Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research
Report
RR-92-02

ΠODA: The Paper Interface to ODA

**Andreas Dengel, Rainer Bleisinger, Rainer Hoch,
Frank Hönes, Frank Fein, Michael Malburg**

February 1992

**Deutsches Forschungszentrum für Künstliche Intelligenz
GmbH**

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, SEMA Group, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- ❑ Intelligent Engineering Systems
- ❑ Intelligent User Interfaces
- ❑ Computer Linguistics
- ❑ Programming Systems
- ❑ Deduction and Multiagent Systems
- ❑ Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Friedrich J. Wendl
Director

Π_{ODA}: The Paper Interface to ODA

Andreas Dengel, Rainer Bleisinger, Rainer Hoch, Frank Hönes, Frank Fein, Michael Malburg

DFKI-RR-92-02

A short version of this report will be published in IEEE Computer, Special Issue on Document Image Analysis Systems, June 1992.

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-9003 0).

Π_{ODA}: The Paper Interface to ODA[1]

ANDREAS DENGEL, RAINER BLEISINGER, RAINER HOCH,
FRANK HÖNES, FRANK FEIN, MICHAEL MALBURG

German Research Center for Artificial Intelligence (DFKI)
Project ALV
P.O. Box 2080, 6750 Kaiserslautern, FRG
phone: (+49 631) 205 3215
email: dengel@dfki.uni-kl.de

Abstract

In the past, many people have proclaimed the vision of the paperless office, but today offices consume more paper documents than ever before. As computer technology becomes more and more important in daily practice of modern offices, intelligent systems bridging the gap between printed documents and electronic ones, called paper-computer-interfaces, are required.

In this report our model-based document analysis system ΠODA is discussed in detail. Basic ideas of the ODA standard for electronic representation of office documents are the foundation of our document model. Moreover, different knowledge sources essential for the analysis of business letters are incorporated into the ΠODA model. The system comprises all important analysis tasks. Initially, *layout extraction* includes a necessary low-level image processing and segmentation to investigate the layout structure of a given document. While *logical labeling* identifies the logical structure of a business letter, *text recognition* explores the captured text of logical objects in an expectation-driven manner. By this way, word hypotheses are generated and verified using a dictionary. Finally, a *partial text analysis* component syntactically checks well-structured text objects, primarily the recipient of a letter.

As output, ΠODA produces an ODA conforming symbolic representation of a document originally being captured on paper. Now, the document is available for any further automatic processing such as filing, retrieval or distribution.

The inherent modularity of our system, however, allows a reuse of knowledge sources and constituents of the architecture in other document classes such as forms or cheques. Additionally, ΠODA is an open and flexible system: improved and new analysis methods can be integrated easy without modifying the overall system architecture.

Keywords

Document Analysis, Document Representation, Layout Extraction, Logical Labeling, Text Recognition, Partial Text Analysis, Lexical Knowledge, ODA.

[1] This work has been supported by the Germany Ministry for Research and Technology BMFT under contract ITW 9003 0.

CONTENTS

1 Introduction and System Overview

In the last years, many people have proclaimed that computer technology is going to provide an office environment where paper would be obsolete. Despite the efforts that have been made, it appears, however, that offices produce more paper than ever before ([Schäfer & Fröschle 86], [Hough 89]). One obstacle to realize the *paperless office* is the fact that standardization of existing systems and means for external communication in connection with high-speed networks are still lacking. But the biggest handicap lies in the also lacking capabilities of how getting existing paper documents into a computer (paper-computer interfaces) and moreover, how to represent and process them once they have been acquired.

Office tasks are distributed processes that comprise a sequence of information processing steps in which different clerks are involved. As a means for communication, *documents* play the central role for exchanging any kind of *information*.

Generally, information may be of different modes, for example, text and graphics as well as raster images, formulas, and tables. However, documents are not only characterized by their individual contents, but also by a logical organization into components that relate to a human perceptible meaning, called *logical objects*, e.g. the *author* of an article or the *recipient* of a letter. This logical structuring is done in order to enhance the comprehension of a document´s contents. For visualization (displaying or printing), the information is formatted by defining corresponding two-dimensional presentation aspects such as positions, shapes, and styles. The resulting layout structure, verified by a certain block order, line spacing, number of columns, etc., underlines the originator´s intention of logical document organization.

Usually, a document is locally generated by a given (in-house) document processing system. Document interchange, by contrast, often takes place between different and heterogeneous systems. In order to ensure that the processing of the document´s components at the receiving end is consistent with that of the originator, a fundamental common understanding of the structure of a document is necessary. Consequently, using a well-known and accepted standard for the representation of structured documents might be a promising vehicle solving all these problems of exchange. In IEEE Computer of October 1985 ([Horak 85]) the international standard representation for office documents *ODA* (Office Document Architecture) [ISO8613] has been presented. This standard provides constructs for the representation of documents enabling their interchange between open systems.

In the daily practice of modern offices, many companies rely on converting existing as well as incoming paper documents into an electronic representation that allows for information management including content-based retrieval and distribution. If it would be possible to convert or transform printed information into an electronic standard representation, it is not relevant whether the information is transmitted by electronic means or by paper. As a consequence, both, paper documents and electronic ones are represented by the same formalism, and may be further processed by the same software tools.

In this report, we present our document analysis system ΠODA (**P**aper **I**nterface to **ODA**). ΠODA is a model-driven system bridging the gap between paper and computer. Furthermore, the system is based on the ODA platform according to the ideas described above. To support and to improve document analysis, various knowledge sources such as typesetting knowledge, geometric knowledge, and lexical or syntactic knowledge are integrated. Exemplarily and rather pragmatical, the system is devoted to a particular class of documents, the domain of *business letters*. The inherent modularity of the system, however, allows a reuse of knowledge sources and constituents of the architecture in other document classes such as forms or cheques. Figure 1 gives an impression of our system including the underlying document model. Note that this architecture will be refined in later chapters.

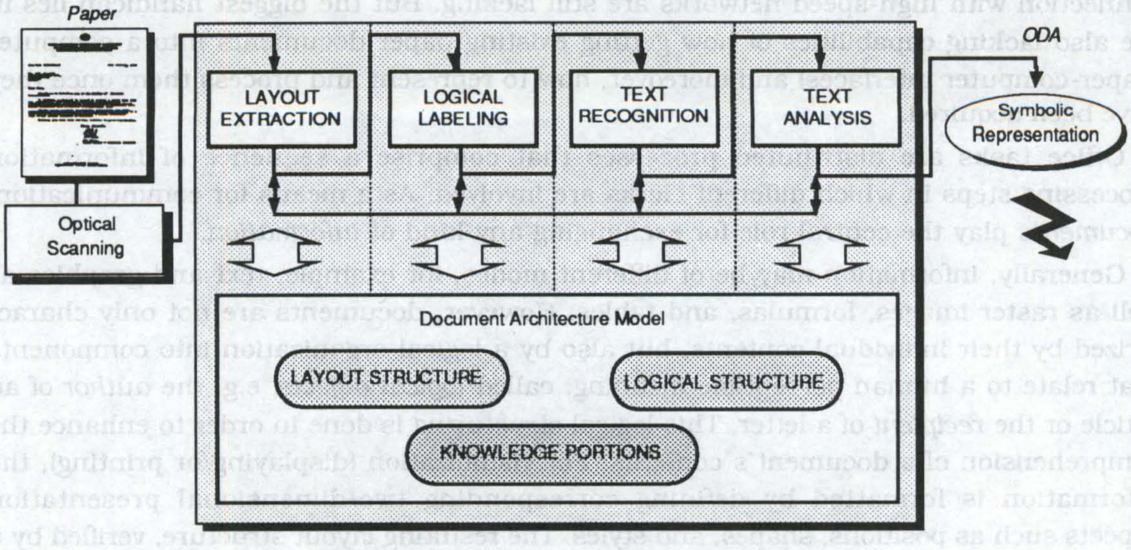


Figure 1: System architecture of ΠODA.

The entire system comprises several interlocked phases of analysis: *Layout extraction* includes low-level processing such as image capturing, skew angle adjustment and segmentation to investigate the layout structure of the given document ([Dengel 92]). While *logical labeling* identifies the logical structure of a business letter ([Dengel 92]), *text recognition* explores the captured text of logical objects in an expectation-driven manner. By this way, word hypotheses are generated and verified using a dictionary ([Hönes et al 90]). Finally, a *partial text analysis* component syntactically checks some simply structured objects (sender, recipient, date). As output, ΠODA produces a symbolic representation of a business letter conforming to the ODA standard.

Our report is organized as follows: Chapter 2 considers international standards for the representation of electronic documents and motivates why it is advantageous to use ODA for document analysis. While Chapter 3 gives a brief survey of ODA and explains the corresponding document architecture, Chapter 4 discusses the similar document modeling of ΠODA and corresponding extensions with respect to document analysis. Subsequently, Chapter 5 exhibits more details of our system architecture and processing steps. The following four chapters describe all phases of analysis in detail: layout extraction, logical labeling, text recognition, and text analysis. These phases are illustrated by a general example. Then, Chapter 10 takes account of similar approaches and related work. Finally, Chapter 11 concludes the report and points to our current research activities.

2 STANDARDS FOR THE REPRESENTATION OF STRUCTURED DOCUMENTS

In the last years the transfer and processing of electronic documents have taken a central role in the field of office information systems. Hence, standards become more and more important being a prerequisite for a successful exchange of documents between heterogeneous systems.

Primarily, two international standards in the domain of office information systems have been developed and used in several (prototypical) systems: the *Office Document Architecture and Office Document Interchange Format* (ODA/ODIF)[2] (cf. [ISO8613], [Horak 85]) and the *Standard Generalized Markup Language* (SGML) (cf. [ISO8879], [Bryan 89]).

Both standards provide formalisms for document structure representation, in especially, for defining the logical structure of a document. This logical structure divides the contents of a document (text, graphics, images) into logical entities that are associated with an author's intellectual meaning. For example, a business letter may be divided into logical objects such as sender, recipient, subject, and body. In SGML, this formalism is described through so-called document type definitions (DTD's), while in ODA it is designated by a generic logical structure of a document. The concept of logical structure will be detailed in the next chapter.

Another possibility to consider the organization of a document is by its layout. The layout structure is determined by hierarchically nested rectangular blocks. These may be entire pages, graphic frames, image frames, and text frames, while the latter ones may be further subdivided into lines, words and characters. Both structures, layout as well as logical, are strictly hierarchical and express two different but complementary views to the contents of a document.

In contrast to ODA, however, SGML does not support a description of the layout structure of a document for reasons of simplicity and universality. SGML is designed for the representation of any kind of structured text. For instance, SGML is typically used in a publishing environment, where an author logically marks up a document's components and the publisher performs all future processing such as copy-editing, proof-reading and production, including the final distribution. In this closed application area, standardized layout characteristics are less important. In contrast, the scope of ODA covers office documents (business letters, reports, forms) in particular. An office environment requires that documents may be sent to arbitrary recipients allowing for an automatic reproduction and interactive modifications of the document at the receiving end.

To complete the discussion about standards, a third and more commercial standard, named *EDIFACT* (Electronic Data Interchange for Administration Commerce and Transport) (cf. [ISO9735], [Frank 91]) should be considered here. EDIFACT specifies the structure and formal semantics of a data stream for exchanging fixed and predefined types of business letters, called *message types*, and enables further processing of the message content. Each message type description includes optional or mandatory

[2] ODIF is a convention how ODA structures are mapped into a corresponding data stream for electronic exchange. Because this article is concerned with aspects of document modeling, we neglect ODIF in the following.

segments (records), data element groups, or data elements respectively. Such elements represent logical components of a document; any layout information is taken away. At moment, only two message types for business letters, invoice and order, have been standardized ([Frank 91]).

Note that a multitude of in-house styles have also been developed (e.g. Interscript, Scribe, DCA), but a discussion is beyond the scope of this report (for details see also [Joloboff 89] and [Quint 89]).

Since low-level routines of document image analysis mainly focus on layout and presentation aspects (e.g. skew adjustment, block segmentation), we base the document model of our analysis system on the ODA platform, but we enhance the standard to the requirements of document analysis as needed. Moreover, logical elements identified and captured by EDIFACT message types have a strong influence on the design of our logical model of business letters.

The next chapter gives a short introduction to the concepts of ODA emphasizing crucial points with respect to our ΠODA system.

3 DOCUMENT ARCHITECTURE MODEL IN ODA

One of the most characteristic features of the document architecture model of ODA [ISO8613] is a strict separation between the contents of a document and its structural representation. Consequently, the notion of *structure* is a key concept of ODA (see also [Brown 89]).

In ODA, there are two distinct, but complementary structures of a document, the *layout structure* and the *logical structure*. Both structures are represented by a tree whose nodes correspond to document components (layout objects, logical objects). The leaves of each tree are associated with specific *content portions* of a document. An object that is not subdivided into smaller objects (i.e. a leaf of a tree) is called a *basic object* in ODA. All other objects are called *composite objects*, in especially the document root of each tree. By this way, ODA provides a hierarchical and object-oriented document model.

ODA defines the following types of layout objects in the document architecture:

- *block:* a basic layout object corresponding to a rectangular area on the presentation medium containing a portion of the document content;
- *frame:* a composite layout object corresponding to a rectangular area containing one or more frames or blocks;
- *page:* a basic or composite layout object corresponding to a rectangular area on the presentation medium, or containing one or more frames or blocks respectively;
- *page set:* a set of one or more page sets or pages;
- *document layout root:* the highest level object in the hierarchy of the layout structure.

For logical objects, the classification is less concrete comprising the types basic logical object, composite logical object, and document logical root. Hence, logical objects (e.g. of a business letter) are strictly application-dependent (e.g. sender, recipient, ...).

In a document, layout objects as well as logical objects can often be classified into groups of similar objects, the so-called object classes. An object class is comparable to the well-known class concept in object-oriented programming paradigm. Such a class can be considered as a specification of characteristics, a pattern, that is common to its members. The specification includes methods for creating new objects, methods to determine the values of the object attributes, and methods to control the consistency among objects.

Using these object classes, the logical structure of similar documents can be modelled by a set of logical object classes. Analogous, their layout structure may be composed of a set of layout object classes. This approach is called the *generic structure concept.* Generic structures (generic logical structure, generic layout structure) provide a means for defining document classes or "styles" that define the types and combinations of objects allowed.

In ODA, the structures that are particular to a given document instance are named *specific logical structure* and *specific layout structure*. Consequently, the generic logical structure represents a set of rules from which specific logical structures are derived during the editing process, while the generic layout structure comprises rules from which specific layout structures are derived during the formatting process. Figure 2 shows the example of a business letter and both specific structures.

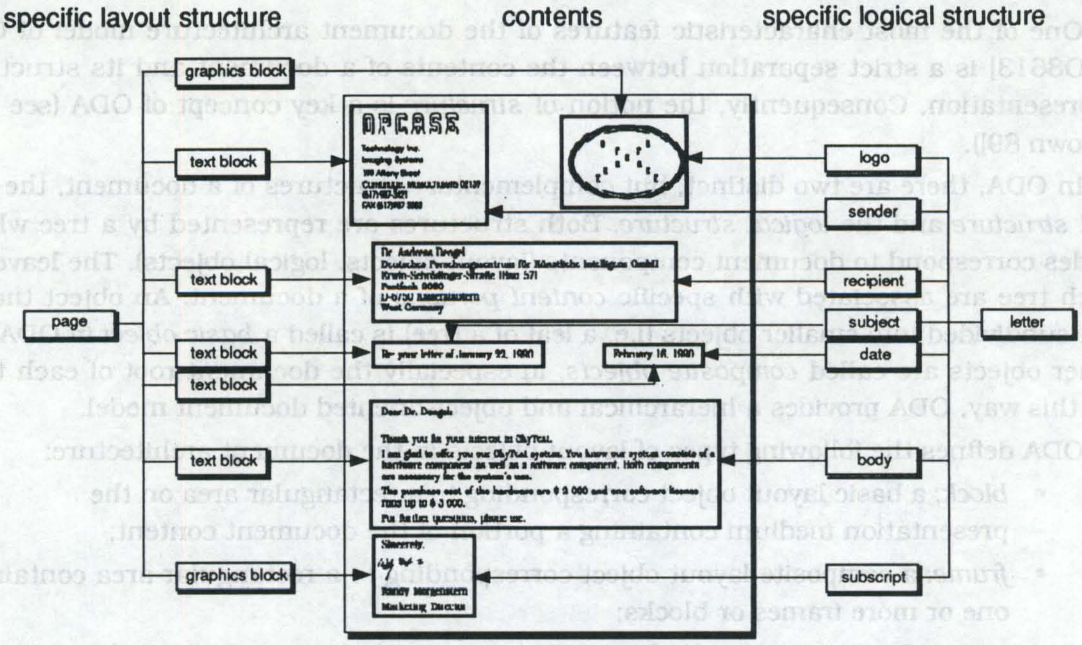

specific layout structure
contents
specific logical structure
graphics block
text block
text block
page
text block
text block
text block
graphics block
OPCASE
Technology Inc.
Imaging Systems
100 Albany Street
Cambridge, Massachusetts 02139
(617)497-3621
FAX (617)497 3880
Dr. Andreas Dengel
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Erwin-Schrödinger-Straße (Bau 57)
Postfach 2080
D-6750 Kaiserslautern
West Germany
Re: your letter of January 22, 1990
February 18, 1990
Dear Dr. Dengel:
Thank you for your interest in SkyTeal.
I am glad to offer you our SkyTeal system. You know our system consists of a hardware component as well as a software component. Both components are necessary for the system's use.
The purchase cost of the hardware are $ 2 000 and one software license runs up to $ 3 000.
For further questions, please call.
Sincerely,
Randy Morgenstern
Marketing Director
logo
sender
recipient
subject
date
letter
body
subscript

Figure 2: A specific business letter represented in ODA (simplified).

All objects of a document are supplied with specific characteristics or properties, known as *attributes* in ODA. Attributes control document generation, in particular its process of editing, layouting and imaging (processing model). Each attribute is identified by its name and has a value that describes the property or also identifies constituents (construction and relationship attributes).

The set of attributes associated with a document as a whole can be categorized into *layout attributes*, *logical attributes* and *shared attributes*. When applied to object descriptions, these attributes are either mandatory, non-mandatory (i.e., optional), or defaultable (i.e., the attribute need not be specified for the constituent and the corresponding value can be derived using a defaulting mechanism). Layout attributes are further classified into property attributes (positions, dimensions), formatting attributes, imaging attributes (imaging order, transparency, color, page position), and with respect to content type, into presentation attributes. Logical attributes define e.g. protection rights and layout styles. Shared attributes can be specified from both logical objects and layout objects; they are subdivided into

- identification attributes (object type, object identifier, object class identifier),
- relationship attributes (subordinates, content portions, presentation styles),
- content architecture class attributes (content architecture class, content type),
- miscellaneous attributes (user-visible name, comments, bindings), and
- construction attributes (generator-for-subordinates, content generator).

The latter ones are responsible for controlling the generation of subordinate objects and for controlling the generation of content.

Additionally, there are attributes which are attached to so-called *layout styles* (block alignment, fill order, offset, concatenation, indivisibility, separation, etc.), *presentation* *styles* (dependent from content architecture) and content portions (identification attributes, common coding). Presentation styles affect the layout and imaging of the content associated with basic objects and hence are content type specific. In contrast, layout styles only affect the layout of objects, not their content. While presentation styles control the mapping of content portions into layout blocks in a first stage, layout styles place these blocks into appropriate pages and frames onto the presentation medium in the second stage. For that purpose, there are no conflicts in processing both styles.

One important type of attributes should be described in more detail, namely the *generator-for-subordinates* attribute. This attribute defines how an object of a document is built up from subordinate objects and combinations of these subordinates, e.g. a text line may be built up from several words, or the recipient is built up from name, street, city, and country. In addition, the generator-for-subordinates specifies an ordering among these subordinates. It can best be thought of as *construction mechanism*. Subordinates of an object may be optional (OPT), required (REQ), repetitive (REP, i. e., one or more occurrences), or optional repetitive (OPT REP). The relationship between subordinates may be expressed as a sequence (SEQ, sequential order), an aggregate (AGG, any order) or a choice (CHO, i. e., only one subordinate of a group occurs). For example, the body of a business letter can be a sequence of the logical objects "salutation" (required), "paragraph" (repetitive), "regards" (optional), and "signature" (required):

SEQ (REQ (salutation), REP (paragraph), OPT (regards), REQ (signature))

As mentioned above, the ODA standard specifies that only basic objects (logical as well as layout), can be associated with content portions of a concrete document. These content portions may have a more detailed internal structure depending on the type of content.

