Recent Advances in the CLiDE Project: Logical Layout Analysis of Chemical Documents

Anikó Simon[†] and A. Peter Johnson*[,†]

ICAMS, School of Chemistry, University of Leeds, Leeds LS2 9JT, U.K.

Received June 30, 1996[⊗]

The CLiDE system for chemistry document image processing consists of three major parts: physical layout analysis, recognition of the primitives, and logical layout analysis. This paper presents the new methods for logical layout analysis: role assignment to the elements of the document with a use of a style description language. The results are illustrated by application to generic reaction interpretation.

The CLiDE (Chemical Literature Data Extraction) system, which aims to automate the process of abstracting chemical information from the literature, has been under continuous development at the University of Leeds since 1990. The methods used for the recognition of chemical structures have already been reported.[1,2] However a more complete understanding of the content of a publication requires much more than just the structures. At the very minimum, any logical association which exists between structure and text segments must be determined. The starting point for this process is the assignment of roles to the various elements found in a chemistry publication, and this paper reports our progress in this area.

The CLiDE process for document image processing consists of three major parts: physical layout analysis, recognition of the primitives, and logical layout analysis. The first phase takes a document image which has been captured by scanning, and performs physical layout analysis.[3] This process divides the image into its physical components: words, lines, text and graphic blocks, columns, and stripes and is often termed the segmentation process. In the next phase the basic components, the primitives of the image, are recognized. Primitive recognition means the creation of a description for the characters, lines, and dashed lines, at a higher level than the raw pixel-based bitmap representation. The last phase of the CLiDE document image processing system at present is the layout perception phase. There is an interesting parallel with a typesetting of word-processing system, in which there is a phase when the author of the document associates labels (such as title, author, abstract, header, etc.) with the paragraphs. Thus the author expresses the logic of the document structure and leaves the system to format it automatically according to a set of rules (e.g., the document-class in case the LAT$_E$X typesetting). In document processing, the problem is reversed: the formatted blocks are given, leaving us the problem of finding their relationships. This process is termed **logical layout analysis** (Figure 1). Logical layout analysis is based on two properties of the objects: content and context. In this context, the logical structure, i.e., layout, consists of a hierarchy of visually observable separate components that make up the document, e.g., sections and subsections. The automatic determination of the logical structure of documents is a key step in automatic document handling and has many desirable consequences; i.e., automatic keywording, automatic retrieval of bibliographic information, and support for hierarchical browsing.

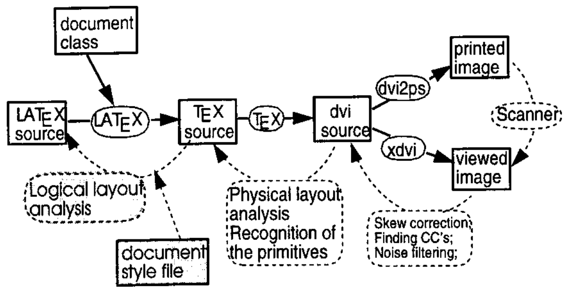


Figure 1. LAT$_E$X(T$_E$X) and the reverse stream: document image processing. The processes in the shaded boxes are discussed in this paper.

This paper presents a chemistry specific Document Format Description Language (DFDL), which describes chemistry journals and book styles (section 2). With the assistance of DFDL style files, the proposed methods determine the logical structure of the document images (sections 3 and 4). In the remaining part of the this paper the results are analyzed and discussed.

1. RELATED WORK

Although the problem of logical structure analysis has attracted a vast amount of research attention in the past decade, it is still in its infancy. Two main approaches have already emerged from the published works: one applies typical AI techniques (such as semantic nets or frames); the other applies formal linguistic methods.

Bayer *et al.*[4] use a semantic net to model the logical objects of business letters. The attributes of the objects which are considered are geometric properties, spatial relationships, lexical entities, and lexical relationships. The model strongly depends on the OCR (Optical Character Recognition) results, i.e., in case of high OCR accuracy, the results of the labeling are highly accurate, and vice versa. Construction of the semantic net requires data extraction from large image sets and is therefore time consuming. A different approach, also using AI techniques, was published by Niyogi *et al.*[5] for newspaper structure retrieval. This method applies a knowledge-based structure derivation system, where a hierarchical rule-based control system guides the classifications. The

[†] E-mail: aniko@mi.leeds.ac.uk and johnson@mi.leeds.ac.uk. URL: http://chem.leeds.ac.uk/ICAMS/.

[⊗] Abstract published in *Advance ACS Abstracts,* December 15, 1996.

rules, formulated in terms of first-order predicates, describe the control mechanism and the strategy of the algorithm and the knowledge concerning the geometric properties and spatial relationships of the objects. This model is a complex but robust system for structure recognition of short articles such as newspaper articles. Summers[6] has proposed a document structure classification algorithm, based on layout prototypes. The prototypes are constructed considering the geometric properties and spatial relationships of the objects and some information about the content of the blocks. The document objects are analyzed by considering their distances from the prototypes. The advantages of this method are minimal dependency on OCR accuracy and flexible handling of novel document styles; however, there are some disadvantages: a lower rate of overall classification accuracy (about 86%) and an increased computational complexity, due to complex distance evaluation. Still, this method is the most general and flexible amongst the ones discussed.

