277-283.

[345] Trier O.D. et al: Recognition of digits in hydrographic maps: binary vs topographic analysis, *IEEE Trans. PAMI*, 19(4), pp. 399-404.

[346] Trier O.D. et al: Goal-directed evaluation of binarization methods, *IEEE Trans. PAMI*, 12(11), pp. 1191-1201, 1995.

[347] Tsai W.H., Fu K.S.: Error-correcting isomorphisms of attributed relational graphs for pattern recognition, *IEEE Trans. SMC*, 9(8), pp. 757-768, 1979.

[348] Tsai W.H., Fu K.S: Attributed grammar - a tool for combining syntactic and statistical approaches to pattern recognition, *IEEE Trans. SMC*, 10(9), pp. 873-885, 1980.

[349] Tsai W.H., Fu K.S.: Subgraph error-correcting isomorphism for syntactic pattern recognition, *IEEE Trans. SMC*, 13(1), pp. 48-62, 1983.

[350] Tsai W.H., Yu S.S.: Attributed string matching with merging for shape recognition, *IEEE Trans. PAMI*, 7(4), pp. 453-462, 1985.

[351] Ukkonen E.: Algorithms for approximate string matching, *Inf. Control*, 64(2), pp. 100-118, 1985.

[352] Ullman J.R.: An algorithm for subgraph isomorphism, *J. ACM*, 23(1), pp. 31-42, 1976.

[353] Umeyama S.: An eigendecomposition approach to weighted graph matching problems, *IEEE Trans. PAMI*, 10(5), pp. 695-703, 1988.

[354] Vaxiviere P., Tombre K.: Cellestin: CAD conversion of mechanical drawings, *IEEE Computer*, 25(5), pp. 46-54, 1992.

[355] Vaxiviere P., Tombre K.: Subsampling: A structural approach to technical document vectorization, in: Dori D., Bruckstein A., *Shape, structure and pattern recognition*, World Scientific, Singapore, pp. 323-332, 1995.

[356] Viswanathan M., Krishnamoorthy M.S: A syntactic approach to document segmentation, in: [233], pp. 197-215.

[357] Vossepoel A.V. et al: Memory efficient skeletonization of utility maps, *Proc. 4th ICDAR*, Ulm, Germany, pp. 797-800, 1997.

[358] Wagner R.A., Fisher M.J.: The string-to-string correction problem, *J.ACM*, 21, pp. 168-173, 1974.

[359] Wahl F.M, et al: Block segmentation and text extraction in mixed text/image documents, *CVGIP*, 20(3), pp. 375-390, 1982.

[360] Wall K., Danielsson P.: A fast sequential method for polygonal aproximation of digitized curves, *CVGIP*, 28(2), pp. 220-227, 1984.

[361] Wang P.S.P. (ed): Special Issue on array grammars, patterns and recognizers, *Int. J. Pattern Recogn. Artif. Intell.*, 3(3/4), 1989.

[362] Wang C.H.: A framework for object recognition in a visually complex environment and its application to locationg address blocks in mail pieces, *Int. J. Comp. Vision*, 2, pp. 125-151, 1988.

[363] Wang P.S.P, Zhang Y.Y.: A fast and flexible thinning algorithm. *IEEE Trans. on Comput.*, 38(5), 741-745,1989.

[364] Wenyin L., Dori D.: Incremental arc segmentation algorithm and its evaluation, *IEEE Trans. PAMI*, 20,4, pp. 424-431, 1998.

[365] Wolberg G.: A syntactic omni-font character recognition system, *Int. J. Pattern Recognition and Artificial Inteligence*, 1(3/4), pp. 303-322, 1987.

[366] Wong A.W.: Three-dimensional object recognition by attributed graphs, in: [51], pp. 381-314.

[367] Wos L., et al: The concept of demodulation in theorem proving, *J. ACM*, 14(4), pp. 698-709, 1967.

[368] Yamada H.K.: Contour DP matching method and its application to handprinted Chinese character recognition, *Proc. 7th ICPR*, Montreal, pp. 389-392, 1984.

[369] Yamada H.K. et al: Recognition of elevation value in topographic maps by Multi-Angled parallelism, *Int. J. Pattern Recognition and Machine Intelligence*, 8(5), pp. 149-173, 1994.

[370] Yamada H.K. et al: MAP: Multi-angled parallelism for feature extraction from topographical maps, *Pattern Recognition*, 24(6), pp. 479-488, 1991.

[371] Yamada H.K. et al: Directional mathematical morphology and reformalized Hough transform for the analysis of topographic maps, *IEEE Trans. PAMI*, 15(4), pp. 380-387, 1993.