The rules for processing different kinds of document contents are known as *content architectures*. Currently, ODA defines three types of architectures: character content (ASCII code), raster graphics content (images) and geometric graphics content (graphics primitives). The contents of a basic logical object or a basic layout object is structured according to only one content architecture.

The character content architecture comprises presentation attributes and control functions that control the form and positioning of all ASCII characters. For instance, characters are placed left-to-right starting at the top of a layout block and progressing downwards. Other arrangements are allowed. Relevant content attributes are character path, line progression, alignment, initial offset, orphan size, and so on.

A raster graphics content portion represents a two-dimensional pictorial image in the form of a rectangular two-dimensional area of pixels. A large set of attributes is used to control the presentation of image information, such as line progression, path information, origin, dimensions, clipping region, etc.

At last, the geometric graphics architecture describes graphical primitives like lines, rectangles, circles. It is entirely based on the Computer Graphics Metafile (CGM) standard.

The following chapter shows how the philosophy and concepts of ODA are pursued for the analysis of business letters.

4 Document Architecture Model in ΠODA

Document analysis is a transformation problem, whereby the entire image of a digitized document page has to be explored and converted into a corresponding symbolic representation. The effectiveness of model-based analysis depends on the certainty and completeness of the underlying model. Because any document is characterized by its contents and its internal organization, we have developed a document architecture model that obeys the ideas of ODA, but is extended in some parts that are fundamental for document analysis.

Primarily, ODA has been developed as a guideline to represent and interchange document structures that are electronically generated or at least electronically available. However, the formalism can also be related to other media that capture documents in a two-dimensional structural representation, e.g. paper or microfiches.

While generating an electronic document in ODA, the entered content portions are related to specific logical objects (editing process) as well as to specific layout objects (formatting process). The first task is done manually by the originator of the document. The second task is performed automatically and is intended to produce a paper document.

In ΠODA, by contrast, the problem is to transform document structures from a non-electronic medium, like paper, to the electronic medium. Here, following questions arise: how to extract automatically specific layout objects from a given document image, how to identify the specific logical objects of the document and, finally, how to relate specific layout and logical objects.

For that purpose, various knowledge sources are incorporated into the document architecture model of ΠODA. The knowledge sources and their usage are:

- formatting knowledge is used in form of distance parameters during layout extraction,
- geometric knowledge describing logical object arrangements employed for logical labeling,
- lexical knowledge is needed in text recognition and text analysis tasks,
- syntactic knowledge is taken for partially text analysis of single logical objects.

Figure 3 schemes the document architecture model in ΠODA inclusive the knowledge sources attached .

The entire model in ΠODA is composed of a two-layered architecture:

- a *meta layer* providing a framework for defining object classes of the several structural views, namely, layout view, and logical view;
- a *generic document class layer* that reflects the generic structure of a specific document class, such as of business letters.

The following sections focus on the document architecture model used in ΠODA by giving detailed information about the facilities for different document structure representations. First, the meta layers of the two views are described. Second, the patterns for layout objects as well as for logical objects are presented in the context of business letters. Thereby, the generic structural representation of business letters is proposed.

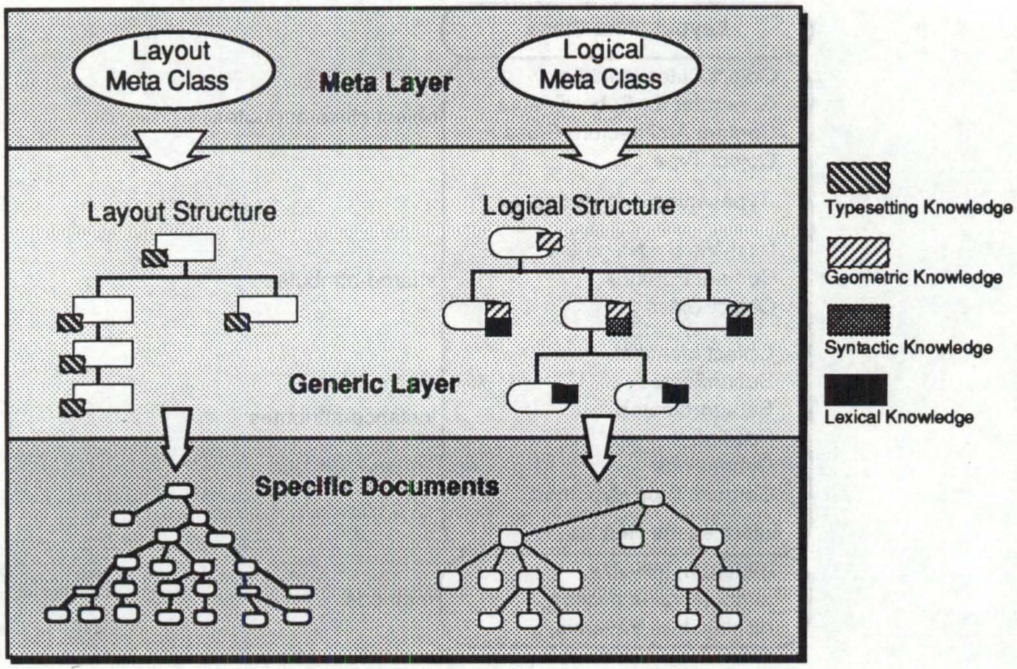


Figure 3: Document Architecture Model in ΠODA.

4.1 Meta Layer

In the former discussion about ODA, formalisms for describing object classes including different sets of attributes and special construction operators have been introduced. Therefore, the meta layer of the document architecture in ΠODA provides mechanisms for defining object classes, for relating object classes to another, and for attaching additional knowledge which is required for an analysis. According to the two structural views used in ΠODA, the following paragraphs describe layout as well as logical aspects. For both, layout and logical object class generation, the meta class concept of the object-oriented paradigm is transferred to the ΠODA-model.

4.1.1 Meta Layer Layout Description

At least, the attribute specification of the layout meta class has to cover relevant features which are required for representing a document´s presentation with respect to document image analysis and document representation in an ODA conforming manner. Consequently, a classification of these attributes is useful.

All attributes are applicable either to object classes or object instances, or to both. While creating an object class as subclass of the meta class, all attributes are inherited automatically and only values for object-class-attributes and shared-attributes may be defined. In contrast, by instantiating an object class, only the instance-attributes as well as the shared-attributes including the default values are inherited. As in the object-oriented programmmg, values of the object-class-attributes are only readable. After instantiation the instance-attributes may be filled and the values of the shared-attributes considered as default may be changed.

Figure 4 shows the layout meta class of the ΠODA-model containing typical attributes. Moreover, basic methods, e.g. for creating objects or accessing values, are indicated.

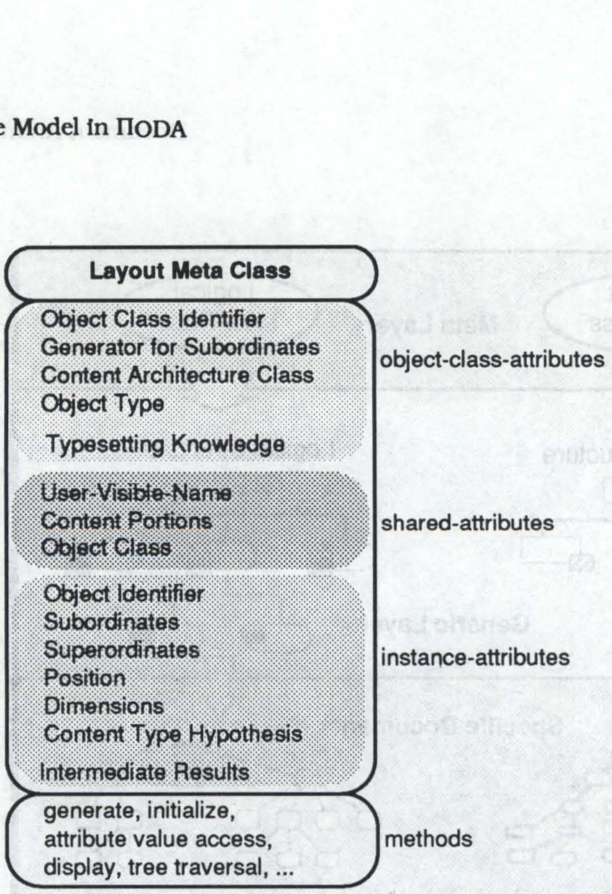


Figure 4: Layout Meta Class.

Most of the attributes are *ODA standard attributes* for layout objects, e.g. object class identifier, position and dimensions which are mandatory. Additionally, any important optional ODA standard attributes, such as generator-for-subordinates, are incorporated.

Some attributes, however, are additional defined as *ΠODA extensions*. Once, there is an attribute to store intermediate results of the analysis, but also attributes are included that enable an attachment of knowledge portions assisting the analysis process. The knowledge portions are object dependent information used from several experts for expectation driven analysis. Additional instance attributes are introduced, one for representing the relationship *superordinates* to refer to the parent instance — inverse to *subordinates* — and one for showing the actual *content type hypothesis* of layout objects. The explicit representation of the superordinates relationship facilitates any analysis tasks, especially bottom up oriented strategies. The content type hypothesis is needed as a flag indicating the hypothesized mode of information, text or non-text.

For the ΠODA attribute *intermediate results*, a list of name-value pairs may be specified. The name identifies the type of result, e.g. character hypotheses (see Chapter 8.1), and the value quantifies the corresponding data. In this way, object related results depending on algorithms used for document analysis are available for the whole analysis task.

Furthermore, a list of name-value pairs can be associated with the ΠODA attribute *typesetting knowledge*. For example, the maximal horizontal as well as the maximal vertical permissible distance of layout objects may be defined (cf. Chapter 6).

4.1.2 Meta Layer Logical Description

Analogous to the definition of the layout meta class, however, focussing on the logical view, attributes for the logical meta class are specified. The resulting description is shown in Figure 5.

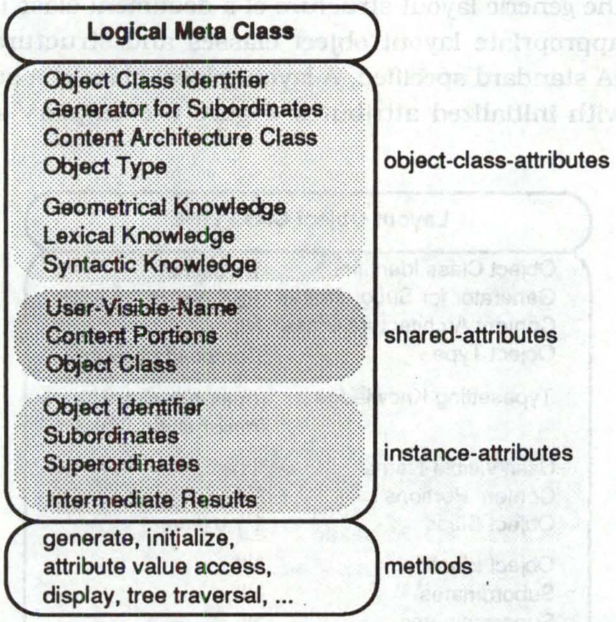


Figure 5: Logical Meta Class.

The set of ΠODA specific attributes is treated in the same manner as those in the layout meta class. For example, intermediate results may be relations from logical objects to layout objects, stored explicitly in the object during logical labeling.

Finally, three attributes for knowledge attachment are incorporated. Geometric knowledge is used by logical labeling to identify logical objects. This knowledge comprises, for example, that the recipient is located in the upper right part of a business letter (for details see Chapter 7). Lexical knowledge corresponds to groups or clusters of words, for example, including all names of employees that are possible recipients of a business letter sent to a company. This knowledge is mainly used for verifying text recognition results (cf. Chapter 8). Syntactic knowledge is concerned with syntactic structures of textual content of logical objects and assists text analysis and verification. For instance, text within the logical object recipient can be described by a simple context free grammar with attributes (cf. Chapter 9).

ODA standard attributes of both, layout and logical, are explained in the ISO standard [ISO8613] in detail.

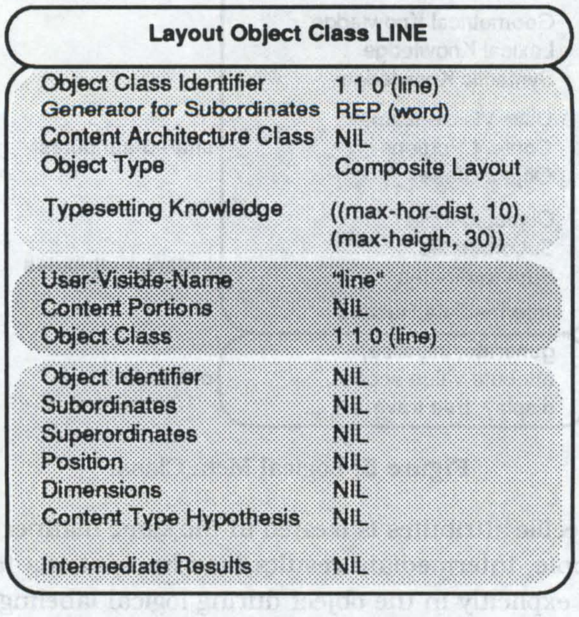
4.2 Document Class Layer

In this section, a concrete document class model for business letters is established. In particular, layout and logical object class descriptions on the basis of their corresponding meta classes are introduced. The resulting architecture serves as a basis for an automatic transformation of scanned letter images into an ODA conforming format.

4.2.1 Layout Object Classes

In ΠODA, the document class layout part is used for directing the analysis, especially document segmentation following a model-based approach. In this sense, the segmentation creates a specific layout structure according to a generic layout structure.

For establishing the generic layout structure of a document class (e.g. business letter), one has to define appropriate layout object classes and structurally compose them, similar as in the ODA standard specified. A layout object class is a subclass of the ΠODA layout meta class with initialized attributes. Figure 6 exemplary shows layout object class "LINE".



Layout Object Class LINE	
Object Class Identifier	1 1 0 (line)
Generator for Subordinates	REP (word)
Content Architecture Class	NIL
Object Type	Composite Layout
Typesetting Knowledge	((max-hor-dist, 10), (max-heigth, 30))
User-Visible-Name	"line"
Content Portions	NIL
Object Class	1 1 0 (line)
Object Identifier	NIL
Subordinates	NIL
Superordinates	NIL
Position	NIL
Dimensions	NIL
Content Type Hypothesis	NIL
Intermediate Results	NIL

Figure 6: Layout Object Class "LINE".

As mentioned above, the generator-for-subordinates provides facilities for a structural combination of object classes. In this sense, an entire generic layout structure for business letters can be attained (cf. Figure 7).

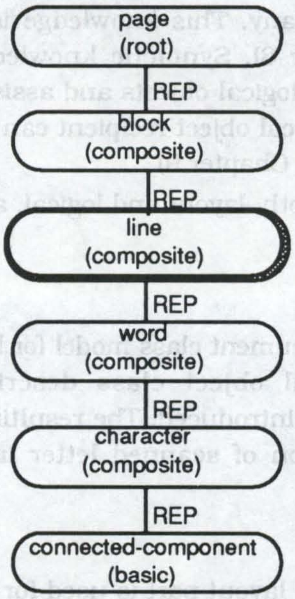


Figure 7: Generic layout structure for document class *business letter.*

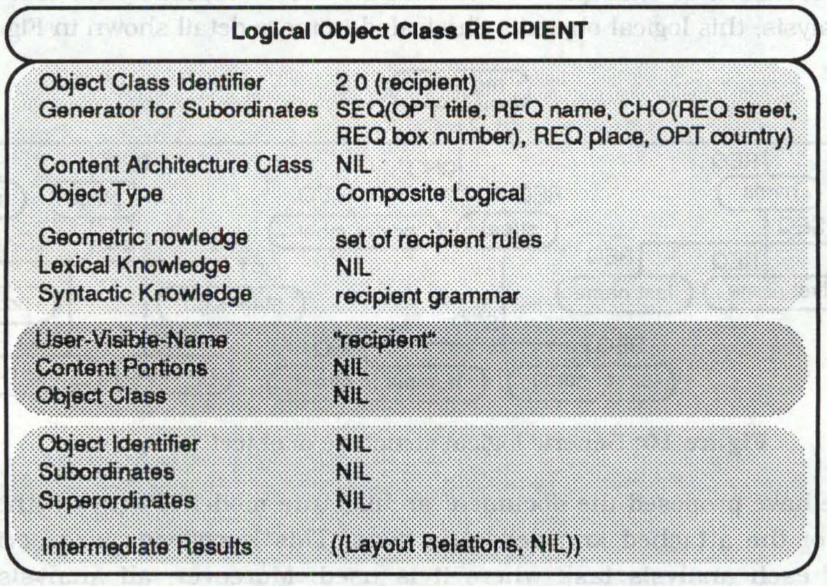
For image analysis of a given document, the declarations of ODA are not specific enough and the objects defined are to abstract. Thus, it is necessary to redefine the layout structure of ODA by objects which correspond to the layout primitives resulting from document image processing, e.g. characters and connected components of image pixels.

4.2.2 Logical Object Classes

As in the layout, a generic logical structure and the containing object classes are applied for both, directing model-based analysis and representation of a given document. That means, the specific logical structure is successively, but partially generated during logical labeling (see Chapter 7) according to the generic logical structure. While performing text analysis and verification, the specific document structure is completed.

The generic logical structure of a document class consists of logical object classes which have to be defined at first. They capture attributive descriptions of logical document components which comprise ODA standard attributes as well as attributes that are needed during the analysis.

Figure 8 shows the logical object class "RECIPIENT" filled with specific values.



Logical Object Class RECIPIENT

Object Class Identifier	2 0 (recipient)
Generator for Subordinates	SEQ(OPT title, REQ name, CHO(REQ street, REQ box number), REQ place, OPT country)
Content Architecture Class	NIL
Object Type	Composite Logical
Geometric nowledge	set of recipient rules
Lexical Knowledge	NIL
Syntactic Knowledge	recipient grammar
User-Visible-Name	"recipient"
Content Portions	NIL
Object Class	NIL
Object Identifier	NIL
Subordinates	NIL
Superordinates	NIL
Intermediate Results	((Layout Relations, NIL))

Figure 8: Logical Object Class "RECIPIENT".

Within the document architecture model of ΠODA, we have specified the following logical object classes for business letters. First of all, a grouping in three parts is performed: the *letter thematic part* contains the subject and the body of a letter; in the *sender specific part* objects such as sender, sender short form, company logo, and company specific printings are incorporated; the *procedure relevant part* is composed of relation data, e.g. "your sign" or "our sign", date, and the recipient. These object classes are structurally related and constitute the generic logical structure of the document class "business letter" (see Figure 9).

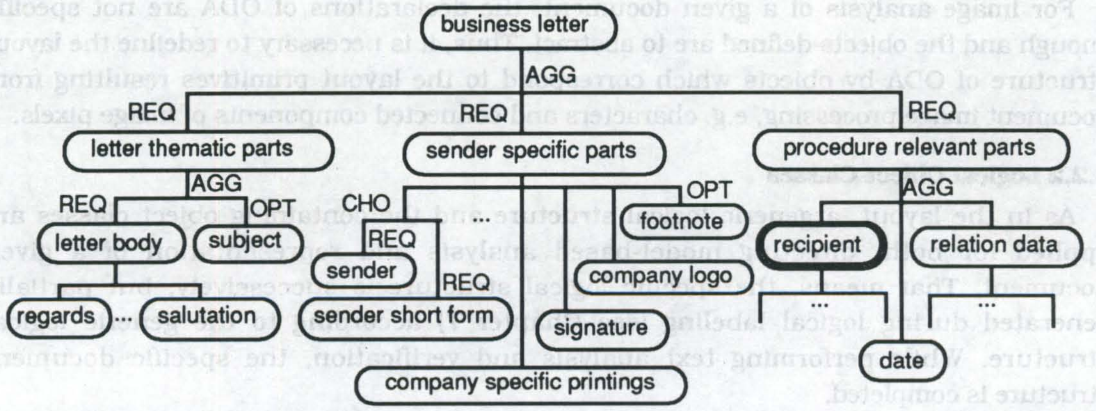


Figure 9: Generic logical structure for document class *business letter* (incomplete).