Solving the problem of layout perception with formal linguistic methods was initiated by form documents. Higashino *et al.*[7] proposed a flexible format perception method for Japanese business forms using a document definition language. A similar approach was reported by Tang et al.,[8] who have developed a form description language for perception of financial documents (such as bank checks).

The layout perception module implemented in CLiDE uses a formal linguistic approach but differs from the similar systems discussed above by its recognition power. The style files written in DFDL are journal-specific, rather then document-specific. This means that a single style profile covers a wide range of documents. Forms, being special-purpose documents, have a finite set of fixed structures, which can be described with context-free language structures. Journals have context-dependent logical structure; therefore, only a language of higher complexity, namely a context-dependent language, can describe their structure.

2. THE DOCUMENT FORMAT DESCRIPTION LANGUAGE

The International Standard ISO 86131:1989(E)[9] defines the logical structure of a document as follows: "The **logical structure** is the result of the division and subdivision of the content of a document into increasingly smaller parts on the basis of the human-perceptible meaning of the content—for example, into chapters, sections, subsections, and paragraphs".

A **logical object** is an element of the specific logical structure of a document. The defined classes of the logical objects are basic, composite, and root objects. Logical object categories such as chapter, section, and paragraph are application dependent and can be defined using an object class mechanism.[9] Such a class mechanism is defined with the documental format description language (DFDL).

DFDL describes the physical appearance and the mutual relationships of the logical objects in a document. Objects of the documents compose a finite set. This set has been established by examining various types of chemical journals and books. The elements of the set, such as "scheme", "figure", "title", etc., are keywords of the language. It has been observed that the logical objects can be characterized with five appearance attributes (Figure 2):

1. the font attribute, defined by its size (e.g., 32 pixels), its type (e.g., Times) and its style (e.g., bold);

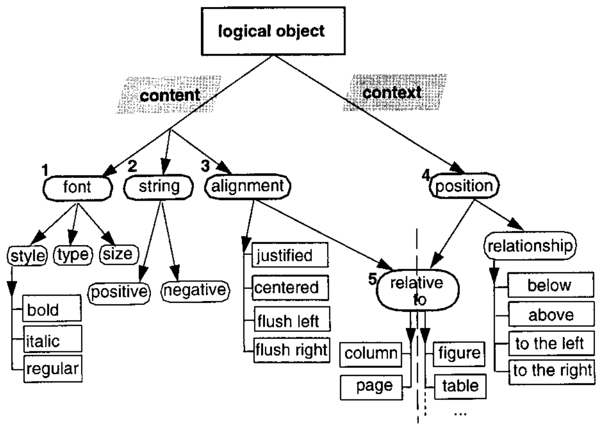


Figure 2. Representation attributes of the logical objects.

2. the string attribute, which can be a positive or a negative attribute of an object, i.e., the required or the forbidden character string;
3. the alignment attribute which can take four values: justified, centered, flush left, and flush right;
4. the position attribute (e.g., below a, above a);
5. the relativity attribute expresses either the relation of alignment (e.g., centered in a column) or the relation of position (e.g., below a table).

The font and the alignment attributes are essential for identification of the objects of the document. The string attribute is important, whenever it is appropriate for the object (e.g., string patterns for the "headers" are appropriate, but not for the "titles"). The other two attributes—alignment and relativity—vary according to the object type. These variations define the three basic *types of statement* for description in the DFDL:

Aligned type contains the font and the alignment attributes only. It is mainly used for the description of headers and footers, as no position and/or relativity information is necessary for these items, because they are always placed at the top and bottom of the page.

RelAligned type groups the font, the alignment, and the relativity attributes. It is mainly used for describing independent objects that do not demand the presence of any other object such as formulas without formula identification labels, figures without captions, etc.

RelPosAligned type contains all the attributes. This is the most frequently used type, because most objects appear in a relative position to another object, e.g., figure captions appear only with a figure, i.e., below or above a figure.

Each of these types has an additional component: the positive string form of the object. It represents the typical word or words of an object in a typical order. Negative symbol definition has been left out from this representation. To express this, the language allows the specification of the string format of an object. This format can be compared to the format control string in the C language. For example, the header of a paper usually contains the journal name, the volume number, and the interval of the pages of the paper. This sequence of information can be expressed with the format string of DFDL. The actual volume number and the page interval are integer type *variables* of the style file, while the name of the journal is a *string constant*. The DFDL definition for the header is the following:

```
<Header>    ::= (A) | if (C) (A) | if (C) (A) else Header
<A>         ::= F Align Sf | F Align Sf ; A
<F>         ::= Size Style Type
<Size>      ::= int
<Style>     ::= { bold, italic, plain, NULL}
<Type>      ::= { times, helv, NULL}
<Align>     ::= { justified, centered, flushleft, flushright}
<Sf>        ::= NULL | "Character"V | "Character"V Sf
<Character> ::= {ASCII}\{<} | {ASCII}\{<} Character
<V>         ::= <Variable
<Variable>  ::= { int, char, string}
<C>         ::= { odd-page, even-page, title-page, odd-title-page,
        even-title-page, non-title-page, even-non-title-page,
        odd-non-title-page, last-page, odd-last-page, even-last-page}
```