[372] Yamamoto K. et al: Symbol recognition and surface reconstruction from topographic map by parallel method, *Proc. 2nd ICDAR*, pp. 914-917, 1993.

[373] Yamasaki S., Tonomura T.: On a bottom-up least error-correcting algorithm for context-free languages, *J. Info. Process. Soc.*, 18, pp. 781-788, 1977.

[374] Yanowitz S.D., Bruckstein A.M.: A new method for image segmentation, *CVGIP*, 46(1), pp. 82-95, 1989.

[375] You K.C., Fu K.S.: Distorted shape recognition using attributed grammars and and error-correcting techniques, *CGIP*, 13(1), pp. 1-16, 1980.

[376] You K.C., Fu K.S.: A syntactic approach to shape recognition using attributed grammars, *IEEE Trans. SMC*, 9(6), pp. 334-345, 1979.

[377] Young T.Y.: *Handbook of pattern recognition and image processing: computer vision*, Academic Press, New York, 1993.

[378] Xu L., Oja E.: Improved simmulated annealing, Boltzmann machine, and attributed graph matching, *LNCS 412*, Springer Verlag, pp. 151-161, 1990.

[379] Zahn C.T., Roskies R.Z.: Fourier descriptors for plane closed curves, *IEEE Trans. Comput.*, 21(2), pp. 269-281, 1972.

[380] di Zenzo, et al: Optical recognition of handprinted characters of any size, position, and orientation, *IBM J. Res. Develop.*, 36(3), pp. 487-501, 1992.

[381] Zhang T.Y., Suen C.Y.: A fast parallel algorithm for thinning digital patterns, *Commun. ACM*, 27(3), 236-239, 1984.

Integration of structural pattern recognition methods into knowledge-based framework

Application to geographic map image analysis

Abstract

In this thesis the generally applicable reasoning framework with the developed and properly integrated within it structural classifiers for automatic conversion of paper documents into a digital form is proposed. As the application of the proposed methods, the Polish Fundamental Land Map has been chosen. The proposed framework is based on the knowledge-based approach, and is composed of the three schemas: 1) *map model*, represented in the form of the developed hybrid semantic network which is a combination of the different knowledge representation formalisms: semantic network, rules, frame system and the object-oriented computation paradigm, 2) *object detection methods* (structural classifiers), 3) *image analysis flow scheme* composed of the two structures: mixed control mechanism and the non-monotonic reasoning method called complementary reasoning. The image analysis flow scheme, which is based on the proposed intelligent analysis strategy, is accomplished by having the initial recognition of seed objects followed by the progressive extraction of the remaining layers of geographic objects through the iteration of the 4-step interpretation cycle: hypotheses generation, compatibility examination, scenario selection, and scenario verification.

Each of the developed structural classifiers of the different graphics/text components of map drawing is based on the appropriate object model. The simplest is the polygon detector which is based on the defined polygon models. The second detector relies on the relational model of a general 2D curve, and the recognition method consists of matching relational structures (i.e. searching for relational homomorphism), followed by distance calculation in a parameter space. The third detector is based on the error-tolerant graph matching procedure between objects represented by the attributed graphs which is a search for an optimal many-to-one graphs homomorphism. The most general detector is the fourth one, in which objects are represented by the programmed, higher-dimensional extension of a string grammar, and the recognition is performed by error-correcting, mixed (top-down/bottom-up) parsing. All detectors are translational and rotational invariant and are tolerant to many complex structural deformations, including topological ones.

The proposed framework is composed of the following modules: 1) *reasoning engine* which is is a global coordinator of the reasoning performed by the independent object instances, 2) *detectors manager*, 3) *detectors* which are based on the different pattern recognition techniques and use the intrinsic properties of objects for model-based recognition, 4) *vector graph construction* module, 5) *resegmentation* module which realizes the refining matches by resegmentation technique to recover the missing object features, and 5) *evidence pool* which is a database where accumulation of object instances and hypotheses is performed during the analysis.

The following 4-level structure of geographic map image analysis process is proposed in the thesis: 1) *pre-processing* level composed of binarization, filled region extraction and thinning, 2) *vector graph construction* level, i.e. line structure tracking, vector graph segmentation, small and big graph vectorizations, 3) *graphics and text recognition* level, 4) *understanding* level.

The proposed, general reasoning framework, together with the developed structural classifiers respond to the all posed problems of geographic map image analysis (reasoning with incomplete information, control strategy, contextual layer separation, reliable analysis), as well as of the structural pattern recognition (limited descriptive power of string grammars, errorneous patterns, high number of prototypes, interaction of segmentation with recognition, selection of region of interest). The proposed approach to document image analysis enables for increased flexibilty: the image analysis flow scheme is independent of the application, the map model, dependent on the application, can be used for another ones only after small adjustments.