Any logical object class, such as *letter body* or *recipient*, may be further subdivided into basic logical objects. As example, the definition of the logical object class in Figure 8 constitutes the refined logical structure of the recipient. Because in this paper the recipient is focussed as example for the analysis tasks, especially for text recognition and text analysis, this logical object is illustrated in more detail shown in Figure 10.

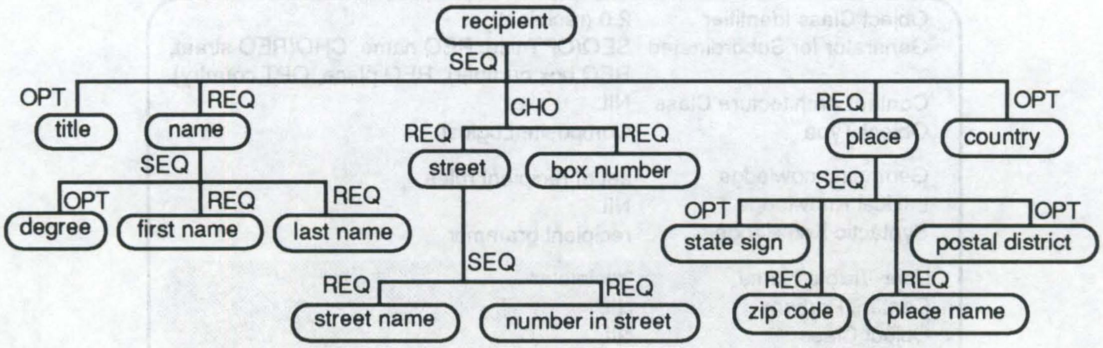


Figure 10: Generic logical structure of object "recipient".

So far we have proposed the document architecture model of ΠODA without detailed discussion of the attached knowledge portions. This knowledge is described in the chapters of each analysis task where it is used. Moreover, all analysis tasks are explained in detail in the following chapters emphasizing how they are influenced by the overall document architecture model.

5 Analysis Overview

So far we have presented our architecture model providing both a layout and a logical view on a document. In the following we describe the distinct phases of analysis and how they make use of the model.

Document analysis can be seen as an automatic *transformation* of printed information into an electronic representation. More than that, it may be viewed as an automatic *generation* of an electronic document by electronically reproducing some non-electronic document. The document architecture model described above provides a variety of knowledge for such a reproduction.

Both, the layout structure and the logical structure are essential knowledge sources as well as an excellent orientation point for model-based document analysis. They enable an understanding of scanned document images and their transformation into an ODA conforming representation.

Within the ΠODA-system, these structural views on a document serve as a basis for several processing components. These are: layout extraction, logical labeling, text recognition and partial text analysis. Figure 11 illustrates these phases indicating their tasks, the underlying document architecture model and specific knowledge sources.

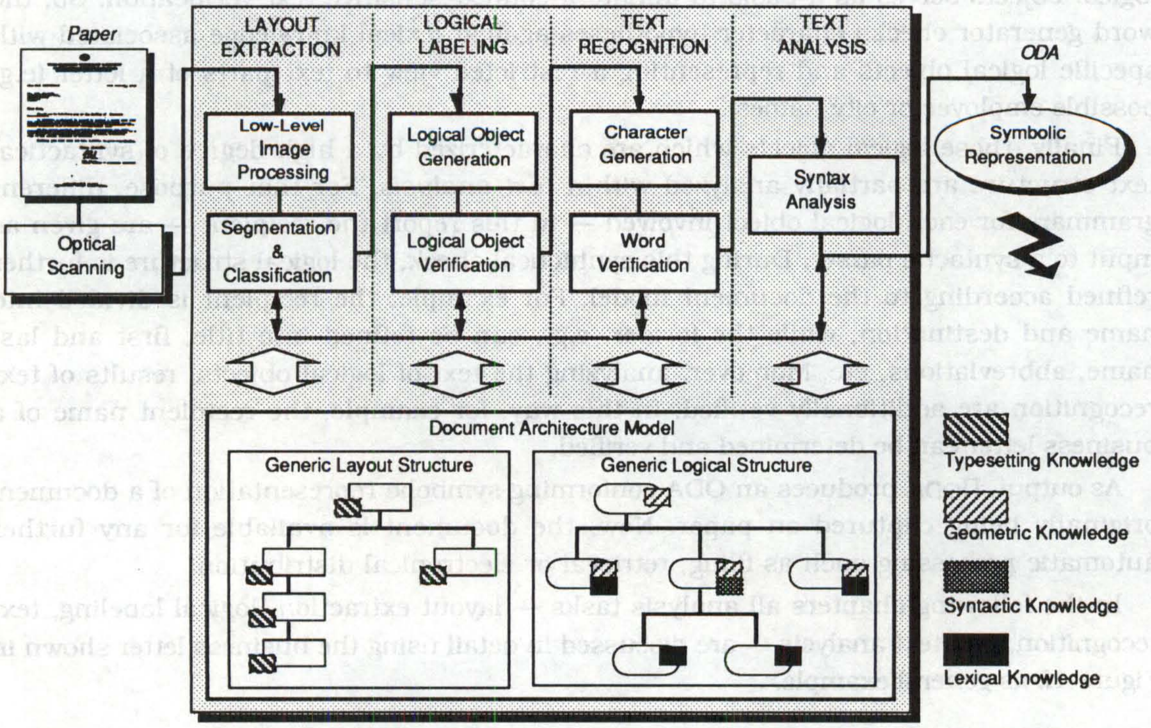


Figure 11: The ΠODA system — analysis steps and architecture model.

In a first step, ΠODA takes the scanned document image and extracts a part-of hierarchy of nested layout objects, such as blocks, words, and characters, on the basis of their presentation on the sheet. For *layout extraction*, low-level image processing is initiated. It includes image capturing and skew adjustment procedures searching for the dominant linear structure in the optically scanned document image to determine the captured skew and to enable an internal correction. In a further step, a top-down procedure stepwise refines the physical structure of the captured information, to separate text from non-text regions, and to map the different physical components into a tree-like nested layout structure.

Subsequently, in a step called *logical labeling*, the layout objects extracted from the image and their compositions are geometrically analyzed to identify corresponding logical objects (e.g. sender, recipient, date in a business letter). This step is divided into a logical object generation and a logical object verification task. First, the layout structure is investigated for object arrangement providing hypotheses about the existence and arrangement of logical objects. For verification, geometric rules that describe local geometric properties of individual logical objects are applied. As a result, the system generates logical objects that are related to one or multiple layout objects.

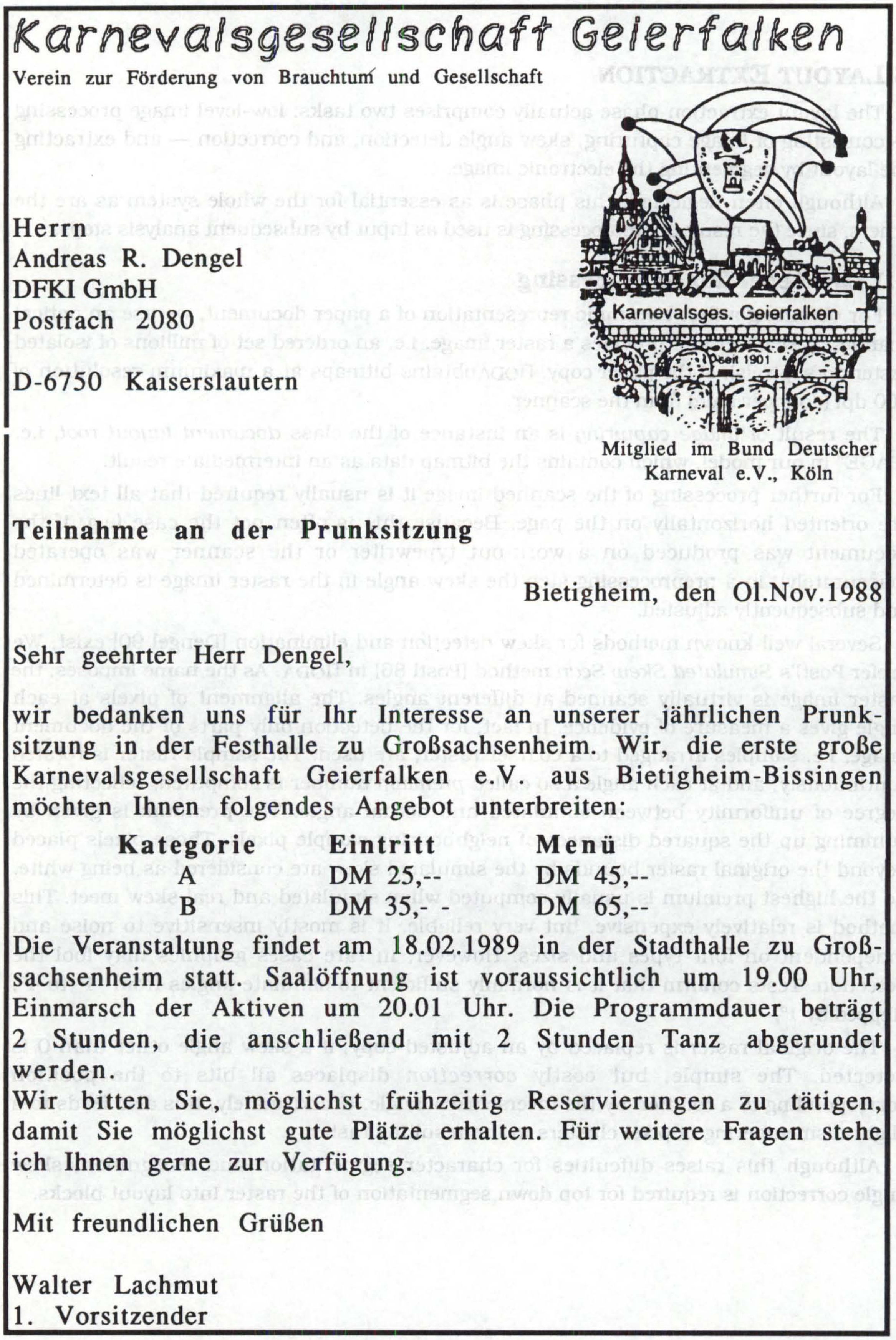
For further processing, PODA is restricted to recognition and analysis of text regions, initially the recipient of a business letter. Therefore, all non-text layout objects are not further considered and stored in a compressed form. The relations of layout objects and logical objects are fundamental representing implicit restrictions of context.

The layout objects "WORD" are the starting point for the phase of *text recognition*. In other words, the image contents of single word-blocks are taken as input to generate a sequence of character hypotheses that might form a word. Subsequently, the character hypotheses are verified using a word candidate generator. In ΠODA, the identification of logical objects serves as a basis to initiate a context-sensitive text verification. So, the word generator checks character hypotheses against lexical knowledge associated with specific logical objects and representing a restricted view to text parts of a letter (e.g. possible employee or city names).

Finally, those logical objects which are characterized by a high degree of syntactical text structure are partially analyzed within *text analysis*. For that purpose, different grammars for each logical object involved — in this report the recipient — are given as input to a syntactic parser. During this syntactical check, the logical structure is further refined according to the document model. For example, the recipient is divided into name and destination, while the former, e.g., can be refined into title, first and last name, abbreviations, etc. Moreover, analyzing the text of logical objects, results of text recognition are additionally verified. In this way, for example, the recipient name of a business letter can be determined and verified.

As output, ΠODA produces an ODA conforming symbolic representation of a document originally being captured on paper. Now, the document is available for any further automatic processing such as filing, retrieval or electronical distribution.

In the following chapters all analysis tasks — layout extraction, logical labeling, text recognition, and text analysis — are discussed in detail using the business letter shown in Figure 12 as general example.



Karnevalsgesellschaft Geierfalken

Verein zur Förderung von Brauchtum und Gesellschaft

Karnevalsges. Geierfalken
seit 1901

Mitglied im Bund Deutscher Karneval e.V., Köln

Herrn
Andreas R. Dengel
DFKI GmbH
Postfach 2080

D-6750 Kaiserslautern

Teilnahme an der Prunksitzung

Bietigheim, den 01.Nov.1988

Sehr geehrter Herr Dengel,

wir bedanken uns für Ihr Interesse an unserer jährlichen Prunksitzung in der Festhalle zu Großsachsenheim. Wir, die erste große Karnevalsgesellschaft Geierfalken e.V. aus Bietigheim-Bissingen möchten Ihnen folgendes Angebot unterbreiten:

Kategorie	**Eintritt**	**Menü**
A	DM 25,--	DM 45,--
B	DM 35,- -	DM 65,--

Die Veranstaltung findet am 18.02.1989 in der Stadthalle zu Großsachsenheim statt. Saalöffnung ist voraussichtlich um 19.00 Uhr, Einmarsch der Aktiven um 20.01 Uhr. Die Programmdauer beträgt 2 Stunden, die anschließend mit 2 Stunden Tanz abgerundet werden.
Wir bitten Sie, möglichst frühzeitig Reservierungen zu tätigen, damit Sie möglichst gute Plätze erhalten. Für weitere Fragen stehe ich Ihnen gerne zur Verfügung.

Mit freundlichen Grüßen

Walter Lachmut
1. Vorsitzender

Figure 12: Business letter example.

6 LAYOUT EXTRACTION

The layout extraction phase actually comprises two tasks: low-level image processing — consisting of image capturing, skew angle detection, and correction — and extracting the layout by segmenting the electronic image.

Although often neglected, this phase is as essential for the whole system as are the others, since the result of preprocessing is used as input by subsequent analysis steps.

6.1 Low Level Image Processing

For obtaining a first electronic representation of a paper document, we use an optical scanner. A scanner just provides a raster image, i.e. an ordered set of millions of isolated raster dots (pixels), of the paper copy. ΠODAobtains bitmaps at a maximum resolution of 300 dpi (dots per inch) from the scanner.

The result of *image capturing* is an instance of the class *document layout root*, i.e. "PAGE" in our model, which contains the bitmap data as an intermediate result.

For further processing of the scanned image it is usually required that all text lines are oriented horizontally on the page. Because this is often not the case (e.g. if the document was produced on a worn-out typewriter or the scanner was operated inaccurately), in a preprocessing step the skew angle in the raster image is determined and subsequently adjusted.

Several well-known methods for skew detection and elimination [Dengel 90] exist. We prefer Postl's *Simulated Skew Scan* method [Postl 86] in ΠODA. As the name imposes, the raster image is virtually scanned at different angles. The alignment of pixels at each angle gives a measure of evidence. In fact, for the detection only parts of the document image, i.e. samples arranged to a coarser raster, are used. The sample raster is rotated continuously, and at each angle a so called *premium* number is computed, reflecting the degree of uniformity between simulated and actual angle. The premium is given by summing up the squared distances of neighbouring sample pixels. Those pixels placed beyond the original raster bounds by the simulated skew are considered as being white. So the highest premium is usually computed when simulated and real skew meet. This method is relatively expensive, but very reliable. It is mostly insensitive to noise and independent on font types and sizes. However, in rare cases graphics may fool the detection. Tests confirm that it is normally sufficient to simulate angles from -4° to 4°, stepped by 1°.

The original raster is replaced by an adjusted copy, if a skew angle other than 0 is detected. The simple, but costly *correction* displaces all bits to the position corresponding to a rotation by the inverse skew angle. Unfortunately, this also leads to a slight dismembering of pixel clusters in the resulting raster.

Although this raises diffculties for character segmentation and recognition, skew angle correction is required for top down segmentation of the raster into layout blocks.

6.2 Segmentation and Classification

The principal task of document layout extraction is to determine nested layout objects like pages, blocks, and lines. Consequently, the entire document can be represented by its layout structure. This function is often referred to as *document segmentation.* Our document architecture model (see Section 5) describes the possible layout classes of ΠODA.

We use two segmentation methods to extract the complete layout from blocks down to connected components. The first one serves to extract paragraphs, lines, and words of the real document in a top-down manner. Subsequently, the second technique takes word images to bottom up search for connected component (of black pixels) and character images.

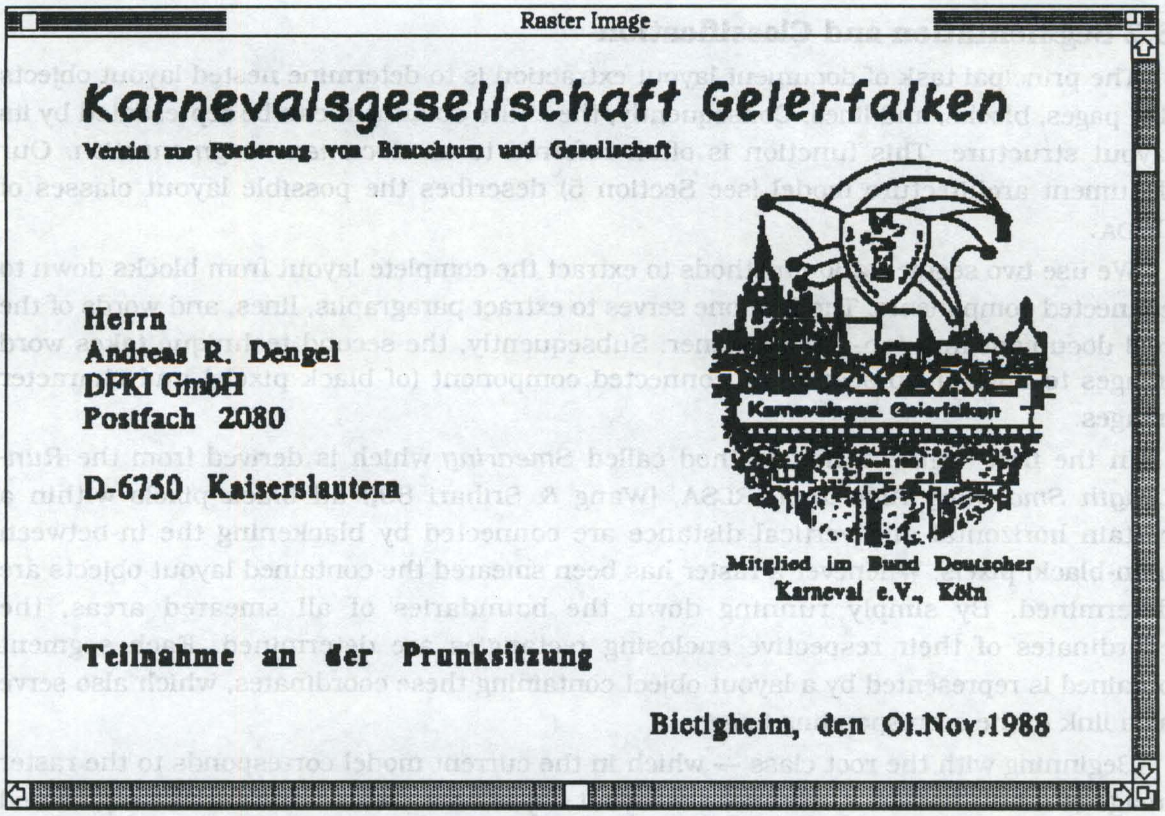
In the first segmentation method called *Smearing* which is derived from the *Run-Length Smoothing Algorithm* (RLSA, [Wang & Srihari 89]) all black pixels within a certain horizontal and vertical distance are connected by blackening the in-between (non-black) pixels. Whenever a raster has been smeared the contained layout objects are determined. By simply running down the boundaries of all smeared areas, the coordinates of their respective enclosing rectangles are determined. Each segment obtained is represented by a layout object containing these coordinates, which also serve as a link to the corresponding image.

Beginning with the root class — which in the current model corresponds to the raster image of a full "PAGE" — instances of layout objects are created and further segmented using the typesetting knowledge prepared by the according generic layout class. Thus, the method recursively proceeds with a big threshold analyzing the "PAGE" and extracting the subordinates of class "BLOCK". The line structure inside the "BLOCK"s is reconstructed by repeating the process with a smaller threshold on all blocks. In the same way words within lines are segmented.

The distinction between text and graphics objects is done by checking segment size and number of subordinates. This is unsatisfactory, but gives us tolerable results, since we do not want to analyze graphics but treat documents consisting mainly of text. Objects supposed as graphics by these criteria are denoted in the content-type-hypothesis by the value "NON-TEXT".

The Smearing method is an adequate technique for segmenting down to word level and easily to implement; disadvantages are its sensitivity against mixed font sizes and high impact of image resolution on the performance.

Figure 13 schemes input and output of Smearing: the input raster, and the final layout structure.