The style file written in DFDL for the Journal of the American Chemical Society (JACS) would contain the following header description:

```
variables
...
<YEAR          int
<VOL           int
<FIRST_PAGE    int
<LAST_PAGE     int
<PAGE_NUMBER   int

description
...
Header  if ( even-title-page )
             ( 32 italic times centered
"J.Am.Chem.Soc."<YEAR","<VOL","<FIRST_PAGE"-"<LAST_PAGE );
             ( 32 bold NULL flush_left <PAGE_NUMBER )
        else
             ( ... )
```

According to this description, the header contains two blocks on the even title pages. The first block is written in Times italic letters with an average size of 32 pixels. It has centered alignment on the page, and it contains the given format string, with the journal name constant and the year, volume, and page interval variables. The second block is written in bold letters also with 32 pixel size. The font type of this block, as NULL indicates, is the default font type of the page. The block has flush left alignment on the page, and its string content consists of only one integer variable: the "<**PAGE_NUMBER**". It has to be emphasized that this type of a description focuses on the typical occurrence (i.e., from style and size) of an object, rather than showing all the characteristics of the object (e.g., this description does not differentiate between parts of the object presented in different font style or size). This design decision has been based on the fact that the input to the system, which is the output of an OCR system, is not accurate enough to capture all the small variances. The authors believe that this decision could be modified if the source of the input was changed.

A journal description written in DFDL consists of two parts: the description of the variables (empty if there are none) and the description of the layout objects, called the description body. The layout object names are predefined, and they are the object-keywords of the language. Currently

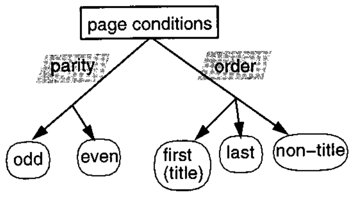


Figure 3. The page attributes determining the page conditions.

28 different object-keywords are in use (Table 1) based on the current chemical literature. This set could be extended in the future by modifying only the configuration tables of the system. The parser is general enough to enable addition of new objects, i.e., keywords, which are of similar type to the other objects. The majority of the objects are defined with one of the three basic object types of the language: **Aligned**, **RelAligned**, or **RelPosAligned**. Exceptions, for example, are the "Footer" and the "Header" objects, which are defined with an **Aligned** type expressed in a conditional structure of the language and, therefore, their type is the **CondAligned** type. In a DFDL code the object description is required to start with the object-keyword, as implied in the JACS header example. This makes parsing the language much easier and quicker.

Variables originate from the object descriptions in a DFDL code containing format string specifications using variables. If variables are used, they must be declared prior to their usage in the variable section of the DFDL code. A variable declaration has the following form:

$$variable\_name \quad variable\_type$$

The variable names (e.g., "**PAGE_NUMBER**", "**ABSTRACT_TEXT**", etc.) can be freely chosen at the time of the specification. The only requirements are that the names are written in upper case letters and that they start with the symbol "<". The types of the variables are also set by the time of the specification, and they have to be one of the following five types: **int**, **char**, **float**, **string**, or the formatted string type of the "C" language (e.g., "%d-%d").

Page conditions, such as "**even-title-page**" in the JACS header description example, are also predefined keywords of the language. They are important for differentiating between layouts occurring under the described page conditions, i.e., to express the layout differences (e.g., in the header) occurring on different page types (e.g., title page and nontitle page). Thus the page conditions always appear as conditional statements in ramification constructions of DFDL. The set of the page conditions is determined by two nonrelated page attributes (Figure 3): one is the parity of the page number, the other is the order of the page with respect to the article. The first attribute results in two mutually-excluding page conditions: "**even-page**" and "**odd-page**". The second attribute results in three excluding conditions: "**title-page**", "**last-page**", and "**nontitle-page**". Combination of these two attributes is allowed and results in six more strict conditions, such as "**even-title-page**", "**even-last-page**", "**even-nontitle-page**", etc.

DFDL contains all of the three structured constructions of the programming languages: sequence and ramification have already been mentioned; iteration is used for description of the tables. Tables are described with iteration of table rows and table columns.

3. FEATURE EXTRACTION OF THE BLOCKS

Block features are implicitly defined in the DFDL by the attributes of the language. The correct extraction of the block features is important, as the logical layout analysis is based on the extracted feature values.

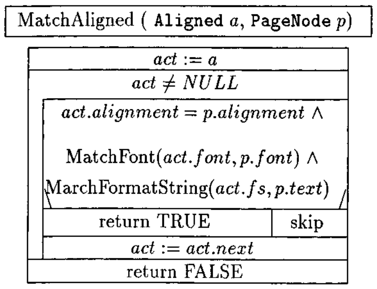
Font attributes are already identified by the OCR engine. Alignment, position, and relativity attributes are in close connection with each other. To determine these attributes, we examine first whether a block fits into a column (when there are at least two columns on the page) or just into the page. The number of columns and their approximate width in pixels are defined in the style file. Having extracted this information, determining the alignment is a trivial task. The relation of the blocks to the columns and to the page has already been captured by examining where the blocks fit. To establish the other relativity attribute, i.e., the relative position, we map the closest neighbors of each block in four directions: below, above, to the left, and to the right. This information will help to match block relationships to the descriptions, such as below a table, etc.