The proposed framework has been implemented as the map conversion module *MAPIN*, which replaced the manual input of paper maps into the Geographic Information System, converting correctly about 70% of the information in the maps and leaving the remaining 30% to the operator.

Integracja metod strukturalnych rozpoznawania obrazów z opartą na wiedzy strukturą wnioskującą

Zastosowanie do automatycznej analizy map geograficznych

Streszczenie

W rozprawie przedstawiono zaproponowaną, ogólną strukturę wnioskującą wraz z wbudowanymi strukturalnymi klasyfikatorami dla celu automatycznej analizy (konwersji) dokumentów z postaci papierowej do odpowiedniego formatu cyfrowego. Jako zastosowanie opracowanych metod wybrano Polską Zasadniczą Mapę Kraju. Zaproponowana struktura wnioskująca bazująca na opartym na wiedzy podejściu złożona jest z trzech schematów: 1) *modelu mapy* w postaci hybrydowej sieci semantycznej opartej na różnych formalizmach reprezentacji wiedzy: sieci semantycznej, reguł, ram i podejścia obiektowo-zorientowanego, 2) *metod detekcji obiektów* (strukturalnych klasyfikatorów), 3) *schematu przebiegu analizy obrazu* złożonego z dwóch podschematów: mieszanego mechanizmu kontroli oraz metody wnioskowania niemonotonicznego, zwanego wnioskowaniem uzupełniającym. Schemat przebiegu analizy obrazu, oparty na zaproponowanej, inteligentnej strategii analizy składa się z wstępnego rozpoznawania obiektów zarodkowych, po którym następuje ekstrakcja pozostałych warstw obiektów geograficznych poprzez iterację cyklu interpretacji złożonego z czterech kroków: generacji hipotez, testu kompatybilności, wyboru scenariusza oraz jego weryfikacji.

Każdy z zaprojektowanych strukturalnych klasyfikatorów graficznych/tekstowych komponentów mapy wykorzystuje odpowiedni model obiektu. Najprostszym jest detektor wielokątów, który oparty jest na zdefiniowanych modelach wielokątów. Drugi z detektorów oparty jest na relacyjnym modelu krzywej, zaś algorytm rozpoznawania polega na uzgadnianiu struktur relacyjnych (poszukiwaniu relacyjnego homomorfizmu), po którym następuje obliczanie odległości w przestrzeni cech. Metoda rozpoznawania w trzecim detektorze jest przybliżonym, odpornym na strukturalne zniekształcenia porównywaniem grafów atrybutywnych reprezentujących obiekty wzorca oraz nieznanego obiektu. Najbardziej uniwersalnym jest detektor czwarty, w którym obiekty są reprezentowane za pomocą rozszerzonej o specjalne operatory, programowej gramatyki łańcuchowej, zaś rozpoznawanie polega na niedokładnej, mieszanej analizie syntaktycznej, z wbudowaną korekcją błędów. Wszystkie opracowane klasyfikatory są odporne na translację oraz obroty, a także wykazują dużą tolerancję na różnego typu deformacje strukturalne obiektów.

Zaproponowana struktura wnioskująca zrealizowana jest za pomocą następujących modułów: 1) *mechanizmu wnioskującego*, który jest globalnym koordynatorem procesów wnioskowania poszczególnych obiektów, 2) modułu *zarządzania detektorami obiektów*, 3) *detektorów*, opartych na różnych metodach strukturalnego rozpoznawania obrazów, 4) modułu *konstrukcji grafu wektorowego*, 5) modułu *resegmentacji*, w którym wykonywane jest wielokrotne powtarzanie procedury segmentacji ze zmienionymi wartościami para-

metrów w przypadku niemożliwości detekcji poszukiwanych obiektów przez detektory, 5) *bazy faktów*, w której akumulowane są wszystkie pośrednie wyniki rozpoznawania (obiektów i hipotez).

W pracy zaproponowano i zrealizowano następującą, 4-poziomową strukturę analizy map geograficznych: 1) *przetwarzanie wstępne*, obejmujące binaryzację, ekstrakcję regionów wypełnionych oraz szkieletyzację, 2) *konstrukcję grafu wektorowego* (śledzenie linii, segmentacja, wektoryzacja małego i dużego grafu, 3) *rozpoznawanie komponentów graficznych oraz tekstowych*, 4) *interpretacja*.