Raster Image

Karnevalsgesellschaft Gelerfalken

Verein zur Förderung von Brauchtum und Gesellschaft

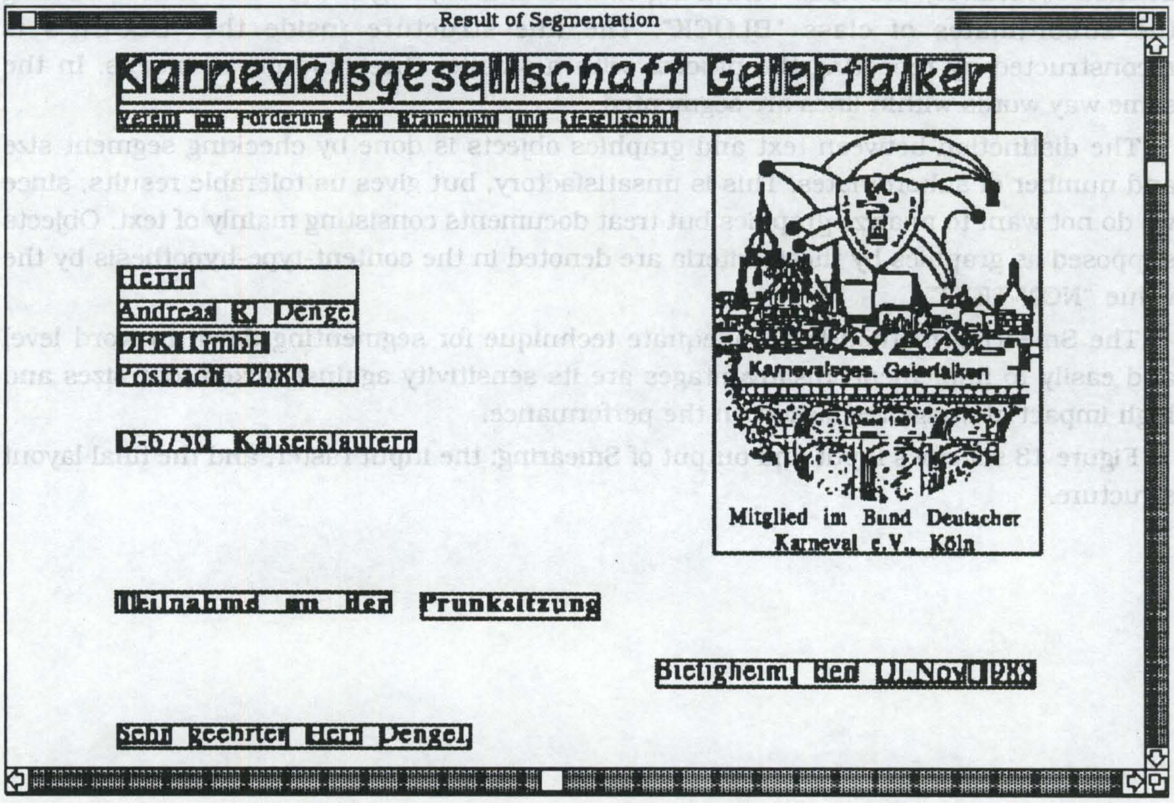
Herrn
Andreas R. Dengel
DFKI GmbH
Postfach 2080

D-6750 Kaiserslautern

Mitglied im Bund Deutscher
Karneval e.V., Köln

Teilnahme an der Prunksitzung

Bietigheim, den 01.Nov.1988



Result of Segmentation

Karnevalsgesellschaft Gelerfalken

Verein zur Förderung von Brauchtum und Gesellschaft

Herrn
Andreas R. Dengel
DFKI GmbH
Postfach 2080

D-6750 Kaiserslautern

Mitglied im Bund Deutscher
Karneval e.V., Köln

Teilnahme an der Prunksitzung

Bietigheim, den 01.Nov.1988

Sehr geehrter Herr Dengel

Figure 13: Segmentation of the example letter by Smearing.

After Smearing, the second method is applied in order to segment the word images into character segments. This needs a finer method and cannot be done by Smearing. Our approach uses chain code descriptions to encode the contours of "CONNECTED-COMPONENTS". By investigating the word raster for certain patterns, starting points for inner and outer contours are determined. Starting at these points, the entire word image is tracked for contours. The underlying technique ([Hönes et al 91]) considers the nesting of inner and outer contours and therefore, represents them by a recursive nesting of "CONNECTED-COMPONENTS".

The vertical arrangement of "CONNECTED COMPONENTS" is used in order to group them to "CHARACTER"s. This also requires knowledge about the allowed overlap between "CONNECTED-COMPONENTS".

The approach is rather time consuming, but allows to extract layout structures of arbitrary depth. A hierarchy of layout objects is displayed with a browser in Figure 14.

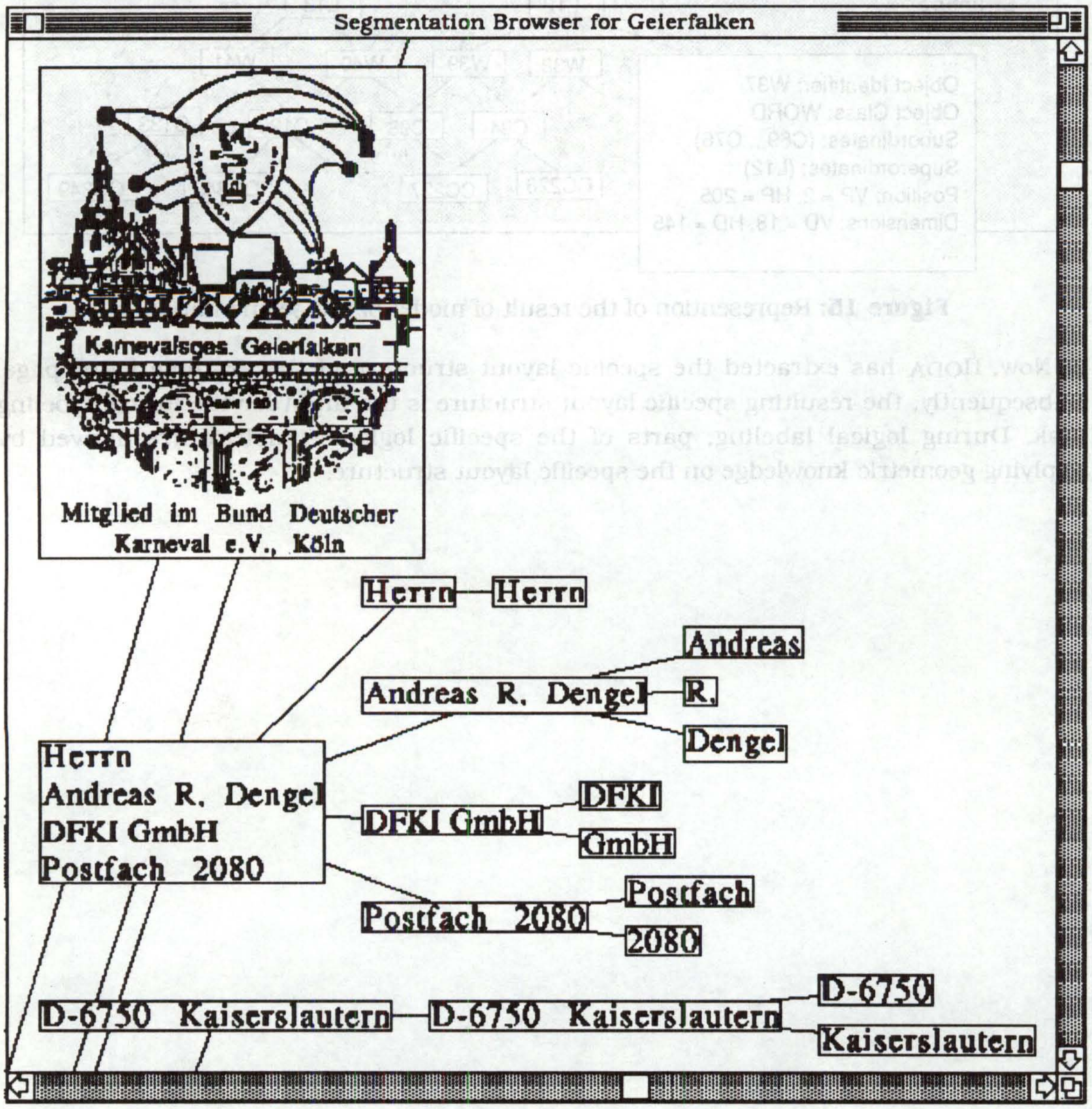
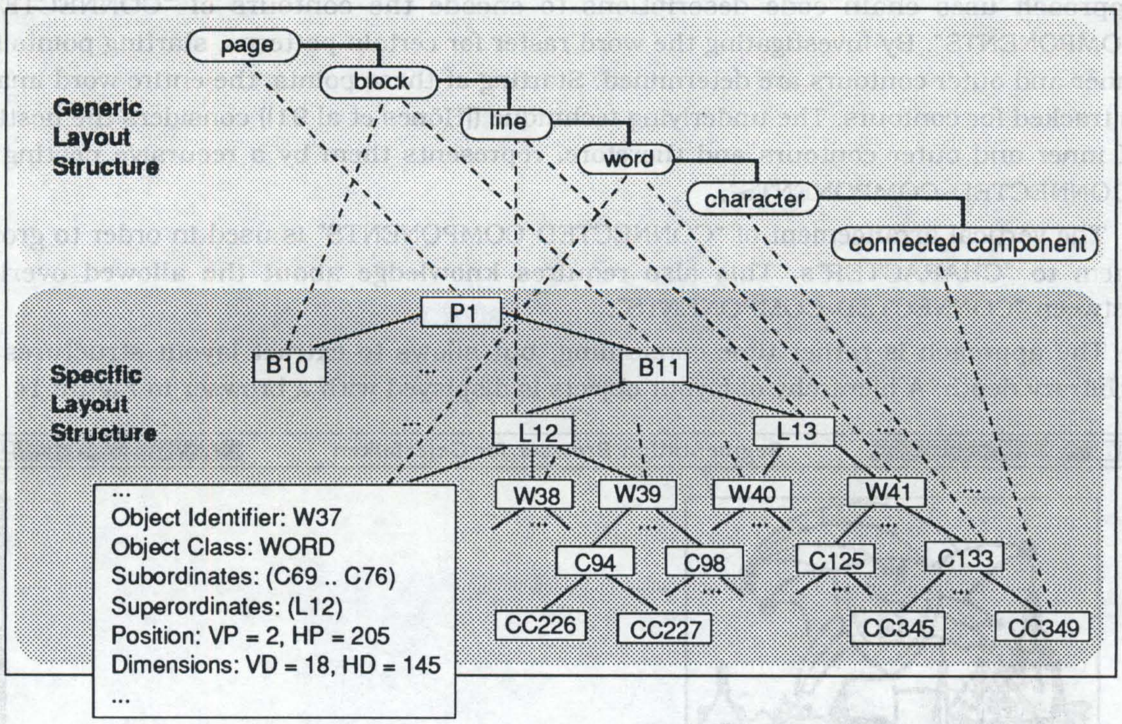


Figure 14: System browser's view of the segmentation result of the recipient.

A typical specific layout together with the underlying generic layout structure is schematically shown in Figure 15.



...
Object Identifier: W37
Object Class: WORD
Subordinates: (C69 .. C76)
Superordinates: (L12)
Position: VP = 2, HP = 205
Dimensions: VD = 18, HD = 145
...

Figure 15: Represention of the result of model-based segmentation.

Now, ΠODA has extracted the specific layout structure of the example letter page. Subsequently, the resulting specific layout structure is the input for the logical labeling task. During logical labeling, parts of the specific logical structure are derived by applying geometric knowledge on the specific layout structure.

7 LOGICAL LABELING

Logical labeling, also designated as *document understanding* ([Tang et al 91]), is one of the necessary and most important goals of document analysis. It refers to the task of mapping the specific layout structure of a document into a specific logical structure. This mapping is a necessary precondition to initiate a further processing of a specific document constituent. If, for example, the layout blocks refering to the addressee can be recognized, a further expectation-driven text recognition may be initiated matching the results with prestored sets of zip codes or destination places ([Srihari et al 87]).

Within our research activities, we focus on an investigation of human reading techniques and an employment of human knowledge sources for the automatic recognition and partial understanding of printed information. One of these knowledge sources is block arrangement (reading order)which is specifically for a certain document type.

Any type of document, such as medical records, reports, protocols, or business letters, may be characterized by a certain arrangement of logical objects adapted to human perception. There are documents with a prescribed structure and documents having a more complex and variable one. Usually, paper documents have a high degree of structure, which is seldom reflected by a human reader, but is directly used to filter relevant information out of the printed data. For a certain type of document, the respective structures are characteristic and can be represented specifically.

In Π_{ODA}, we use an approach described by [Dengel & Barth 88]. It is based on a special decision tree classifier for decribing logical object arrangements in business letters. This tree is processed by a *hypothesize & test* strategy providing logically labeled area items which are equal to specific logical objects.

7.1 Representation of Logical Object Arrangements

To model logical object arrangements in business letters, we have developed a formalism for document page representation that provides a global geometric view on a paper sheet. The structural elements of a document page, such as columns, paragraphs, and titles are generally laid out as rectangular blocks described by positions and dimensions. Additionally, orientation of text is along horizontal and vertical directions, determined by the rectangular shape of a typical sheet of paper. Thus, a single-sided business letter is considered as a rectangle having features width and height. To describe logical object arrangement, the page is divided into smaller rectangles by vertical and horizontal cuts. Model cuts are placed in such a way that they do not intersect with printed text or non-text blocks. The subrectangles can be recursively divided in the same way, until the arrangement of logical objects on the page is described in sufficient detail. To define a specific arrangement, different rectangles are assigned a label which describes the corresponding logical meaning. The various arrangements of logical objects within business letters are collected in a so-called *geometric tree* (cf. Figure 16).

A *geometric tree* is a specialization hierarchy. It allows a representation of logical object arrangements on different specification levels. The root of the tree represents the most general arrangement. Every single-sided document belongs to this class. The internal representation is organized in such a way that logical object arrangements of parental nodes are inherited by children. Consequently, most document pages can be partitioned into nested rectangular areas by order, position, and orientation of cuts as well as by assignment of logical labels. For our experiments, we use a tree having about

40 terminal nodes. This part of model is associated with the root of the generic logical structure, i. e. BUSINESS LETTER, as *geometric knowledge*.

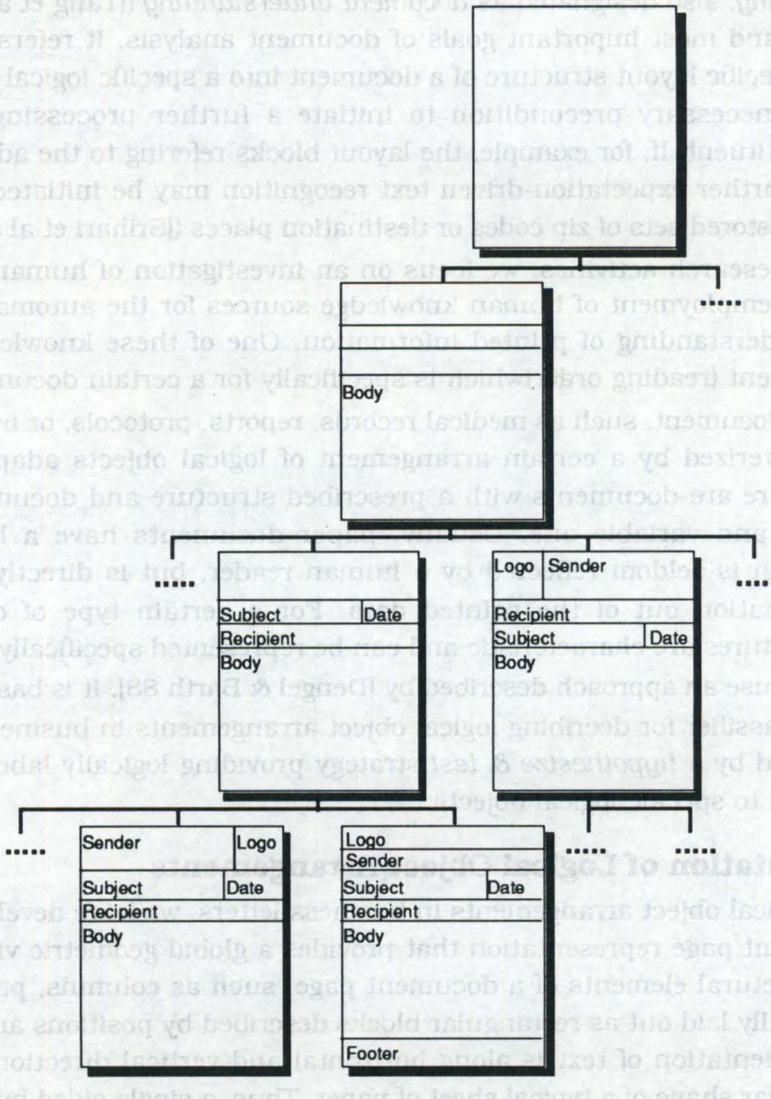


Figure 16: Principle representation of a geometric tree.

7.2 Geometric Description of Logical Objects

In addition to the geometric tree, individual logical objects can be described independent of the text they contain just by their shape. Therefore, each generic logical object also includes a *geometric knowledge* slot capturing statistical results obtained by evaluating about 190 business letters under geometrical aspects. In particular, all logical objects were examined with respect to their intrinsic geometric characteristics. For example, the recipient in a business letter be geometrically characterized as follows:

◊ the position of the recipient is in the first upper third of the page;

◊ the left margin of the recipient is within the left quarter of the page;

◊ the horizontal extension is not longer than a third of the page width;

◊ the recipient is not written in an extremely large or small font;

◊ the recipient consists of four to six text lines, which are left justified.

Note again that these rules do not use content information, such as keywords. Up to this processing step, we do not employ any procedure for the recognition or analysis of text. To avoid all problems in connection with unrestricted text recognition and furthermore to provide a basis for expectation-driven text recognition, we concentrate on geometric features of logical objects. To this end, we describe the following logical objects:

- sender,
- recipient,
- letter-body,
- company-logo,
- date,
- subject,
- relational data,
- sender-short-form,
- signature,
- footnote,
- company-specific-printings.

Most of the probabilities resulting from the examination are represented as measures of belief *(MB)* in subsets of rules. While they are applied for hypothesis confirmation, others are applied for their refutation as measures of disbelief *($MD = 1 - MB$)*. For example, geometric rules for the recipient in Π_{ODA} are as the following:

Feature	Rule	MB	MD
vertical origin (*VO*)	$0 \leq VO \leq 0.25$	0,89	
	$0.25 < VO \leq 0.33$	0,10	
	$0.33 < VO$		0,99
	⋮		
left justified	true	0,99	
	false		0,99
number of lines (*NL*)	$NL < 4$		0,95
	$4 \leq NL \leq 6$	0,92	
	$NL = 7$	0,02	
	$NL > 7$		0,99

Note, that values for a measure-of-(dis-)belief range between 0 and 1. The values in the rules are relative to the width and the height of a page. Actually, in Π_{ODA} 11 logical objects are described each by 11 different geometric features. This local assignment of the subsets of rules to the generic logical objects allows a straightforward testing of intrinsic features of given layout blocks as well as easy addition of new rules. When new rules are added, MBs as well as MDs for existing rules need not to be altered because every subset is independent from each other.

7.3 Logical Object Identification

Logical labeling of a given document amounts to finding a path from the root of the geometric tree to one of its leaves, thereby stepwise matching the specific layout structure against the alternative arrangements (see Figure 17). If an arrangement in a tree node fits with the specific layout structure, i.e. does not intersect with specific layout objects — small deviations are allowed, but reduce the credibility [Dengel & Barth 89] — the respective label is assumed a hypothesis. For verification, all specific layout objects related to the logical object are compared to the geometric rules in the appropriate knowledge slot of its class. The MBs and MDs obtained are combined by Dempster-Shafer´s rules of combination (for details see [Dengel 92]). In the case, hypothesis verification succeeds, a specific logical object is generated as an instance of the corresponding generic logical class.