4. LOGICAL STRUCTURE IDENTIFICATION

After the reading of the style file and the extraction of the block features, the last remaining task is identification of the blocks, i.e., their association with object name labels defined in the style file. This task consists of *matching* the extracted block features against the description of the document given in the style file. This process is done by a recursive parser (section 4.1).

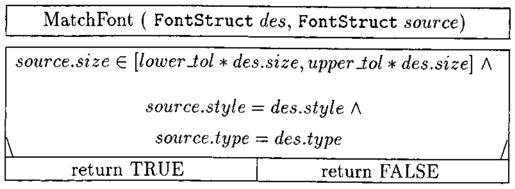
The most problematic part of the matching algorithm is comparison of the extracted features with the described ones. The comparison has to be tolerant of the faults of the previous phases, and, additionally, it has to be able to correct them to a certain degree. For example, the format string of the abstract block, in most styles, would be expressed as "**Abstract**:"**TEXT**, where **TEXT** is a string type variable. If the OCR has identified the first word of one of the blocks as "**Ah8tract**", the system should be able to recognize that the two strings match on six positions out of eight, i.e., that there is a match of 75%. This problem is known as the problem of string matching. Furthermore, if one considers the typical errors of the employed OCR engine, and if the engine is likely to mistake "b" for "h" and "s" for "8", then the match between these two strings is considerably higher, depending on the probability of the occurrence of these errors. This problem is known as the approximate string matching problem. In order to solve the problem of OCR engine specific string matching, a modified version of the approximate string matching algorithm introduced by Ukkonen[10] was implemented in CLiDE. When a block containing OCR errors is successfully matched against an object description with a format string, besides labeling it, the system corrects the results of the OCR in the given text part. The same process is applied to the match of the other features. The tolerances of the match are parameters of the algorithm and can be changed along with the changes of the previous recognition phases.

Feature Matching. The algorithm employed for matching the extracted features against the described ones is a recursive parsing algorithm.[11] The parser processes the hierarchical tree structure of the physical page layout (list of columns, blocks, text lines, and words), extended with adjacency pointers at the block level. Other inputs to this process are the document description coded in DFDL and the recognized text of the page.



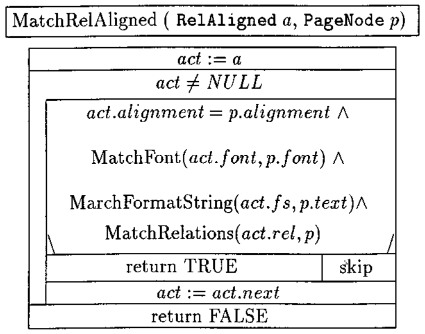
```
MatchAligned ( Aligned a, PageNode p)
act := a
while act ≠ NULL
    if act.alignment = p.alignment ∧
       MatchFont(act.font, p.font) ∧
       MarchFormatString(act.fs, p.text)
        return TRUE
    else
        skip
    act := act.next
return FALSE
```

Figure 4. Function for matching the "**Aligned**" type.



```
MatchFont ( FontStruct des, FontStruct source)
if source.size ∈ [lower_tol * des.size, upper_tol * des.size] ∧
   source.style = des.style ∧
   source.type = des.type
    return TRUE
else
    return FALSE
```

Figure 5. Function for matching the font attributes.



```
MatchRelAligned ( RelAligned a, PageNode p)
act := a
while act ≠ NULL
    if act.alignment = p.alignment ∧
       MatchFont(act.font, p.font) ∧
       MarchFormatString(act.fs, p.text) ∧
       MatchRelations(act.rel, p)
        return TRUE
    else
        skip
    act := act.next
return FALSE
```

Figure 6. Function for matching the "**RelAligned**" type.

As the blocks within the hierarchical tree structure of the page are ordered according to the reading order, the algorithm starts by matching the first block on the top of the page with the header description. The precondition to this match is that no other block exists above the matched one. According to the DFDL specification the header is a "**CondAligned**" type, which type comprises one or more conditional statements and "**Aligned**" block descriptions. Thus, the current block features are matched against the description code with the "MatchAligned" function (Figure 4), which checks the alignment, the font, and the formatted string attributes. The alignment check is simply matching the present block alignment to the allowed type of justification. The more complex font check consists of three parts: size, style, and type matching (Figure 5). The characteristic font size of the objects is given in pixels in the style description, but a global font size tolerance—upper and lower limit—is also required for the whole page, enabling handling of small size variations. The most complex part of the check is the formatted string matching.

In case of header matchings, if the function "MatchAligned" exits with the TRUE return value, then there is still one more criterion, namely the page condition, to be considered before classifying the block as header. If the header specification is given without any conditions, then the current page condition is left unchanged. Otherwise it is changed to the stronger conditional statement from the two, the current page condition and the one given in the

Table 1. Layout Object-Keywords of DFDL with the Corresponding Types[a]

object	type	object	type	object	type
abstract	b	formula_id	b	scheme_caption	c
affiliation	b	footnote	c∧d	subtitle	c
affiliation_comment	b	generic	c∧d	table	d
author_block	b	header	a+	table_caption	c
columns	d	page	d	table_footnote	c
datestamp	b	page_separator	d	tcolumn	a
experimental	c	paper	d	title_block	a
figure_caption	c	paper_title	b	trow	a
footer	a+	reference	b		
formula	b	scheme	c		

[a] Type: a = **Aligned**; a+ = **CondAligned**; type: b = **ReAligned**; type: c = **RelPosAligned**; type: d = some basic type(s) of "C" language (e.g., **int**, **char**, etc.).