Zaproponowana struktura wnioskująca wraz z wbudowanymi strukturalnymi klasyfikatorami pozwala na rozwiązanie postawionych w rozprawie problemów dotyczących zarówno samej automatycznej analizy map geograficznych (wnioskowanie z niepełną informacją, mechanizm kontroli, kontekstowa segmentacja, rzetelna analiza), jak również problemów strukturalnego rozpoznawania obrazów (ograniczona moc opisowa gramatyk łańcuchowych, generacja błędnych struktur, duża liczba strukturalnych prototypów, interakcja rozpoznawania i segmentacji, wybór właściwego obszaru do analizy).

Zaproponowane podejście do automatycznej analizy dokumentów jest w dużym stopniu uniwersalne: schemat przebiegu analizy obrazu jest niezależny od rodzaju aplikacji, natomiast model mapy, zależny od typu dokumentu (syntaktyka), może być w prosty sposób zaadaptowany dla celów innej aplikacji.

Przedstawiona struktura wnioskująca wraz z klasyfikatorami została zaimplementowana w postaci modułu półautomatycznej konwersji map, co pozwoliło na jej 70% automatyzację.

Information for Authors

The journal *STUDIA INFORMATICA* publishes both fundamental and applied Memoirs and Notes in the field of informatics. The Editors' aim is to provide an active forum for disseminating the original results of theoretical research and applications practice of informatics understood as a discipline focused on the investigations of laws that rule processes of coding, storing, processing, and transferring of information or data.

Papers are welcome from fields of informatics inclusive of, but not restricted to *Computer Science*, *Engineering*, and *Life and Physical Sciences*.

All manuscripts submitted for publication will be subject to critical review. Acceptability will be judged according to the paper's contribution to the art and science of informatics.

In the first instance, all text should be submitted as hardcopy, conventionally mailed, and for accepted paper accompanying with the electronically readable manuscript to:

Dr. Marcin SKOWRONEK
Institute of Informatics
Silesian University of Technology
ul. Akademicka 16
44-100 Gliwice, Poland
Tel.: +48 32 237-12-15
Fax: +48 32 237-27-33
e-mail: marcins@ivp.iinf.polsl.gliwice.pl

Manuscript Requirements

All manuscripts should be written in Polish or in English. Manuscript should be typed on one side paper only, and submitted in duplicate. The name and affiliation of each author should be followed by the title of the paper (as brief as possible). An abstract of not more than 50 words is required. The text should be logically divided under numbered headings and subheadings (up to four levels). Each table must have a title and should be cited in the text. Each figure should have a caption and have to be cited in the text. References should be cited with a number in square brackets that corresponds to a proper number in the reference list. The accuracy of the references is the author's responsibility. Abbreviations should be used sparingly and given in full at first mention (e.g. "Central Processing Unit (CPU)"). In case when the manuscript is provided in Polish (English) language, the summary and additional abstract (up to 300 words with reference to the equations, tables and figures) in English (Polish) should be added.

After the paper has been reviewed and accepted for publication, the author has to submit to the Editor a hardcopy and electronic version of the manuscript.

It is strongly recommended to submit the manuscript in a form downloadable from web site http://sun10.iinf.polsl.gliwice.pl/publikacje.html.

To subscribe: *STUDIA INFORMATICA* (PL ISSN 0208-7286) is published by Silesian University of Technology Press (Wydawnictwo Politechniki Śląskiej) ul. Akademicka 5, 44-100 Gliwice, Poland, Tel./Fax +48 32 237-13-81. 2000 subscription price: US$30 per number.

Wydano za zgodą
Rektora Politechniki Śląskiej

Druk z makiet opracowanych przez Autora

Wydawnictwo Politechniki Śląskiej
ul. Akademicka 5
44-100 Gliwice
tel./fax (0-32) 237-13-81
Dział Sprzedaży i Reklamy (0-32) 237-18-48

www.wydawnictwo.polsl.gliwice.pl
wydawnictwo@polsl.gliwice.pl

Nakł. 100 + 83	Ark. wyd. 16,5	Ark. druk. 13,25	Papier offset. kl. III 70 x 100, 80 g.
Oddano do druku 23.11.2000 r.		Podpis. do druku 23.11.2000 r.	Druk ukończ. w grudniu 2000 r.

Fotokopie, druk i oprawę wykonał „ROLEK", Gliwice, ul. Kazimierza Wielkiego 4

BIBLIOTEKA GŁÓWNA
Politechniki Śląskiej
P. 4193/00

Druk: Drukarnia Gliwice, ul. Zwycięstwa 27, tel. 230 49 50