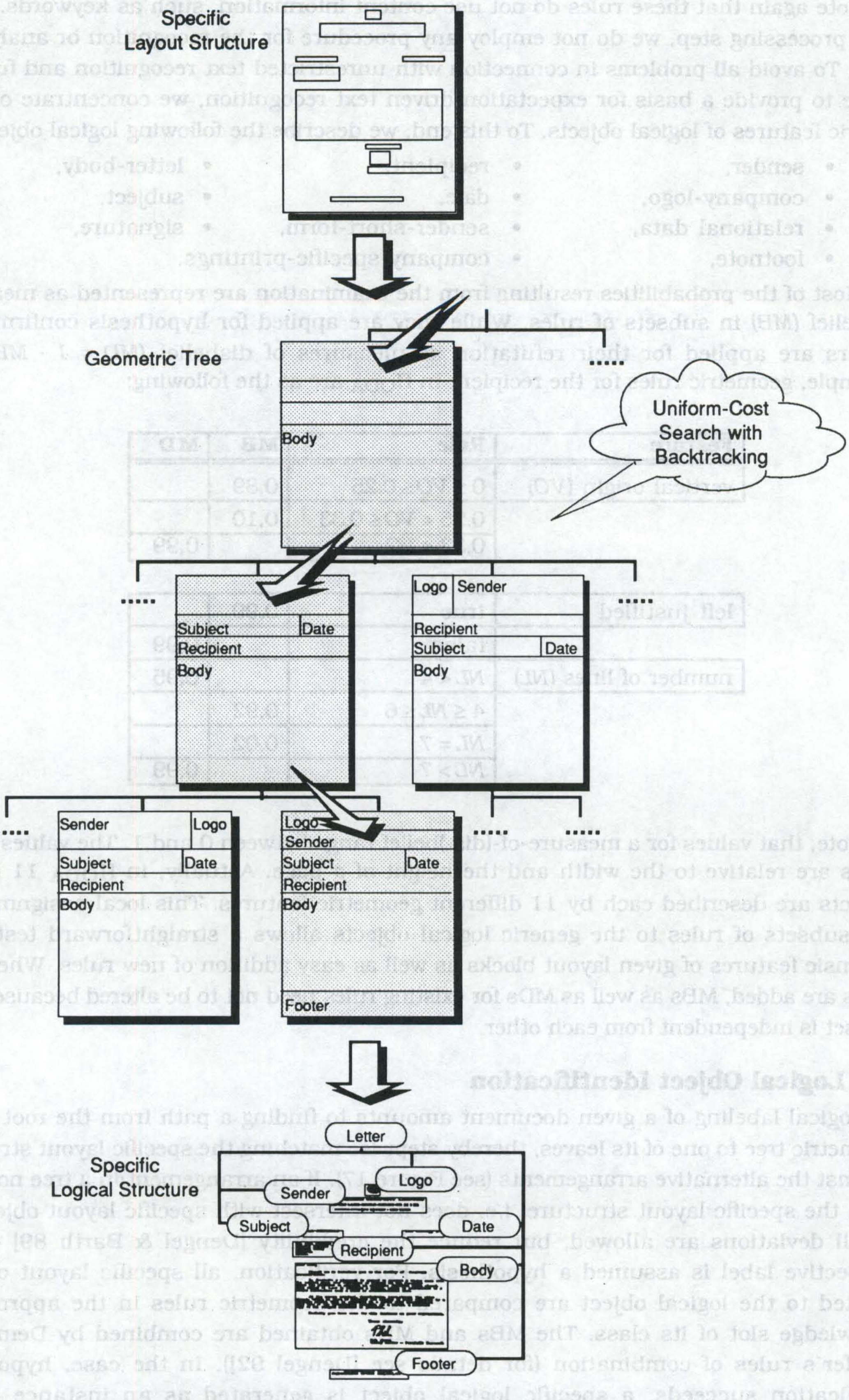


Figure 17: Logical labeling applying the geometric tree.

By combining measures of belief while stepping through the tree, it is possible to quantify the degree of similarity between the current specific document layout structure and the nodes in the *geometric tree*. For handling different intermediate results for document arrangement class matching, the application of the decision tree allows easy reduction of the search space.

All intermediate results are collected in an agenda [Dengel & Barth, 88]. In every stage of the analysis, it provides the best intermediate solution for further examination. In other words, to perform one step on the path in the geometric tree, the intermediate solution with the highest measure of belief is chosen to refine an arrangement. Thus we perform a *best-first search*, which represents a variant of the *uniform-cost search* [Barr & Feigenbaum 81]. However, if logical labeling fails, ΠODA initiates a resegmentation of the document image.

The specific logical objects generated during labeling represent instances of the respective classes in the generic logical structure. Their relations to specific layout objects provide an ODA conforming representation of a given document taking the two structural but complementary views to information into account. The resulting structures are sketched in Figure 18. They are the basis for a further expectation-driven investigation of the text images captured within specific logical objects, e.g. for address recognition in the logical object *recipient*.

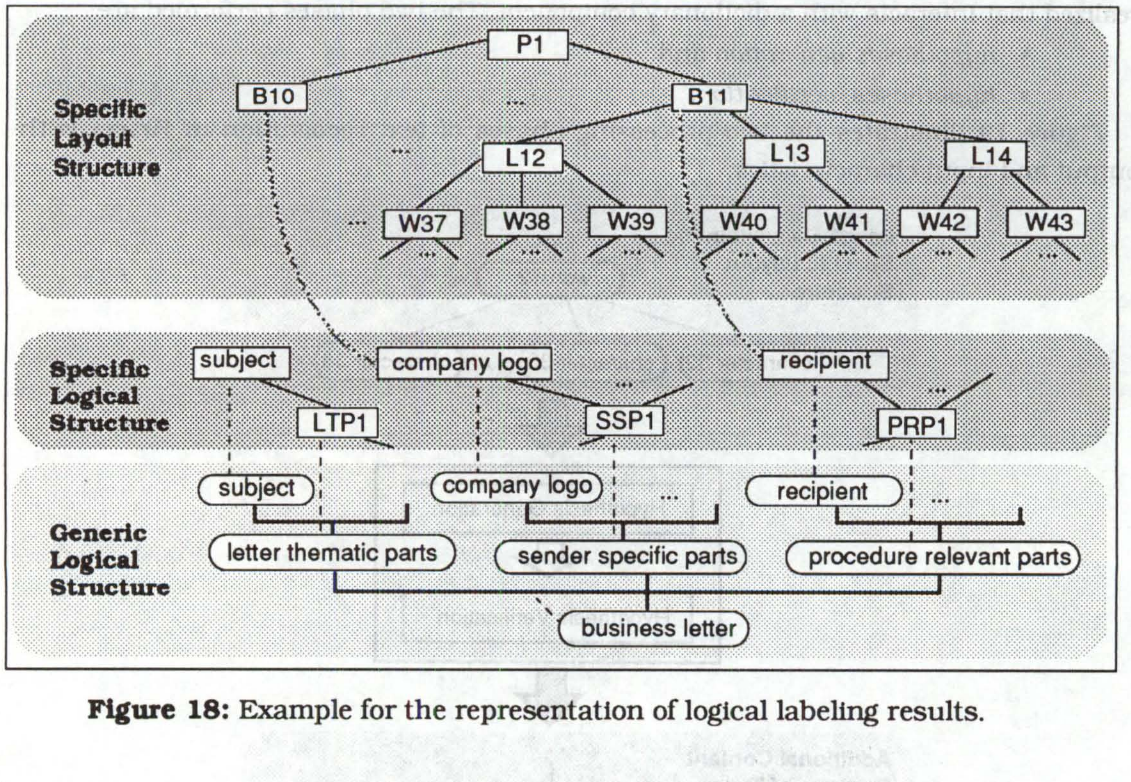


Figure 18: Example for the representation of logical labeling results.

8 TEXT RECOGNITION

For identifying the message of a document, it is not sufficient to solely label important document parts. To serve a broader basis for a global document interpretation and understanding, words captured in the labeled document parts have to be recognized. We prefer words instead of single, isolated characters because only words can be related to meanings. These sometimes ambiguous meanings together with their interrelations within a phrase or a sentence enable the determination of the contents of logical objects and consequently, the intention of the document's sender.

For that purpose, we have implemented a text recognition specialist using word segments as well as their subdivision into characters as input and consequently generates candidates of text strings as output. The features used for recognition are robust and limited in amount. The global aim of this purpose was to generate ASCII strings with a few but simple features. These features should not necessarily enable a definite determination of character sequences. Instead, it should be investigated how and up to which degree word context can be used for assisting recognition process as well as for verifying and completing word fragments.

Regarding the above mentioned restrictions, a text recognition specialist has been realized that interacts with a dictionary component. The two phases performed are

- *hypotheses generation* and
- *hypotheses verification.*

Figure 19 illustrates the two processing phases of text recognition in Π_{ODA} with output and input characteristics.

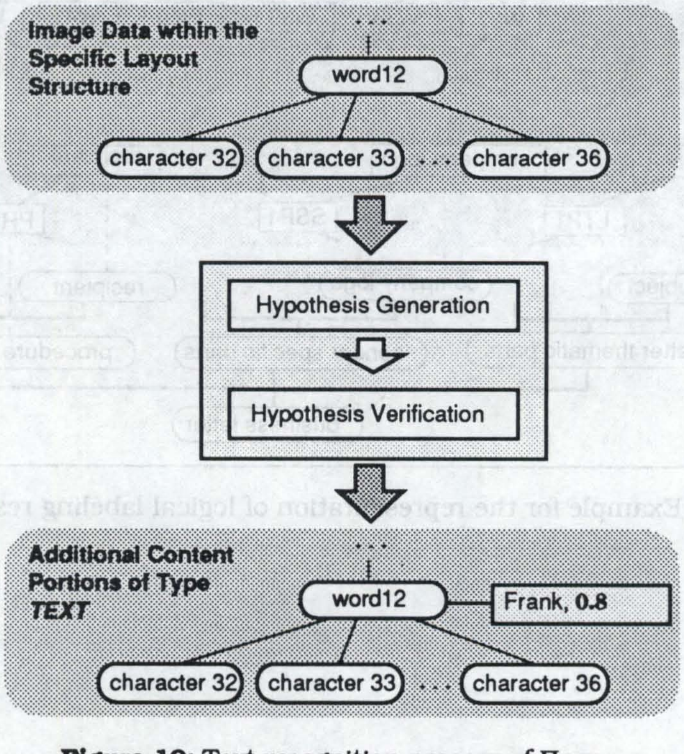


Figure 19: Text recognition process of Π_{ODA}.

The first step tries to generate weighted character alternatives where only image information is considered, while the second step verifies the ordered set of character alternatives as a set of legal, alternative strings. Here, no image information is used. The second phase is based on dictionaries attached to logical objects. The construction and behavior of both processing steps are described in the next sections.

8.1 Character Hypotheses Generation

As mentioned above, layout extraction results in determining image data for word objects. While investigating word images for the purpose of text recognition, two main problems arise:

- selection of relevant image features
- determination of a suitable classification schema

For selection of features, we have to decide if we prefer a set of complex and specific features that is difficult to extract but enables an easy classification or rather few, but general simple features that are easy to identify, but complicate the classification process.

It is not a goal of our research activities to concentrate on improving existing techniques in character feature recognition and combination, but rather to guide text recognition by knowledge from beyond the image data level. Therefore, it seems to be a good strategy to perform a classification based on simple features and to verify the classification results by additional knowledge. Following this strategy, we evaluated the robustness, the expense of extraction, and the significance of the features.

As result, following character features are used:

- circumscribing rectangle
- typographical context of a line segment
- projections to x and y axis

8.1.1 Feature descriptions

Before we start to explain the classification procedure, the mentioned features are briefly described.

Circumscribing Rectangle

The circumscribing rectangle [Bartneck 87] defines two characteristics: the position as well as the dimensions of a character on the document image. Figure 20 shows a binary image of the character "Ä" with its position coordinates and its breadth and height.

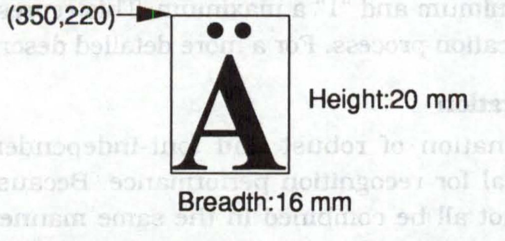


Figure 20: Character "Ä" and its box.

Typographical Context

Using these features and additional information from superior segments, the typographical context of line segments can be calculated. Consequently, each character can be categorized according to position within a line segment [Röhl 89]. As illustrated in Figure 21, there are characters with *ascenders* (e.g. "l", "i"), characters with *descenders* (e.g. "p", "y"), some characters with ascenders and descenders (e.g. "J"), and at last, characters without both (e.g. "a", "m", "e"). For identifying these two areas, the line segment is divided using the four imaginary lines: *upper line*, *half line*, *base line*, and *lower line* (see also Figure 21).

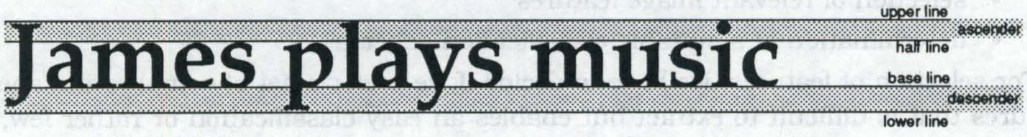


Figure 21: Text line with ascender and descender information.

Projections

As a third category of features we use projections to the x and y axis [Dewes 92]. Both features provide information about the pixel distribution of a character image. A projection to the x axis (y axis) can be established adding all pixel columns (rows) of the binary image. The resulting histogram is designated as x projection (y projection). Figure 22 shows the x and y projections of a character "a".

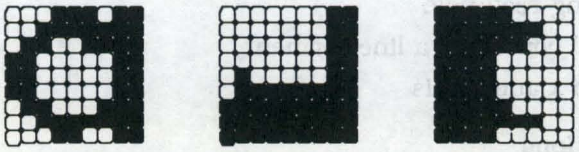


Figure 22: Image of character "a" with x and y projections.

Unfortunately, these projection measurements strongly depend on the size and the font. But we aim to obtain character attributes that are more or less independent of such variations. Therefore, the histogram is normalized. Each projection is divided into three parts where for every part it is determined if there is a maximum (large number of pixels) or a minimum (low number of pixels). In other words, we perform a transformation from a histogram with any size to a input vector $v_I = (x_{I1}, x_{I2}, x_{I3})$ where $x_{I1}, x_{I2}, x_{I3} \in \{0,1\}$, where "0" denotes a minimum and "1" a maximum. This representation produced builds the input of the classification process. For a more detailed description, see [Dewes 92].

8.1.2 Hypotheses Generation

Beside the determination of robust and font-independent image features, their combination is essential for recognition performance. Because features have different credibilities, they cannot all be combined in the same manner. For example, ascender and descender have a high credibility, because the four lines dividing a text segment into three areas can be exactly determined. But the projection measurements are more sensitive to noise and font styles and therefore, have a lower credibility.

Considering these circumstances, we have developed a two-step classification schema where the first step performs a reduction of the candidate set while the second assesses the remaining character hypotheses.

In the first step, each character image is characterized by ascender and descender information. As seen in Figure 23, four possible classes can be established: characters without ascender and descender, characters either with ascenders or descenders, and characters having both. The distribution of characters to the single classes are 19.7%, 69.7%, 6.6%, and 4.0%. The order in which the single classes are established is not of interest, because the two divisions are independent and therefore, both possibilities yield the same result. In our approach, we first divide characters into classes with and without ascenders. As result of step one, we obtain a reduced set of possible characters: in the best case (characters with ascender and descender), the reduction is 96%, in the worst case (characters only with ascender) 30.3%.

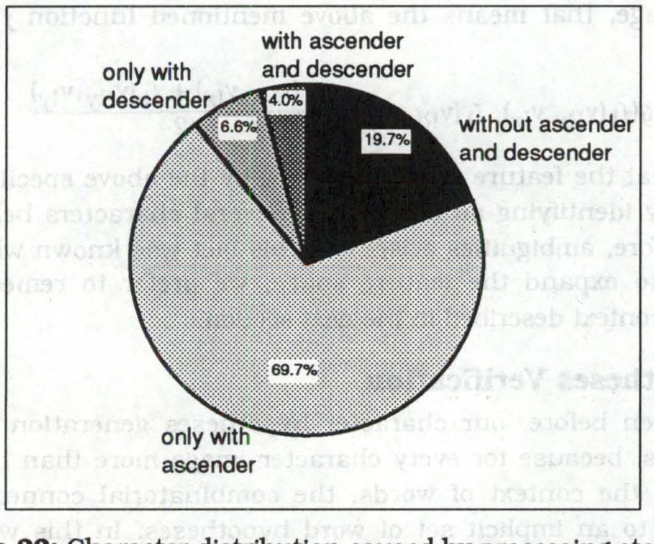


Figure 23: Character distribution caused by processing step one.

The remaining character candidates are assessed in a second step considering the vector of projections. The vectors v_x and v_y denoting the modified x and y projection of a character are exploited to determine two credibilities for candidate to character attachments.

Let v_{Rx} and v_{Ry} be the x and y projection of the reference class *R* of a character and v_{Ix} and v_{Iy} the projections of an input character image *I* with $v_{Rx} = (x_{R1}, x_{R2}, x_{R3})$, $v_{Ry} = (y_{R1}, y_{R2}, y_{R3})$, $v_{Ix} = (x_{I1}, x_{I2}, x_{I3})$, and $v_{Ry} = (y_{I1}, y_{I2}, y_{I3})$, where $x_{Ri}, y_{Ri}, x_{Ii}, y_{Ii} \in \{0,1\}$, $i = 1,2,3$. The function *f* with

$$f(v_R, v_I) = \mathcal{G}(f_1(v_{Rx}, v_{Ix}), f_1(v_{Ry}, v_{Iy})) \qquad (8.1.2.a)$$

yields the credibility measurement that *I* belongs to the class *R*. Function *f* consists of two separate successive applications of the function f_1 of the x and y axis. f_1 yields a credibility measurement from 0 to 1 and has following construction:

$$f_1(v_{Rx}, v_{Ix}) = \frac{1}{\left| \sum_{i=0}^{3} |x_{Ri} - x_{Ii}|^2 * \alpha + \beta \right|} \qquad (8.1.2.b)$$

with α and β as trimming factors. They adjust which differences between a reference class vector and an input vector have which effects for the credibility measurement.

A simple example illustrates the behavior of function f_1. Assume $\alpha = 0.3$ and $\beta = 1.1$ and the vector of the x projection of the reference class "U" be (1, 0, 1). Now, if the input vector would be

$$v_{Ix} = (1, 0, 1), \quad f_1 = \frac{1}{|0^2 * 0.3 + 1.1|} = 0.9,$$

$$\text{if } v_{Ix} = (1, 0, 0), \quad f_1 = \frac{1}{|1^2 * 0.3 + 1.1|} = 0.7, \text{ or}$$

$$\text{if } v_{Ix} = (0, 0, 0), \quad f_1 = \frac{1}{|2^2 * 0.3 + 1.1|} = 0.4.$$

In Π_{ODA} the two measurements $f_1(v_{Rx}, v_{Ix})$ and $f_1(v_{Ry}, v_{Iy})$ are combined using the arithmetical average, that means the above mentioned function f has the following construction:

$$\mathcal{G}(f_1(v_{Rx}, v_{Ix}), f_1(v_{Ry}, v_{Iy})) = \frac{f_1(v_{Rx}, v_{Ix}) + f_1(v_{Ry}, v_{Iy})}{2} \qquad (8.1.2.c)$$

It is obvious that the feature space stretched by the above specified features is too small for uniquely identifying all characters. Several characters belongs to the same cluster and therefore, ambiguities arise. But this fact was known while designing this system. Instead to expand the feature space, we prefer to remedy this weakness considering word context described in the next section.

8.2 Word Hypotheses Verification

As we have seen before, our character hypotheses generation process produces ambiguous results, because for every character image more than one text candidate possibly exist. In the context of words, the combinatorial connection of character hypotheses leads to an implicit set of word hypotheses. In this way, most of these ambiguities — caused of the restricted feature set — can be removed using additional knowledge in form of dictionaries.

For hypothesis verification, two different and alternative techniques have been tested:

- stepwise reduction of candidates
- global candidate assessment

Both techniques are described in the following two sections.