Table 2. Some Statistical Results and Processing Time Requirements for the Page Layout Perception on a Sun 4/25 SPARC Station

	no. of blocks				proc. time (see CPU)	
image	present	lcass.	skipped	misclass.	feature extr.	perception
JACS, v.113, n.20, p.7461	22	9	3	0	0.03	0.04
JACS, v.113, n.20, p.7470	35	15	1	0	0.03	0.05
JACS, v.113, n.20, p.7613	34	15	0	0	0.02	0.03
JACS, v.113, n.20, p.7617	60	17	1	0	0.10	0.18
JOC, v.58, n.11, p.4681†	35	21	0	0	0.03	0.07
JOC, v.59, n.9, p.2277	42	15	2	3	0.05	0.02
JOC, v.59, n.9, p.2328	49	14	2	1	0.07	0.05

actual header specification. This operation is only valid if the two conditions are not contradictory, e.g., a change from "**even-page**" to "**odd-page**" is not allowed. In contradicting cases there is a classification error amongst the classified conditional objects of the page, thus backtracking is necessary.

Matching of the other two object types: the "**Relaligned**" (Figure 6) and the "**RelPosAligned**" is performed in a similar manner to the matching of the "**Aligned**" type. In these cases the algorithm handles no conditional statements, but in addition it checks alignment relationships supported by the alignment attributes and adjacency relationships supported by the adjacency pointers of the blocks. Object descriptions requiring sequential control structures, such as the title block containing, for example, the title, the author, the affiliation, and the abstract objects, respectively, are identified when all the constituent objects have been classified. If a part has been misclassified in this sequence, then backtracking may be applied.

5. RESULTS

The described layout detection method has been implemented in C++, compiled on a Sun 4/25 (21 MHz) SPARCstation with Sun OS, and also on an SG Indy (4600 PC, 110 MHz) running IRIX version 5.3.

A systematic test of the layout detection has been carried out on 40 full-page document images, originating from eight different chemistry journals. The test focused on the speed and accuracy of the algorithms which in practice performed well from both aspects. A high level of accuracy is indicated by a retrieval rate of 94.40%, reached an average classification speed of about 450 blocks/s of CPU time on the Sun. The same results were reached on the SG, but approximately five times faster, i.e., with a speed of 2000 blocks/s of CPU. Cardinality of the correctly labeled, skipped, and misclassified logical objects, along with the processing time requirements (on Sun) for some of the test images are given in Table 2. The second column of this table contains the number of all the blocks in the page found originally by the page segmentation routines. By the end of the segmentation, the blocks are classified into two groups: text and graphics. Further classification of the text is done during the logical layout detection phase. Thus the third, fourth, and fifth columns of the Table 2 state the number of the text blocks which were classified, skipped, and misclassified during this phase. The difference between their sum and the value in the second column gives the number of blocks left as simple text or graphic on the page. The layout detection results for the image marked with "†" are detailed in the next subsection.

Table 3. Error Analysis of the Systematic Test

journal	no. of images	no. of images labeled OK	source of the problems
BIACH	1	1	
JAFC	4	2	OCR err.
JMC	3	3	
JOC	10	3	segmentation and font det. error
LANG	2	1	
MACMO	4	2	segmentation error
ORGMET	2	2	
JACS	11	9	segmentation error
LOCS	3	3	
total	40	31	

Table 3 analyzes the layout recognition results for the full set of the test images. The 40 images, containing 1089 blocks, were identified with about 61 block identification errors, which gives an error rate of 5.60%. Among these errors, two categories can be distinguished: the skipped blocks (50 blocks = 4.59%) and the misclassified blocks (11 blocks = 1.01%). These error rates are fairly small, but the errors occurred on nine different images, which means that 23% of the images needed manual corrections. The errors mainly originated from the inherited and irrecoverable errors of the previous processing phases, such as bad OCR or bad segmentation.