8.2.1 Reduction of the Candidate Set

The first technique assumes that all words of a dictionary are possible candidates for a sequence of character hypotheses (see Section 8.1). Consequently, the verification process attempts to reduce this candidate set considering the single character hypotheses. For that purpose, dictionary words not corresponding with single character hypotheses at a certain position are stepwise neglected. Starting with the first set of character hypotheses, all words in the dictionary are eliminated that do not begin with the corresponding characters. The same procedure is applied to the second and all following word positions. Thus, each character at position i reduces the candidate set produced by the first (i-1) positions. If the last character position is processed, the reduction process ends. For every character, a dictionary look-up is performed that controls if words with such a prefix exist (*prefix check*). If no word is available, the amount of possible words is not decreased.

Finally, we obtain following results:

- a set of dictionary words capturing only words that correspond with the given character hypotheses or
- dictionary words with the same prefix, if the real word is not in the dictionary.

The technique described is efficient and usually yields all corresponding dictionary entries that means at least the correct word. But, if all hypotheses at a position are wrong, especially at the beginning, some problems arise. For example, assume hypotheses generation has produced following output for the input string "Ocean":

$$\left(\left\{ \begin{matrix} \text{"Q"} \\ \text{"D"} \\ \text{"C"} \end{matrix} \right\} \quad \{ \text{"c"} \} \quad \left\{ \begin{matrix} \text{"e"} \\ \text{"c"} \end{matrix} \right\} \quad \{ \text{"a"} \} \quad \left\{ \begin{matrix} \text{"n"} \\ \text{"m"} \end{matrix} \right\} \right)$$

Each parenthesis includes alternatives for one word position.

Since we stepwise reduce the word candidates, in the first step, all words not beginning with "Q", "D", or "C" are eliminated, because there are words beginning with one of these 3 characters. Thus, "Ocean" can never be verified, since it was reduced from the candidate set. If the defect is at the end like in "Oce?n", a reduction caused by the position 4 is not performed, because no word exists starting with "Oce". "Ocean" is still in the candidate set and therefore, can be a possible result. Similar problems occur regarding merged characters.

8.2.2 Global Candidate Assessment

Our second technique, the *global candidate assessment*, avoids this problem. It also checks all combinations of character hypotheses with the dictionary component, but no candidate reduction is performed. Instead, all elements of the dictionary are assessed considering the similarity of word hypothesis and dictionary word. The function

$$f_a = \frac{c}{\max(l,n)} \qquad (8.2.2.a)$$

defines the assessment measurement of a dictionary entry where c denotes the sum of credibility measurements of the single character hypotheses which correspond with the dictionary word, l the number of characters of the word hypotheses, and n the number of characters of the dictionary entry.

The selection of ingenious candidates depends only on this measurement. All dictionary words having a higher assessment measurement than a given threshold T are added to the candidate set. If no such words exist, those one is selected having the highest measurement. Thus, defects at any position in the word hypothesis do not directly disqualify the correct dictionary word. Only its assessment measurement is reduced.

But also the second technique causes two problems: the adequate determination of the threshold T and long runtimes. The parameter T is sensible, because large values strongly reduce and small values glut the word set. The other problem is caused by the global consideration of the entire dictionary. Because till to the end, all dictionary words are considered as possible word candidates, a multitude of dictionary accesses is performed that are time expensive although having a fast access mechanism.

As result, we obtain — like in technique one — dictionary entries, ordered by the assessment measurement that corresponds with the hypothesis. If a defect at any position occurs, the measurement is only decremented. Thus, in the above mentioned example "Ocean" would be an element of the resulting word set.

In practice, this method can be only applied using very small dictionaries. In our tests, the recognition of 300 words in combination with a dictionary of 8.000 entries, the system needs about 6 hours. Therefore, we prefer technique one and tolerate that some words cannot be identified.

One main aspect of both techniques is the identification of a hypothesis as a legal word that means as a word of our dictionary. So far, we have only referenced a dictionary component, but not their construction and abilities. Therefore, the next section describes the lexical knowledge in more detail and explains its structure and access techniques.

8.3 Dictionary Organization and Access Mechanisms

Applying contextual knowledge for the verification of results of text recognition can be classified into three basic approaches: *dictionary look-up methods*, *probabilistic methods* (Markov models) and *combined methods* ([Hanson 76], [Sinha 88], [Elliman 90]). While Markov methods use a-priori (statistical) knowledge about transition probabilities of characters by n-grams (usually bigrams or trigrams), dictionary look-up techniques verify the actual input string against a legal set of words being collected in a dictionary.

We use a dictionary-based approach for two reasons: First, while Markov methods are very fast and efficient, they are extremely sensitive in cases of misspelled or incomplete input words. Second, storing dictionary entries and corresponding lexical information explicitly will enable a subsequent partial understanding of text, e.g., involving keyword analysis and parsing techniques.

A well-known and proved technique for dictionary organization is that of a *trie* memory (re*trie*val; see also [Fredkin 60], [Knuth 73], [Aho et al 83], [Sinha 87], [Wells 90]). Tries are attractive because of their simple and compact storage allocation. A trie considers each word entry as an ordered sequence of characters being represented as nodes in a tree. Common parts in the beginning of words, or prefixes, are stored exactly once. Word access is performed character by character beginning at the root of the trie structure. Note if a search is not successful, at least a best match has been found. Advantageously, tries are well-suited for efficiently storing words which have common prefixes and guarantee a linear access time (*O(n)*, where n is the length of the input word). For that purpose, a prefix check which is required of our word verification component can be implemented easily. On the other hand, a disadvantage is that a trie structure consumes a lot of memory in storing additional link and housekeeping information.

All these properties of tries are only true when the input string completely equals a valid word in the dictionary. In the case of an incomplete or incorrect string, a lot of time can be wasted, for example by a depth-first-search. To minimize search time, we propose a so-called selective-access-matrix which will be described later. For a reduction of size—in order to hold the trie in main memory—we use three different representations of a trie node each of which is optimal under some conditions, including a bit-array representation of a trie node, a character-pointer-array or string compaction. For details of the representation of trie nodes see [Wells 90] and [Dengel et al 92].

So far, we have implemented two trie access mechanisms. First, and being the simplest case, an input string (word or prefix) is completely known and solely needs to be checked against the dictionary. This is a straightforward and basic operation on a trie already explained. Second, when the input is only partially known, a more flexible search method must be applied. Here, following situations may result from recognition:

(1) Alternatives of characters (designated by brackets "[,]")

(2) Rejection of single characters (designated by symbol "?")

(3) Rejection of substrings (wildcards) at beginning or end of word

Situations (1) and (2) are handled by depth-first search. For dealing with situation (3), we developed a so-called *selective-access-matrix* (SAM). The SAM is a (c∗n)-matrix, where *c* is the cardinality of the underlying alphabet and *n* a positive integer indicating the actual position in a word. Each element of the matrix is a pointer array whose entries point to trie nodes at level n containing character c (cf. [Dengel et al 92]). Figure 24 illustrates both the organization of the dictionary as trie and the corresponding SAM.

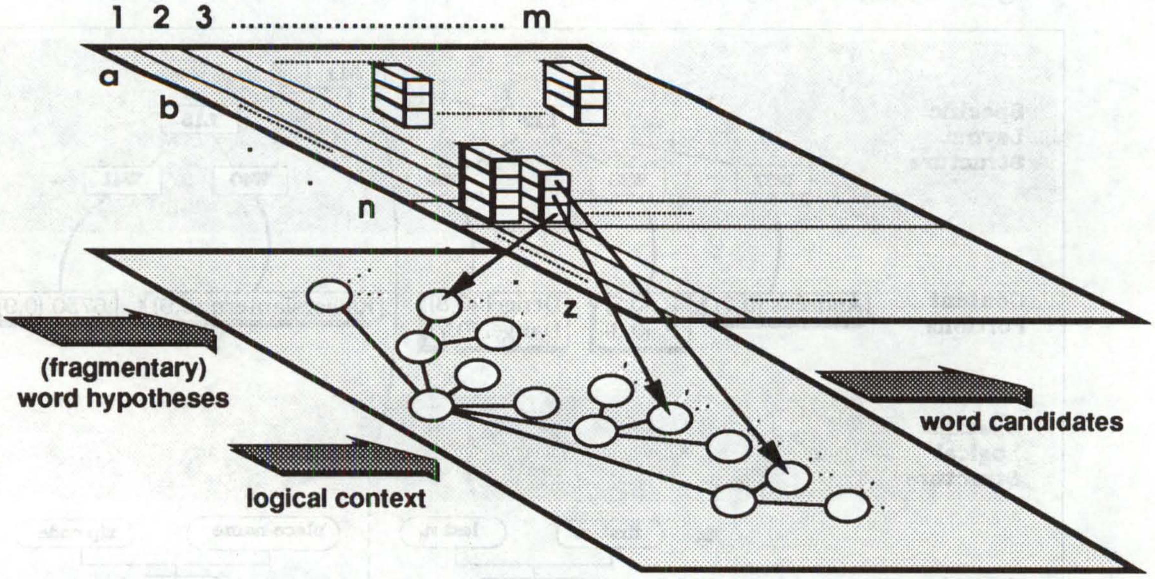


Figure 24: A trie data structure with a selective-access-matrix (simplified).

A hybrid trie representation including the access matrix has been fully implemented and tested. All tests were performed on a basic dictionary of about 8000 most frequent German words. Results show that the additional storage needed for the SAM is smaller than the memory saved by a hybrid organization using different types of trie nodes. Moreover, retrieval of dictionary entries is fast, even if the corresponding input string is rather disturbed through several wildcards (run time measurements are shown in [Dengel et al 92]).

Although new entries are permanently added to the dictionary and hence storage demand will increase successively, our trie remains a very promising component to support the word verification phase. But, in parallel, we examine hash table methods also allowing the access of fragmentary recognized words (see [Schürmann 78], [Kohonen 78]). All in all, we develop a structured lexicon architecture involving several (logical) partitions of lexical data by usage of sophisticated data structures and access techniques. A structured lexicon will support and improve text recognition as well as higher-level interpretations of documents (document understanding).

8.4 Results

In ΠODA only word segments related to the logical object *recipient* are considered as input for recognition process. These layout objects and their subordinated character segments serve as a basis for the character hypothesis generation. Consequently in a second step, the sequences of hypotheses produced are verified as legal words performing dictionary look-ups. In opposite to conventional OCR systems, the dictionary includes only words usually occurring in the recipient block. Therefore, it is attached to the logical object recipient. If for all word segments the corresponding text is ascertained, content portions of type *text* are generated. For that purpose, the class *content-portion* is instantiated and immediately attached to *word* objects of the specific layout structure. As content attribute, the list of alternative words including the assessment measurements are added.

Figure 25 fragmentarily shows our document structure after text recognition.

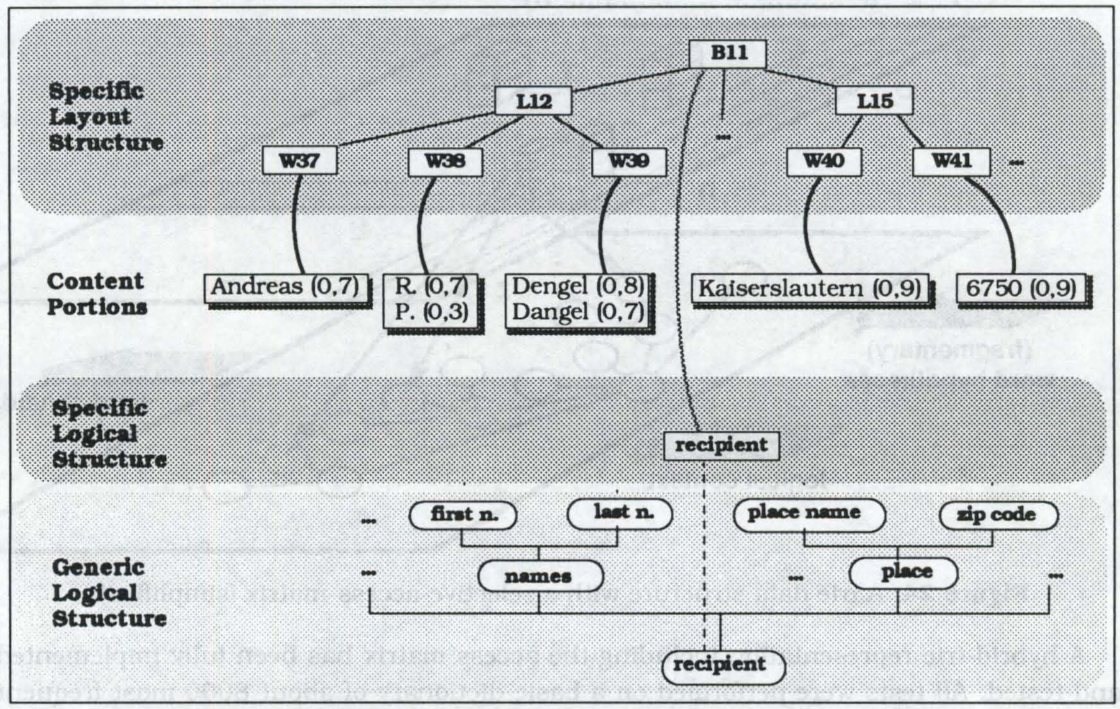


Figure 25: Part of our document structure including text recognition results.

As mentioned, our recognition process is restricted to word objects related to the recipient. But it can be easily expanded to other document parts. The only necessary work we have to do is the establishment of dictionaries containing typically words of that logical objects and their attachment to that object. Because the dictionary component strongly influences the recognition performance, one problem exists: if the word set of a logical object is very large (such as in the text body), recognition ambiguities greatly increase.

In this way, the overall result of our text recognition phase is a specific layout structure, an incomplete specific logical structure, and content portions that are linked to layout objects of type *word*. The content portions include alternative word hypotheses inclusive particular assessment measures.

For completing our document analysis, a text analysis procedure is performed which generates links between content portions and logical objects and is described in the next chapter.

9 Text Analysis of Addresses

Hitherto, text recognition provides a set of candidates for each word. They are associated with respective content portions. As shown above, these content portions are linked to basic layout objects but not yet to any logical objects in the specific structure. For some of the logical objects, such as *recipient* or *date*, their syntactic structure as well as their potentially captured text is relatively fix. Thus, the specification of syntactic knowledge and of sets of typical words, such as keywords, identifiers, and names, allow their structural refinement into specific logical objects, such as *name* and *place*, etc., but also the disambiguation given by alternative readings in the content portions. As a consequence, it is possible to relate content portions of the word level to specific logical objects, such as the recipient´s *first name* or *last name*. Therefore, it is possible to ask for the verified recipient and to accordingly transmit the document via electronic means.

In this chapter, we show how this task is performed by exemplarily analyzing the logical object recipient. With some modifications, the same approach is also working for other well-structured parts like date, salutation or senders address.

The syntactic conventions of a recipients address is relatively strong, conditioned by historical reasons and existing postal standards. By such means, the sequential order of the recipients name, his home street and his home town can be taken as mandatory. Additionally, we can assume a set of well-known designating words, e.g. the customers of a company, zip codes, towns, and street names. Furthermore, there are some "keywords" occurring in the title, and some degrees (academic titles) as part of a recipients name.

An adequate way to represent syntactic conventions of an address is using production rules. Thus, we can relate the logical object *recipient* to the left hand side of a production rule *Recipient* which is composed of the *Addressee* and the *Direction*, i.e. the specification in which town and street a letter has to be delivered. To say it in terms of language theory, a recipient address can be seen as a syntactic category, especially the start symbol in context free grammars or – just another name – the *top category*. It consists of various constituents, in our example *addressee* and *direction*. Some of these but not all of the constituents, are identical to objects in the generic logical structure.

For utilizing familiar words that could appear as instances for the several constituents of a recipient, respective collections of words are associated to corresponding logical objects in the generic structure. For example, all last names of the company´s employees are attached to the *lexical knowledge* slot of the generic logical object *last name*. A next step, attributes are incorporated that allow for checking the consistency among derived addresses, e.g. the agreement between a town's name and the corresponding zip code.

This approach can be mapped to the paradigm of attribute grammars. The historical roots of such formalisms that can be found in Chomsky's theories led to the establishment of a multitude of grammar formalisms especially in the last decade. Introducing such formalisms (e.g.,[Kaplan & Bresnan 82], [Pereira & Warren 80]) would doubtless be beyond the scope of this report. Therefore, we undertake a more pragmatic proceeding in explaining the current state of the implementation by means of an example.

9.1 Encoding the structure of addresses

For illustration, we now take a look at a small sample grammar, noted in an EBNF-like syntax, an extended version of the Backus-Naur form (see Figure 26). Alternatives are separated by vertical bars ("|"), optionals are enclosed in brackets ("[", "]"), optional-repeatable elements in parenthesis ("{" and "}"); terminal symbols are double quoted.

```
Recipient          ::= Addressee Direction .
Addressee          ::= NatAddressee [ Company ] |
                       Company [ [ "z.Hd." ] NatAddressee ] .
NatAddressee       ::= Title PersonName .
PersonName         ::= { Degree } { FirstName } LastName .
Title              ::= "Frau" | "Herrn" | "Familie" | ...

Direction          ::= ( Street | "Postfach" BoxNumber ) Place .
Street             ::= StreetName NumberInStreet .
NumberInStreet     ::= Number .
Place              ::= [ StateSign "-" ] ZipCode PlaceName [ PostalDistrict ] .
ZipCode            ::= Number .
...
```

Figure 26: A small, incomplete grammar for a recipient address.

This notation has to be read as follows. A *recipient* is constituted as a mandatory sequence of *Addressee* and *Direction*. For formulating the *Addressee*, there are two possibilities divided by a vertical bar. The first starts with *NatAddressee*, optionally followed by the specification *Company*, the second starts with the *Company*, optionally followed by the *NatAddressee* which may optionally be preceeded by the string "z.Hd." (means "c/o"). Analogous, the other rules have to be read.

That way, only combinatorial restrictions are formulated. Up to now, no semantic check, e.g. concerning the agreement between zip code and town's name, is incorporated because the grammar gives no possibility to do so.

Guided by the expectation yielded from logical labeling, the *recipient* is syntactically analyzed employing the grammar associated with the corresponding generic logical object. Thus, the *top category*, namely *Recipient* serves as the start symbol for parsing. While constructing the tree in a top down manner, the expansion of the tree branches is pruned by accessing additional collections of words. In particular, when a terminal, such as *FirstName* or *LastName*, corresponding to a generic logical object is derived the associated collections of words are accessed and the rule fires or not. For example, the ambiguity in the rule for *Addressee* which branches to *NatAddressee* and *Company* is easily solved because the branch for *NatAddressee* terminates in a handful of well-known keywords like "Frau" ("Mrs."), "Postfach" ("P.O. Box") etc.

9.2 Verification of Names

The distinction between syntactical categories is not the only source for making restrictions in an address part. Additionally, various semantical constraints exist that restrict the possibilities in a recipient object. For example, the dual relation between zip code and town name, the attachment of streets to towns and of towns to countries as well as the knowledge that certain persons or companies are suited at specific locations can be incorporated in both, the grammar and the lexicon. To do so, we introduce unique *identification tags* (ID's) for several real world objects like persons and places. The agreement of these ID's is tested within the unification component of our parsing system.

As we actually use D-PATR ([Karttunen 86]) for testing our grammar the following examples use a notation similar to that of this system. D-PATR is a development environment for context-free grammars using (equality-) unification for categorial attributes. For a given grammar, D-PATR generates a chart parser using a left-corner strategy, in this case initiated by the top category *Recipient*. This conforms to the expectation-driven analysis which is intended.

In order to explain the central idea, in Figure 27 a transcription of parts of the first three rules in Figure 26 is given. In the formalism of D-PATR no alternative or optional constituents are allowed. Therefore, a set of (alternative) rules in D-PATR correspond to one comprising rule in EBNF. Additionally, the following rules are enhanced by *attribute agreement pairs* where the equality of certain slots is required for two constituents respectively at the right hand side.

```
Recipient        -->  Addressee Direction
                      (Addressee PlaceID) = (Direction PlaceID)
                      (Recipient PersonID) = (Addressee PersonID)
                      (Recipient PlaceID) = (Addressee PlaceID)
Addressee        -->  NatAddressee
                      (Addressee PersonID) = (NatAddressee PersonID)
                      (Addressee PlaceID) = (NatAddressee PlaceID)
NatAddressee     -->  Title PersonName
                      (Title Sex) = (PersonName Sex)
                      (NatAddressee PersonID) = (PersonName PersonID)
                      (NatAddressee PlaceID) = (PersonName PlaceID)
                      (NatAddressee Sex) = (PersonName Sex)
```

Figure 27: Extended grammar with attributes for unification.

Just the first agreement pair of rule one deserves its name, it really is used for checking agreement of features. Here, the agreement of the attribute *PlaceID* is compared to the *Addressee* and the *Direction* constituent. Analogous, the first agreement pair in the third rule checks the *sex* of the person to be the same in *Title* and *PersonName*. Thus, it is guaranteed that a female name is preceded by the title "Frau" ("Mrs."). The two latter equations of rule one in Figure 27 are used only for inheritance, i.e. they guarantee that a composite constituent has references to its subordinator´s attributes. For example, the two latter agreement pairs in the rule for *Recipient* are only for inheritance: the attributes *PersonID* and *PlaceID* from the constituent *Addressee* are passed on to the superordinating constituent.

But where do these attribute values come from? Giving the above grammar excerpt to a parser, only comparison or inheritance of these values is possible. The source for them is the lexical knowledge slots that serve the parser with terminal attribute values for each lexical entry, i.e. each word. In Figure 28 some lexical entries are given that will serve us to parse a small example address in the next subsection.

To exploit the power of the available knowledge, it is necessary to attach all information concerning addresses to the lexical database. This means, all ID's of persons and places and the additional information, like *sex* in the grammar above, has to be incorporated into the lexicon. Therefore, the above lexicon has ID's for persons (PID), streets (SID), p.o. boxes (BID), towns (TID), and countries (CID).

```
Lexical Entries:
Title               --> "Herrn" + Mas-Temp
FirstName           --> "Andreas" + ARD-ID
LastName            --> "Dengel" + ARD-ID
PlaceName           --> "Kaiserslautern" + KL-ID
Lexical Templates:
Mas-Temp            --> (Sex) = mas
ARD-ID              --> (PersonID) = PID#4711 + Mas-Temp + DFKI-ID
DFKI-ID             --> (PlaceID StreetID) = SID#0815
                        (PlaceID POBoxID) = BID#2080 + KL-ID
KL-ID               --> (PlaceID TownID) = TID#6750 +D-ID
D-ID                --> (PlaceID CountryID) = CID#0049
```

Figure 28: Lexicon containing lexical entries and lexical templates.

For means of abbreviation, the lexicon is enhanced with templates which are expanded during loading the lexical entries. They are easy recognized in the above example lexicon because their occurrence is preceded by a plus sign (a "+" followed by the template name). While loading such a lexicon various attribute-value pairs may be generated. Figure 29 shows the entry concerning the word "Andreas" from Figure 28 after template expansion.

```
FirstName           --> "Andreas"
                        (FirstName PersonID) = PID#4711
                        (FirstName Sex) = mas
                        (FirstName PlaceID StreetID) = SID#0815
                        (FirstName PlaceID POBoxID) = BID#2080
                        (FirstName PlaceID TownID) = TID#6750
                        (FirstName PlaceID CountryID) = CID#0049
```

Figure 29: Lexical entry for "Andreas" after expansion.

9.3 A short example

In the following we illustrate how the recognition of an address is performed. Initially, in a preprocessing step a textual segmentation is performed. The output of an example address that serves as input for the subsequent syntactical pars is shown in Figure 30. The result of the textual segmentation must not be confused with that of layout segmentation since they may, but not must, be different, for example, the three word segments "D", "-" and "6750" are recognized as one layout object *word* by layout segmentation. But, this is not a main problem and therefore is neglected.

```
"Herrn"
"Andreas" "R" "." "Dengel"
"DFKI" "GmbH"
"Postfach" "2080"
"D" "-" "6750" "Kaiserslautern"
```

Figure 30: Input address for the parser.

For this case, the lexicon contains all of the input words. Thus, a full recognition of all ID's becomes possible. The result of the parse is shown in Figure 31 as a set of attribute-value pairs that is inherited up to the *top category* constituent. The integer number in the head of each list denotes the constituents of the parse, where the digit 0 stands for the top-level constituent *Recipient* itself and 1 and 2 are its sub-constituents. Abbreviations in brackets <> are pointers to "shared" attributes. For instance, in the feature set of constituent 1 (*Addressee*) pointer <0 PlaceID> means that its *PlaceID* is physically the same as that of constituent 0.

```
(0 (cat Recipient)     (PlaceID    (TownID TID#6750 )
                                   (CountryID CID#0049)
                                   (StreetID SID#0815))
                                   (POBoxID BID#2080))
                       (PersonID PID#4711))
(1 (cat Adressee)      (PlaceID <0 PlaceID>)
                       (PersonID <0 PersonID>))
(2 (cat Direction)     (PlaceID <0 PlaceID))
```

Figure 31: Feature sets as result of a parse.

The name lexicon is not assumed to be complete. Therefore, we define *default categories* for unknown words. For the address part these are all name categories, like names for persons, streets, towns etc. On the other side, keywords are assumed to be known by the dictionary. In that way, an unknown address can be recognized only by the keywords and numbers it contains, but, as it is impossible, it cannot be identified.

9.4 Results and Discussion

After text recognition, the content portions of our document structure were filled and linked to layout objects of class *word*. Right now, we are able to link single content portions to the logical objects, too. In review to Figure 31 which shows the parsing results we achieved such a linkage can be done as follows.

The *top category* of the grammar, i.e. *Recipient*, belongs in our case to the logical object *recipient*. For refining the specific logical structure for a recipient, we step through the parse tree, and in the case of a successful verification, appropriate specific logical objects are generated. Note, instantiating basic logical objects of the generic structure, not necessarily implies that preterminal nodes of the parse tree are reached. At each basic logical object, we can assign the corresponding parsing subtree by linking the concerned content portions to that logical object.

For example, remember the name part of the grammar in Figure 26. The rule for *PersonName* was

PersonName ::= { Degree } { FirstName } LastName .

Applied to the example in Figure 30, we obtain the simplified parse tree in Figure 32, where all attributes are omitted and recursive node paths are pulled at top. The latter occurs below the node *FirstName*, which actually has a recursive structure.

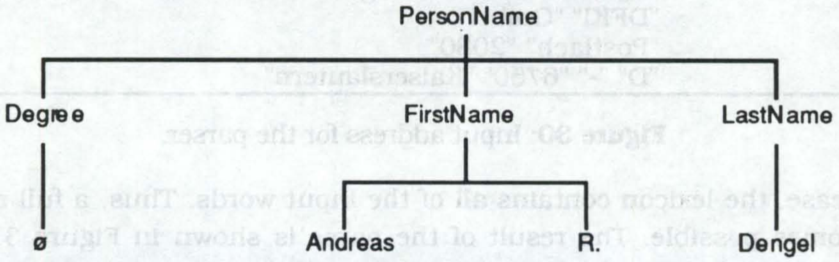


Figure 32: Parsing subtree for constituent *PersonName*.

Now, we can link the terminal nodes of the parse tree to the corresponding logical object. Thereby, it happens that more than one content portion is connected to one basic logical object, in our example this occurs for the node *first name*.

In doing so, we get the completed logical structure shown in Figure 33 where all content portions leading to a successful parsing result are related to the corresponding logical objects. Within one content portion all of these word hypotheses are rejected which do not lead to this successful parse (cf. hypotheses of last name).

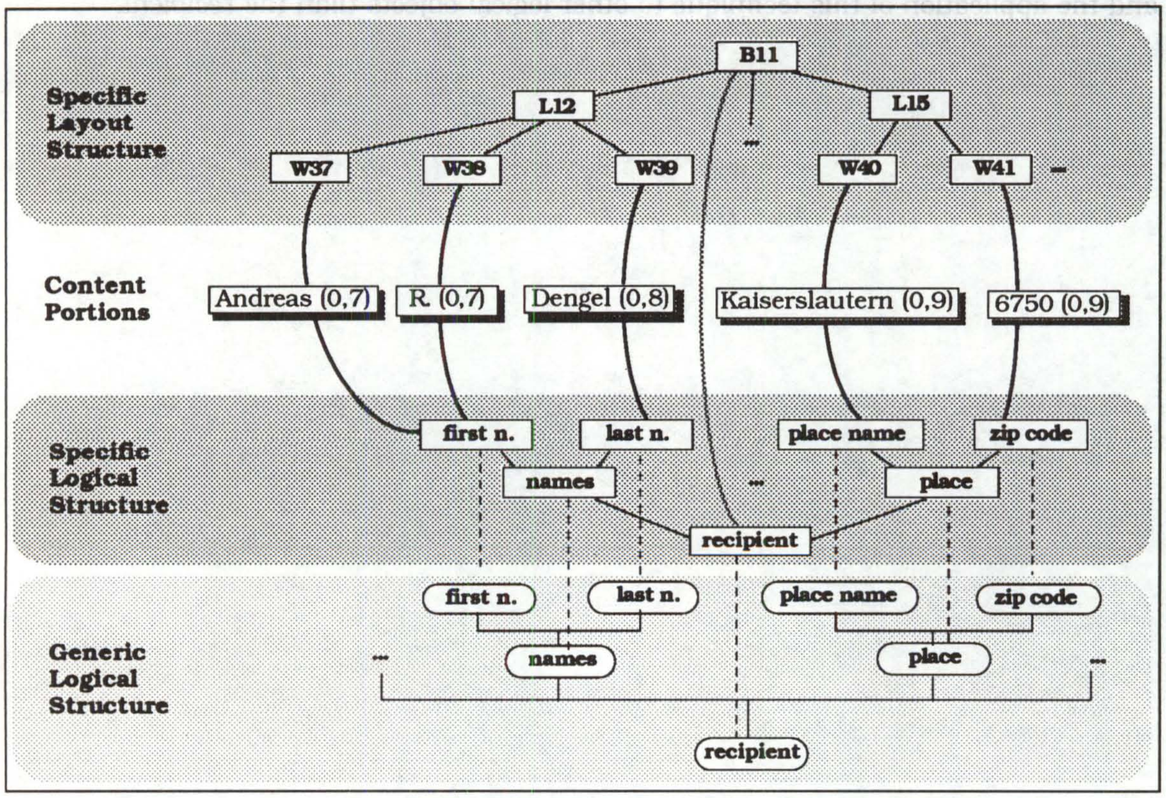


Figure 33: Part of the document structure after partial text analysis.

By using the text analysis component the way shown here, only an analysis after the text recognition phase is possible. Additionally, the knowledge used herein could also serve in building an integrated text recognition and text analysis component. The text recognition phase often generates a lot of word hypotheses that are rejected during text analysis because they do not lead to a complete parse. This hypotheses generation overhead could be prevented by a more co-operative parser that gives more feedback to other analysis components. For illustration consider the following situation.

In case the recipient of the letter is known by the system, it is possible to make very concrete predictions. For example, the (correct) identification of the recipient's name is sufficient to determine the following *direction* in the address block. The same holds for the inverse direction: after identifying the *direction* specification in the address, it is possible to list a set of candidates for the *persons* that can be reached at this place, e.g. the employees of a company.

To enable such predictions, a more flexible parser strategy is required and additional heuristics guiding the parser have to be found out.

The application of the presented technique to the well structured parts of a letter can be done analogous. Thus, date, salutation and sender can be analyzed with an appropriate grammar. For more complex parts of a letter, such as *subject* and entire letter *body*, it is much more difficult to find such a grammar. In fact, no powerful (descriptive) grammar for natural language is known by now. Only for the logical object *subject*, a simplified noun phrase grammar seems to handle this problem.

Our future activities will include both the integration of flexible parsing strategies and the application of this technique to other logical objects than the *recipient*.

10 Related Work

[Tang et al 91] gives a good survey of document analysis and document understanding in general. In his review, Tang tries to define the structure of documents systematically and theoretically using an algebra for documents and entropy functions. Similar to our approach, his document layout model is also based on the international standard ODA [ISO8613]. By document analysis, however, the author means the extraction of geometric structures from a document image (Smearing, Projection Profile Cuts, Run Length Encoding, etc.) which corresponds to layout extraction in ΠODA. Moreover, Tang characterizes document understanding as the mapping of geometric structures into logical structures of a document using tree transformation algorithms (form definition language, see also [Fujisawa & Nakano 90] below). This is slight distinct from our comprehension of document analysis. In contrast, we use geometrical (statistical) rules for logical labeling being a *first step* of understanding a document´s contents; further steps of text recognition and partial text analysis follow.

The FRESCO System — proposed by [Bayer 91] — is based on a own document representation formalism and uses a frame-like inheritance network considering layout as well as logical aspects. Each object can comprise two types of knowledge: domain-specific and domain-independent portions. While the second one is valid for all types of structured documents the first one describes spatial dependencies between different objects. For analysis process, different problem solving methods can be integrated using a rule-based control mechanism.

A similar approach has been realized by [Kreich et al 91]. In this approach, the analysis process is divided into image capturing, segmentation, text recognition, and labeling. In opposite to ΠODA, labeling is based on textual information and therefore, strongly depends on the recognition results. The overall result is also ODA-oriented, i. e., consists of a specific layout and logical structure where content portions of type text relates them.

Belaid [Belaid et al 90] has proposed a blackboard-based system that uses an ODA-like description of documents as model for analysis. The particular knowledge portions are not imbedded in the model layer. Instead, it is attached to the three processing phases *document segmentation*, *structure analysis*, and *text recognition*. Thus, the document model is only used for representation of the analysis results while in ΠODA it directs the analysis process. The global analysis process is performed by several specialists that are activated by choosing the best hypothesis.

[Yashiro et al 89] and [Fujisawa & Nakano 90] propose a method for document structure extraction based on the ODA standard. Similar to ΠODA, generic layout knowledge is used to transform document images into a hypertext representation. The generic layout knowledge is described in terms of rules in a special language called FDL (form definition language). As applications, images of technical papers are analyzed, and bibliographic items such as title and author can be extracted for subsequent filing.

Another approach focussing on technical papers is described in [Nagy 90] and [Viswanathan 90]. Using a special formalism, called x-y-tree, a respective image is recursively segmented with multiple alternating horizontal and vertical cuts according to publication-specific layout information. The resulting rectangular blocks are labeled in terms of functional components such as author, title, and abstract. The intention of this approach should allow library access to interactively select portions of a page image defined as functional components and give them as input to OCR.

Also related work can be found in particular application domains such as forms processing, address block location and cheque recognition. [Casey & Ferguson 90] describes a system called IFP (*Intelligent Forms Processing*) which accepts conventional forms for transforming them into a symbolical electronic representation in order to create corresponding records in a database. IFP allows the definition of different classes of forms using a special forms editor. Processing forms in IFP includes a classification of forms, adjustment of skew, the extraction of simple layout structures (segmentation), character recognition by decision trees, elimination of touched or broken characters, and a restricted lexical analysis. A sophisticated understanding of text, however, is not necessary in forms processing.

ABLS (address block location subsystem) is a rule-based system integrating different knowledge sources for the recognition of postal addresses [Srihari et al 87]. Here, phases of analysis comprise gray-scale and color thresholding, bottom-up segmentation to determine connected components, skew detection, discrimination of handwriting and machine-printing, and icon (rectangles in the image) detection. Additionally, a control mechanism is responsible for coordinating all these analysis modules and the exchange of corresponding results via a blackboard. Note that in ABLS spatial relationships between physical (layout) objects are explicitly expressed as rules.

11 CONCLUSIONS

Due to the large amount of information and the multitude of different kinds of printed material, paper-computer interfaces have to be developed allowing a transformation of printed information into an electronic representation.

To support and to simplify the processing and exchange of electronic documents, we concentrate on the international standard representation ODA for office documents. This standard provides mechanisms to describe a type of document, such as business letter, physically in terms of hierarchically nested layout objects as well as semantically in terms of logical objects.

For a better integration of paper documents into an electronic environment it is not sufficient to transform printed data to electronic medium, i.e. to use traditional systems only recognizing isolated characters, but rather provide information in a multitude of various aspects, such as of structure, layout styles, type of document, or the message conveyed.

For these reasons, modern systems for document analysis must consider various knowledge sources rather than improving classification techniques for isolated image patterns. Analogous to human reading techniques, such systems have to take account of structural knowledge (layout and logical structure), geometrical knowledge (reading order), syntactical knowledge (grammars), the type of document, character and word shapes, and much more.

As a first result of our research, this report presents the model-based document analysis system ΠODA for transforming printed business letters into an ODA conforming electronic representation. ΠODA has been implemented for the analysis of single-sided business letters in German; it runs on Sun SPARCstation under UNIX. All implementations including the dictionary are done in Common Lisp, except the scanner interface which is written in MPW Pascal on a Macintosh IIfx computer.

To support document analysis, the generic structures of the underlying architecture model are enhanced by locally relate knowledge portions to generic objects. Thus, the design of ΠODA reveals two fundamental advantages:

- All stages of document image analysis — layout extraction, logical labeling, text recognition and syntactic text analysis — are driven by the architecture model and additional knowledge sources.
- The representation of the resulting electronic document conforms to ODA.

Starting with the binary image of a given document, ΠODA extracts specific layout objects describing the physical presentation of the printed information. Already this segmentation task is model driven.

Taking the intrinsic geometric features of the layout objects as well as their arrangement on a sheet, logical labeling identifies mandatory logical objects within simple business letters and consequently generates respective instances. Note that logical labeling only is based on the idea to utilize knowledge about the arrangement and geometric shape of objects.

Instead of improving traditional OCR techniques, our intention is rather a search for possibilities to integrate knowledge from beyond the image data level. Therefore, ΠODA takes the context restrictions that are provided by logical labeling, i.e. the specific logical objects, as a starting point for text recognition using dictionaries.

After text recognition, some of the logically identified objects are characterized by typical and simple syntactic conventions in which the captured text is organized. Thus, syntactic knowledge about logical objects, e.g. the *recipient*, allow their structural refinement into more specific logical objects, such as *name*, *place*, etc.

In general, the advantages of ΠODA are the following: the resulting electronic representation is ODA conforming; the architecture of the whole system is divided in separate modules; and the overall system comprises all stages of document analysis. Especially, the inherent modularity enables a modification or extension of the system in a simple way. So, for example, additional methods for analysis tasks or revised knowledge can be integrated efficiently.

In this way, the final result of ΠODA provides a basis for several applications. such as mail or fax distribution, document indexing and filing, intelligent task processing, etc. Thus, subsequent services could start at a point where questions arise such as:

- What is the date of the letter?
- Who is the recipient of the letter?
- What is the letter´s subject?
- Who is the sender of the letter?

The syntax-oriented expectation-driven text analysis of ΠODA is successful for the smaller and well-structured constituents of a letter, such as *sender*, *recipient*, or *date*, but for the entire *letter body* it would fail.

To extend the capabilities of ΠODA, our current activities focus on the definition of message types which could be conveyed in a business letter. Each message type describes categories of information which a business letter may deal with, such as *offer*, *order*, *invoice* or *receipt*. Each of these message types might be seen as a frame representing a collection of entities of the real world being interrelated specifically. The entities may describe individuals, such as persons and animals, physical objects, abstract terms, or events which are characteristic for a certain message type. For example, an *offer* may be composed of entities such as *supplier*, *recipient*, *subject*, *price*, *number*, *date*.

As an ultimate goal of our research, one of these message types have to be identified in a current document. In other words, the entities belonging to a message type should be instantiated. Therefore, message types contain methods and rules, e.g. "check the supplier vocabulary and compare each entry with the name of the letter's sender", and constraints controlling the consistency among objects, e.g. "the sender is identical to the subscriber".

To obtain a certain expectation, a first inspection of the body concentrates on the determination of keywords which are characteristic for individual message types. In a first step, the letter body should be statistically examined with respect to word frequency. The occurrence and frequency of words, as well as their positions in the text are evaluated according to message type keywords and the system should come up with a set of weighted hypotheses about messages that might be conveyed in the given document. In a next step, these should be further investigated in a best-first manner by goal-oriented application of text analysis methods such as island parsing. Furthermore, we also focus on the construction of a structured lexicon representing syntactical and semantical knowledge being a prerequisite for successful higher-level interpretation of documents.

REFERENCES

[Aho et al 83] A. V. Aho, J. E. Hopcroft, J. D. Ullman. Data Structures and Algorithms. Addison-Wesley, Reading, Mass., 1983.

[Barr & Feigenbaum 81] A. Barr, E. A.Feigenbaum. The Handbook of Artificial Intelligence, vol. 1, William Kaufmann Inc., Los Angeles 1981.

[Bartneck 87] N. Bartneck. A Procedure for Transformation of Ikonical Information of Digitized Images in Data Structures for Interpretation, Doctoral Thesis, University of Braunschweig, Germany, 1987 (in German).