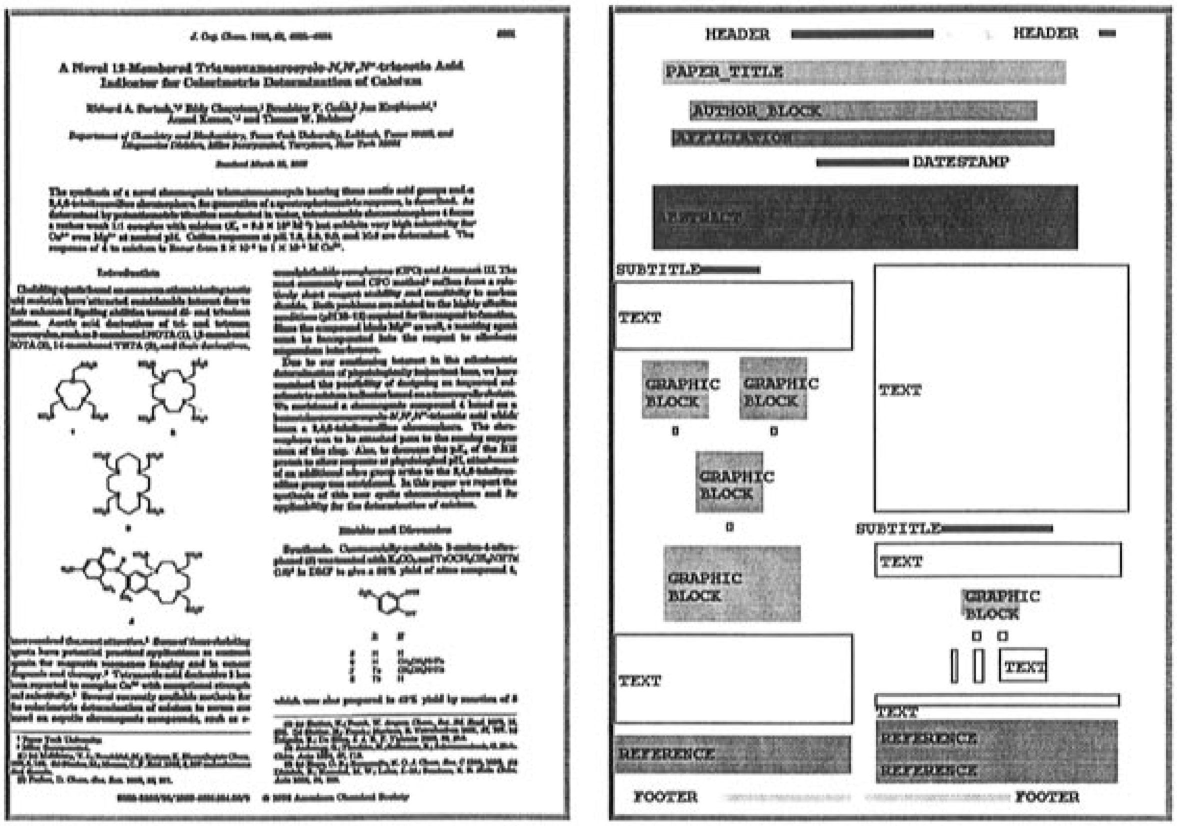


Figure 7. (a, left) A sample image and (b, right) the results of logical layout analysis (Source: JOC, vol. 58, no. 11, p 4681).

Table 4. Identified Variables and Conditions

id.	name	type	value
		Variables	
1.	PAGE_NUMBER	int	4681
2.	YEAR	int	1993
3.	VOL	int	58
4.	PAPER_FIRST_PAGE	int	4681
5.	PAPER_LAST_PAGE	int	4684
6.	DATE	string	"March 29, 1993"
7.	REF_NUM	int	1
8.	REF_TEXT	string	"(a) Stetter,H.; ..."
9.	REF_NUM	int	2
10.	REF_TEXT	string	"(a) McMurry,T.J.;..."
11.	REF_NUM	int	3
12.	REF_TEXT	string	"(a) Hope,D.B.;..."
		Conditions	
1.	PAGE_COND	statement	<Image is an Odd title page>

5.1. Sample Image: A Title Page. A JOC page, Figure 7a, was scanned at 300 dpi resolution and loaded into CLiDE. After the initial processing, which includes detection of the connected components, segmentation of the page, and recognition of character and graphic primitives, the next task was to extract the logical layout of the page. The results from the implemented layout perception method of CLiDE are shown in Figure 7b. In addition, Table 4 presents the list of the identified variables and the page condition. Manual verification of the results shows that the output is entirely correct according to the specification given in the style description file.

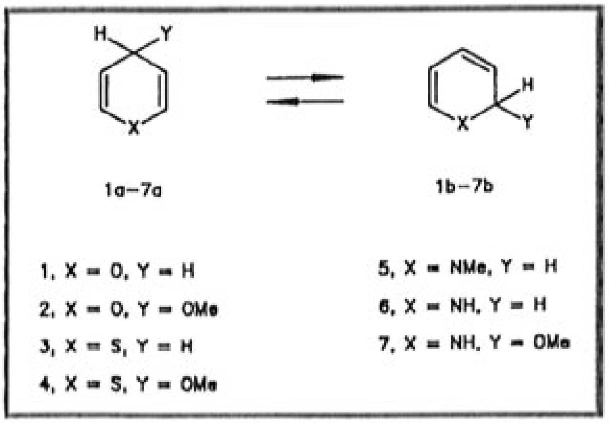
The statistical results also indicate a good recognition rate for the logical structure retrieval. The 35 blocks found by the initial page segmentation (five graphics and 30 text blocks) were all given a correct logical role in the structure retrieval phase: headers (2), paper_title, author_block, affiliation, date stamp, abstract, subtitles (2), references (3), footers (2), graphics (5), and unclassified text blocks (16). The JOC style description script used to achieve these results can be found in the appendix.

5.2. Sample Image: A Generic Reaction. In order to process an image containing generic structures (Figure 8a), a method for understanding generic notation has been implemented. The method is capable of identifying generic blocks (generic text descriptions and the influenced structure drawings) within a document image, of parsing their contents, and of displaying or saving the results, either in a compact generic form or in a listing of full structures. The three main parts of the implemented system are the generic block identification, the generic text parsing, and the influence range detection.