[Bayer 91] T. Bayer. Representation and Usage of Different Knowledge Sources for the Analysis of Structured Documents, Proc. of DAGM'91, München, Germany, Sept. 1991 (in German).

[Belaid et al 90] A. Belaid, J.J. Brault, Y. Chenevoy. Knowledge-Based System for Structured Document Recognition, *MVA'90*, IAPR Workshop on Mashine Vision Application, Tokyo, Nov. 1990, pp. 465-469.

[Brown 89] H. Brown. Standards for Structured Documents. The Computer Journal 32, 6 (1989), pp. 505–514.

[Bryan 89] M. Bryan. SGML: An Author's Guide to the Standard Generalized Markup Language. Addison-Wesley, England; Reading, MA et al., 1988.

[Casey & Ferguson 90] R. G. Casey, D. R. Ferguson. Intelligent Forms Processing. IBM Systems Journal, vol. 29, no. 3, November 1990, pp. 435-450.

[Dengel & Barth 88] A. Dengel, G. Barth. High Level Document Analysis Guided by Geometric Aspects, Internat. Journal on Pattern Recognition and AI, vol. 2, no. 4, Dez. 1988, pp. 641-656.

[Dengel & Barth 89] A. Dengel, G. Barth: ANASTASIL: A Hybrid Knowledge-based System for Document Layout Analysis. In IJCAI-89, Eleventh International Joint Conference on Artificial Intelligence Proceedings, Detroit, Michigan, August 1989, pp. 1249–1254.

[Dengel 90] A. Dengel. A Step Towards Understanding Paper Documents. Research Report RR-90-08, DFKI GmbH, Kaiserslautern, June, 1990.

[Dengel, 92] A. Dengel. ANASTASIL: A System for Low-Level and High-Level Geometric Analysis of Printed Documents, in: H. Baird, H. Bunke, K. Yamamoto (eds.), Structured Document Image Analysis, Springer Publ. (1992).

[Dengel et al 92] A. Dengel, A. Pleyer, R. Hoch. Fragmentary String Matching by Selective Access to Hybrid Tries. Submitted to: 11th International Conference on Pattern Recognition, The Hague, Aug./Sept., 1992.

[Dewes 92] S. Dewes. Prototype of a Word-based Text Recognition System, Bachelor Thesis, University of Kaiserslautern, Dep. of Computer Science, Germany, 1992 (in German).

[Elliman & Lancaster 90] D. G. Elliman, I. T. Lancaster. A review of segmentation and contextual analysis techniques for text recognition. Pattern Recognition, vol. 23, no. 3/4, 1990, pp. 337-346.

[Frank 91] U. Frank. Anwendungsnahe Standards der Datenverarbeitung: Anforderungen und Potentiale. Wirtschaftsinformatik, vol. 2, no. 2, April 1991, pp. 100-111 (in German).

[Fredkin 60] E. Fredkin. Trie memory. Communications of the ACM, vol. 3, no. 9, 1960, pp. 490-500.

[Fujisawa & Nakano 90] H. Fujisawa, Y. Nakano. A Top-Down Approach for the Analysis of Document Images. In Proc. of the SSPR '90, 1990, pp. 113–122.

[Hanson et al 76] A. R. Hanson, E. M. Riseman, E. Fisher. Context in Word Recognition. Pattern Recognition, vol. 8, 1976, pp. 35-45.

[Hönes et al 90] F. Hönes, R. Bleisinger, A. Dengel. Intelligent Word-Based Text Recognition. In High-Speed Inspection Architectures, Barcoding, and Character Recognition, Chen, M.J., SPIE — The International Society for Optical Engineering, 1991, pp. 305–316.

[Hönes et al 91] F. Hönes, E.G. Haffner, F. Fein, A. Dengel. A Hybrid Approach for Document Image Segmentation and Encoding. In Proc. ICDAR '91, 1st International Conference on Document Analysis and Recognition, 1991, pp. 444-453.

[Horak 85] W. Horak. Office Document Architecture and Office Document Interchange Formats: Current Status of International Standardization. In IEEE Computer, October 1985, pp. 50-60.

[Hough 89] D. Hough, The Paperless Office, Byte, July 1989, pp. 241-246.

[ISO8613] ISO 8613, Information Processing — Text and Office Systems — Office Document Architecture (ODA) and Interchange Format, vol. I-III, parts 1-8, 1988.

[ISO8879] ISO 8879, Information Processing — Text and Office Systems — Standard Generalized Markup Language (SGML), 1986.

[ISO9735] ISO 9735, Electronic data interchange for administration, commerce and transport (EDIFACT), Application Level Syntax Rules, 1988.

[Joloboff 89] V. Joloboff. Document Represenation: Concepts and standards. In J. André, R. Furuta, V. Quint (eds.), Structured Documents, Cambridge University Press, 1989, pp. 75-105.

[Kaplan & Bresnan 82] R. M. Kaplan, J. Bresnan. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In: J. Bresnan (ed.): The Mental Representation of Grammatical Relations; MIT Press, Cambridge, Mass., 1982, pp. 173-182.

[Karttunen 86] L. Karttunen. D-PATR: A Development Environment for Unification-Based Grammars; Artificial Intelligence Center, SRI International and Center for the Study of Language and Information, Stanford University (Report no. CSLI-86-68); Stanford 1986.

[Knuth 73] D. E. Knuth. The Art of Computer Programming. vol. III: Sorting and Searching. Addison-Wesley, Reading, Mass., 1973.

[Kohonen & Reuhkala 78] T. Kohonen, E. Reuhkala. A Very Fast Associative Method for the Recognition and Correction of Misspelt Words, Based on Redundant Hash Addressing. Proceedingds of the Int. Joint Conference on Pattern Recognition,. November 1978, pp. 807-809.

[Kreich et al 91] J. Kreich, A. Luhn, G. Maderlechner. An Experimental Environment for Model Based Document Analysis. In Proc. ICDAR '91, 1st International Conference on Document Analysis and Recognition, 1991, pp. 50.

[Nagy 90] G. Nagy. Towards a Structured-Document-Image Utility. In Proc. of the SSPR '90, 1990, pp. 293–309.

[Mattos et al 90] N. Mattos, B. Mitschang, A. Dengel, R. Bleisinger. Integration of Document Representation, Processing and Management. In Proc. of the COIS-90 — 5th Conference on Office Information Systems, ACM, Cambridge, MA, April 1990, pp. 118–122.

[Pereira & Warren 80] F. C. N. Pereira, D. H. D. Warren. Definite Clause Grammars for Language Analysis — A Survey of the Formalism and a Comparison with Augmented Transition Networks; Artificial Intelligence 13, 1980, pp. 231-278.

[Peterson 80a] J. L. Peterson. Computer Programs for Detecting and Correcting Spelling Errors. Communications of the ACM, vol. 23, no. 12, December 1980, pp. 676-687.

[Postl 86] W. Postl. Detection of Linear Oblique Structures and Skew Scan in Digitized Documents. In Proceedings of the 8th Int. Conference on Pattern Recognition, Paris, 1986, pp. 240.

[Quint 89] V. Quint. Systems for the Manipulation of Structured Documents. In J. André, R. Furuta, V. Quint (eds.), Structured Documents, Cambridge University Press, 1989, pp. 39-74.

[Röhl 89] R. Röhl. Implementation of the Hypothesis Generator of a Word-based Technique for Text Recognition, Bachelor Thesis, University of Stuttgart, Dep. of Computer Science, Germany, 1989 (in German).

[Schäfer & Fröschle 86] M. Schäfer, H. P. Fröschle. Die Vision vom papierlosen Büro, Funkschau 19, 1986, pp. 45-59.

[Schürmann 78] J. Schürmann. A Multifont Word Recognition System for Postal Address Reading. IEEE Transactions on Computers, vol. C-27, no. 8, 1978, pp. 721-732.

[Sinha 87] R. M. K. Sinha. Some Characteristics for Dictionary Organization with Digital Search. IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-17, no. 3, May/June 1987, pp. 520-527.

[Sinha & Prasada 88] R. M. K. Sinha, B. Prasada. Visual Text Recognition Through Contextual Processing. Pattern Recognition, vol. 21, no. 5, 1988, pp. 463-479.

[Srihari et al 87] S.N. Srihari, C.-H. Wang, P.W. Palumbo, J.J. Hull. Recognizing Address Blocks on Mail Pieces: Specialized Tools and Problem-Solving Architecture, AI Magazine, vol. 8, no. 4, Winter 1987, pp. 25-36.

[Tang et al 91] Yuann Y. Tang, Ching Y. Suen, Chang D. Yan, M. Cheriet. Document Analysis and Understanding: A Brief Survey, In Proceedings of the ICDAR-91, St. Malo, France (1991), pp. 17-31.

[Viswanathan 90] M. Viswanathan. Annalysis of Scanned Documents - a Syntactic Approach. In Proc. of the SSPR '90, 1990, pp. 450–459.

[Wang & Srihari 89] D. Wang, S.N. Srihari. Classification of Newspaper Image Blocks Using Texture Analysis. In Computer Vision, Graphics, and Image Processing, 47 (1989), pp. 327–352.

[Wells et al 90] C. J. Wells et al., Fast Dictionary Look-Up For Contextual Word Recognition, Pattern Recognition, vol. 23, no. 5, 1990, pp. 501-508.

[Yashiro et al 89] H. Yashiro, T. Murakami, Y Shima, Y. Nakano, H. Fujisawa. A New Method of Document Structure Extraction using Generic Layout Knowledge. In Proc. of the International Workshop on Industrial Applications of Machine Intelligence and Vision, Tokyo, 1989, pp. 282-287.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
D-6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse bezogen werden.

Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all publisched papers so far can be ordered from the above address.

The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-90-16
Franz Baader, Werner Nutt: Adding Homomorphisms to Commutative/Monoidal Theories, or: How Algebra Can Help in Equational Unification
25 pages

RR-90-17
Stephan Busemann:
Generalisierte Phasenstrukturgrammatiken und ihre Verwendung zur maschinellen Sprachverarbeitung
114 Seiten

RR-91-01
Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, Gert Smolka: On the Expressivity of Feature Logics with Negation, Functional Uncertainty, and Sort Equations
20 pages

RR-91-02
Francesco Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, Werner Nutt: The Complexity of Existential Quantification in Concept Languages
22 pages

RR-91-03
B.Hollunder, Franz Baader: Qualifying Number Restrictions in Concept Languages
34 pages

RR-91-04
Harald Trost: X2MORF: A Morphological Component Based on Augmented Two-Level Morphology
19 pages

RR-91-05
Wolfgang Wahlster, Elisabeth André, Winfried Graf, Thomas Rist: Designing Illustrated Texts: How Language Production is Influenced by Graphics Generation.
17 pages

RR-91-06
Elisabeth André, Thomas Rist: Synthesizing Illustrated Documents: A Plan-Based Approach
11 pages

RR-91-07
Günter Neumann, Wolfgang Finkler: A Head-Driven Approach to Incremental and Parallel Generation of Syntactic Structures
13 pages

RR-91-08
Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, Thomas Rist: WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation
23 pages

RR-91-09
Hans-Jürgen Bürckert, Jürgen Müller, Achim Schupeta: RATMAN and its Relation to Other Multi-Agent Testbeds
31 pages

RR-91-10
Franz Baader, Philipp Hanschke: A Scheme for Integrating Concrete Domains into Concept Languages
31 pages

RR-91-11
Bernhard Nebel: Belief Revision and Default Reasoning: Syntax-Based Approaches
37 pages

RR-91-12
J.Mark Gawron, John Nerbonne, Stanley Peters: The Absorption Principle and E-Type Anaphora
33 pages

RR-91-13
Gert Smolka: Residuation and Guarded Rules for Constraint Logic Programming
17 pages

RR-91-14
Peter Breuer, Jürgen Müller: A Two Level Representation for Spatial Relations, Part I
27 pages

RR-91-15
Bernhard Nebel, Gert Smolka:
Attributive Description Formalisms ... and the Rest of the World
20 pages

RR-91-16
Stephan Busemann: Using Pattern-Action Rules for the Generation of GPSG Structures from Separate Semantic Representations
18 pages

RR-91-17
Andreas Dengel, Nelson M. Mattos:
The Use of Abstraction Concepts for Representing and Structuring Documents
17 pages

RR-91-18
John Nerbonne, Klaus Netter, Abdel Kader Diagne, Ludwig Dickmann, Judith Klein:
A Diagnostic Tool for German Syntax
20 pages

RR-91-19
Munindar P. Singh: On the Commitments and Precommitments of Limited Agents
15 pages

RR-91-20
Christoph Klauck, Ansgar Bernardi, Ralf Legleitner
FEAT-Rep: Representing Features in CAD/CAM
48 pages

RR-91-21
Klaus Netter: Clause Union and Verb Raising Phenomena in German
38 pages

RR-91-22
Andreas Dengel: Self-Adapting Structuring and Representation of Space
27 pages

RR-91-23
Michael Richter, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Akquisition und Repräsentation von technischem Wissen für Planungsaufgaben im Bereich der Fertigungstechnik
24 Seiten

RR-91-24
Jochen Heinsohn: A Hybrid Approach for Modeling Uncertainty in Terminological Logics
22 pages

RR-91-25
Karin Harbusch, Wolfgang Finkler, Anne Schauder:
Incremental Syntax Generation with Tree Adjoining Grammars
16 pages

RR-91-26
M. Bauer, S. Biundo, D. Dengler, M. Hecking, J. Koehler, G. Merziger:
Integrated Plan Generation and Recognition
- A Logic-Based Approach -
17 pages

RR-91-27
A. Bernardi, H. Boley, Ph. Hanschke, K. Hinkelmann, Ch. Klauck, O. Kühn, R. Legleitner, M. Meyer, M. M. Richter, F. Schmalhofer, G. Schmidt, W. Sommer:
ARC-TEC: Acquisition, Representation and Compilation of Technical Knowledge
18 pages

RR-91-28
Rolf Backofen, Harald Trost, Hans Uszkoreit:
Linking Typed Feature Formalisms and Terminological Knowledge Representation Languages in Natural Language Front-Ends
11 pages

RR-91-29
Hans Uszkoreit: Strategies for Adding Control Information to Declarative Grammars
17 pages

RR-91-30
Dan Flickinger, John Nerbonne:
Inheritance and Complementation: A Case Study of *Easy* Adjectives and Related Nouns
39 pages

RR-91-31
H.-U. Krieger, J. Nerbonne:
Feature-Based Inheritance Networks for Computational Lexicons
11 pages

RR-91-32
Rolf Backofen, Lutz Euler, Günther Görz:
Towards the Integration of Functions, Relations and Types in an AI Programming Language
14 pages

RR-91-33
Franz Baader, Klaus Schulz:
Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures
33 pages

RR-91-34
Bernhard Nebel, Christer Bäckström:
On the Computational Complexity of Temporal Projection and some related Problems
35 pages

RR-91-35
Winfried Graf, Wolfgang Maaß: Constraint-basierte Verarbeitung graphischen Wissens
14 Seiten

RR-92-02
Andreas Dengel, Rainer Bleisinger, Rainer Hoch, Frank Hönes, Frank Fein, Michael Malburg:
Π_{ODA}: The Paper Interface to ODA
53 pages

RR-92-03
Harold Boley:
Extended Logic-plus-Functional Programming
28 pages

RR-92-04
John Nerbonne: Feature-Based Lexicons: An Example and a Comparison to DATR
15 pages

RR-92-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Michael Schulte, Rainer Stark: Feature based Integration of CAD and CAPP
19 pages

RR-92-08
Gabriele Merziger: Approaches to Abductive Reasoning - An Overview -
46 pages

DFKI Technical Memos

TM-91-01
Jana Köhler: Approaches to the Reuse of Plan Schemata in Planning Formalisms
52 pages

TM-91-02
Knut Hinkelmann: Bidirectional Reasoning of Horn Clause Programs: Transformation and Compilation
20 pages

TM-91-03
Otto Kühn, Marc Linster, Gabriele Schmidt: Clamping, COKAM, KADS, and OMOS: The Construction and Operationalization of a KADS Conceptual Model
20 pages

TM-91-04
Harold Boley (Ed.):
A sampler of Relational/Functional Definitions
12 pages

TM-91-05
Jay C. Weber, Andreas Dengel, Rainer Bleisinger: Theoretical Consideration of Goal Recognition Aspects for Understanding Information in Business Letters
10 pages

TM-91-06
Johannes Stein: Aspects of Cooperating Agents
22 pages

TM-91-08
Munindar P. Singh: Social and Psychological Commitments in Multiagent Systems
11 pages

TM-91-09
Munindar P. Singh: On the Semantics of Protocols Among Distributed Intelligent Agents
18 pages

TM-91-10
Béla Buschauer, Peter Poller, Anne Schauder, Karin Harbusch: Tree Adjoining Grammars mit Unifikation
149 pages

TM-91-11
Peter Wazinski: Generating Spatial Descriptions for Cross-modal References
21 pages

TM-91-12
Klaus Becker, Christoph Klauck, Johannes Schwagereit: FEAT-PATR: Eine Erweiterung des D-PATR zur Feature-Erkennung in CAD/CAM
33 Seiten

TM-91-13
Knut Hinkelmann:
Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

TM-91-14
Rainer Bleisinger, Rainer Hoch, Andreas Dengel: ODA-based modeling for document analysis
14 pages

TM-91-15
Stefan Bussmann: Prototypical Concept Formation
An Alternative Approach to Knowledge Representation
28 pages

TM-92-01
Lijuan Zhang:
Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

DFKI Documents

D-91-01
Werner Stein , Michael Sintek: Relfun/X - An Experimental Prolog Implementation of Relfun
48 pages

D-91-02
Jörg P. Müller: Design and Implementation of a Finite Domain Constraint Logic Programming System based on PROLOG with Coroutining
127 pages

D-91-03
Harold Boley, Klaus Elsbernd, Hans-Günther Hein, Thomas Krause: RFM Manual: Compiling RELFUN into the Relational/Functional Machine
43 pages

D-91-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1990
93 Seiten

D-91-06
Gerd Kamp: Entwurf, vergleichende Beschreibung und Integration eines Arbeitsplanerstellungssystems für Drehteile
130 Seiten

D-91-07
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
TEC-REP: Repräsentation von Geometrie- und Technologieinformationen
70 Seiten

D-91-08
Thomas Krause: Globale Datenflußanalyse und horizontale Compilation der relational-funktionalen Sprache RELFUN
137 Seiten

D-91-09
David Powers, Lary Reeker (Eds.):
Proceedings MLNLO´91 - Machine Learning of Natural Language and Ontology
211 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-10
Donald R. Steiner, Jürgen Müller (Eds.):
MAAMAW´91: Pre-Proceedings of the 3rd European Workshop on „Modeling Autonomous Agents and Multi-Agent Worlds"
246 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-11
*Thilo C. Horstmann:*Distributed Truth Maintenance
61 pages

D-91-12
Bernd Bachmann:
$H^{iera}C_{on}$ - a Knowledge Representation System with Typed Hierarchies and Constraints
75 pages

D-91-13
International Workshop on Terminological Logics
Organizers: Bernhard Nebel, Christof Peltason, Kai von Luck
131 pages

D-91-14
Erich Achilles, Bernhard Hollunder, Armin Laux, Jörg-Peter Mohren: KRIS : Knowledge Representation and Inference System
- Benutzerhandbuch -
28 Seiten

D-91-15
Harold Boley, Philipp Hanschke, Martin Harm, Knut Hinkelmann, Thomas Labisch, Manfred Meyer, Jörg Müller, Thomas Oltzen, Michael Sintek, Werner Stein, Frank Steinle:
μCAD2NC: A Declarative Lathe-Worplanning Model Transforming CAD-like Geometries into Abstract NC Programs
100 pages

D-91-16
Jörg Thoben, Franz Schmalhofer, Thomas Reinartz:
Wiederholungs-, Varianten- und Neuplanung bei der Fertigung rotationssymmetrischer Drehteile
134 Seiten

D-91-17
Andreas Becker:
Analyse der Planungsverfahren der KI im Hinblick auf ihre Eignung für die Abeitsplanung
86 Seiten

D-91-18
Thomas Reinartz: Definition von Problemklassen im Maschinenbau als eine Begriffsbildungsaufgabe
107 Seiten

D-91-19
Peter Wazinski: Objektlokalisation in graphischen Darstellungen
110 Seiten

Π_{ODA}: The Paper Interface to ODA

Andreas Dengel, Rainer Bleisinger, Rainer Hoch, Frank Hönes, Frank Fein, Michael Malburg

RR-92-02
Research Report