Generic block identification is one of the many tasks accomplished by the logical page layout detection algorithm. The process starts by interpreting the given document description file. On the basis of this description the algorithm detects the generic units present in the image. The identified generic units can be visualized in CLiDE, as shown in Figure 8b. The good recognition rates of this algorithm ensure that the effectiveness of the generic block identification is greater than 86%.

Once the generic blocks have been identified, the system starts the parsing phase. For parsing of the generic text, a context-free formal language was defined.[12] The parser raises error events in the case of unprocessable syntax, causing the text editor window to pop up and an error message to be displayed in the execution shell, explaining which character or sequence of characters forced the parser to stop.

After parsing, the influence ranges are verified on the basis of their content and context. A recursive search method,



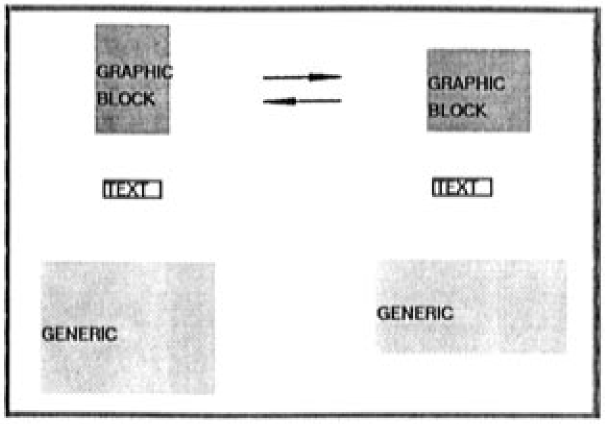


Figure 8. The original bitmap image of a generic reaction (a, left) and the found logical layout structure (b, right).

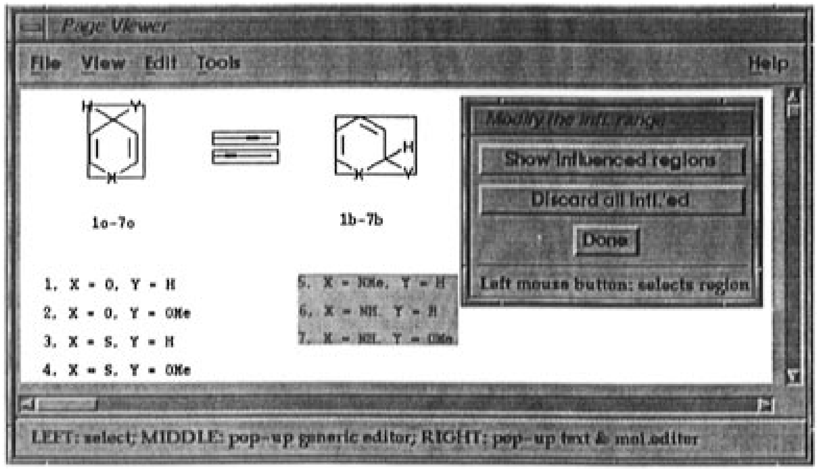

Page Viewer
File View Edit Tools Help
Modify the infl. range
Show influenced regions
Discard all infl.'ed
Done
Left mouse button: selects region
LEFT: select; MIDDLE: pop-up generic editor; RIGHT: pop-up text & mol.editor

Figure 9. The generic influence range editor.

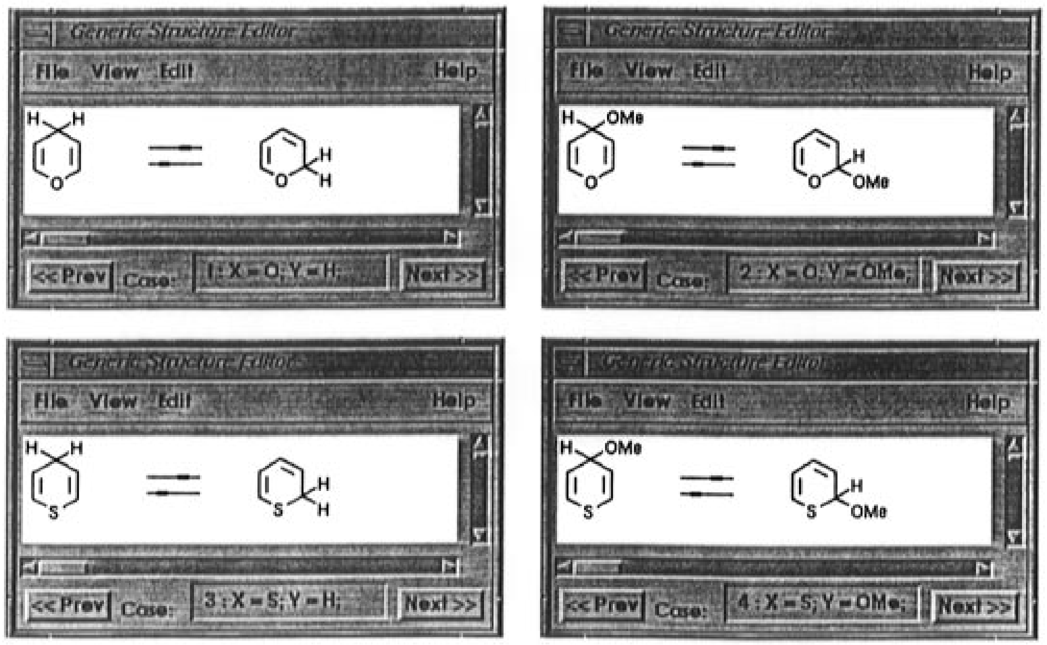

Generic Structure Editor
File View Edit Help
<<Prev Case: 1 : X = O; Y = H; Next>>
<<Prev Case: 2 : X = O; Y = OMe; Next>>
<<Prev Case: 3 : X = S; Y = H; Next>>
<<Prev Case: 4 : X = S; Y = OMe; Next>>

Figure 10. Visualization of the association between the first generic block and influenced structures.

which iteratively identifies the influenced structures is based on the extracted generic variables. The resulting influence ranges can be edited with the influence range editor shown in Figure 9. The available editing operations include both selection and unselection of influenced structures and arrows, which belong to a particular generic text block. The links between the generic variables in the structures and their values are generated on request whenever the user loads a generic object into the generic structure editor (Figure 10).

6. CONCLUSIONS

For the purpose of understanding chemical documents, a new logical layout detection method has been implemented that can extract regions containing, for example, biblio-

Chart 1

```
variables
14
<AUTHOR             string
<COPYRIGHT_NUM      "%d-%d/%d/%d-%d$%d.%d/%d"
<NO                 int
<PAGE_NUMBER        int
<PAPER_FIRST_PAGE   int
<PAPER_LAST_PAGE    int
<REF_TEXT           string
<REF_NO             string
<TITLE_ABR          string
<VOL                int
<YEAR               int
<GEN_ID             string
<GEN_VAL            string
<DATE               string

descr_body
Header   if ( Even_title_p  )
            ( 0 italic NULL centered "J.\tOrg.\tChem,"<YEAR","
                   <VOL","<PAPER_FIRST_PAGE"-"<PAPER_LAST_PAGE ) ;
            ( 30 bold NULL flush_left <PAGE_NUMBER )
         else if ( Odd_title_p  )
            ( 0 italic NULL centered "J.\tOrg.\tChem,"<YEAR","
                   <VOL","<PAPER_FIRST_PAGE"-"<PAPER_LAST_PAGE ) ;
            ( 30 bold NULL flush_right <PAGE_NUMBER )
         else if ( Even_p )
            ( 0 italic NULL flush_left <PAGE_NUMBER"J.\tAm.\tChem.
                   \tSoc.,\tVol."<VOL",\tNo."<NO","<YEAR ) ;
            ( 0 italic NULL flush_right <AUTHOR"et\tal." )
         else if ( Odd_p )
            ( 0 NULL NULL flush_left <TITLE_ABR ) ;
            ( 0 italic NULL flush_right "J.\tOrg.\tChem.,\tVol."
                   <VOL",No."<NO","<YEAR<PAGE_NUMBER )
Footer   if ( Title_p )
            ( 0 NULL NULL centered <COPYRIGHT_NUM"o"
                   <YEAR"American\tChemical\tSociety" )
Page_separator ( 980 )
Footnote    ( 27 NULL NULL justified under_of 2 { Reference |
                                          Affiliation_comment } )
Reference   ( 27 NULL NULL justified column "("<REF_NO")"<REF_TEXT )
Title_block Paper_title ; Author_block ; Affiliation ;
                                            Datestamp ; Abstract
Paper_title ( 41 bold NULL centered page NULL )
Author_block( 30 NULL NULL centered page NULL )
Affiliation ( 32 italic NULL centered page NULL )
Datestamp   ( 28 italic NULL centered page "Received"<DATE )
Abstract    ( 0 NULL NULL centered page NULL )
Subtitle    ( 0 bold NULL centered first_in_block column NULL )
Generic     ( 21 NULL NULL 4 { justified | centered |
                             flush_right | flush_left }
                       2 { under_of | to_the_left_of }
                       <GEN_ID"="<GEN_VAL )
```

graphical items. The layout style of the targeted documents, i.e., the chemical literature, is interpreted as being publication specific, and, therefore, a language has been designed to represent the various layout structures found in the current chemical literature. One of the many tasks accomplished by the logical page layout detection algorithm is the identification of the generic blocks. This task is an important component of the interpretation of the generic structures found in document images. Through experimental investigation it has been found that the proposed method for analyzing documents, based on DFDL, is fast and accurate.

The code written in DFDL, called the style file, along with the document image represents a vital input for the CLiDE system. The style files are produced manually at present. Development of a semiautomatic interactive environment for generating these files is planned.

A general problem of document processing, especially in the case of chemical documents, is to define what information is necessary and satisfactory for other applications. In other words, changes in the reader perception of the literature could result in changes of the logical objects. Although DFDL is capable to cope with some of these changes (new objects can be easily defined), structural changes or type changes would require a redesign of the language.

Notwithstanding these problems this method is significant for two reasons: DFDL covers the most of the presently known chemical document layouts, and the style files, written in this language, describe the layout rules in a clear form, separately from the algorithms. The separation of the style files from the algorithms has the benefit of being adaptable to new layouts without having to change the program.

APPENDIX

A style description script for the Journal of Organic Chemistry (JOC) is shown in Chart 1.

REFERENCES AND NOTES

(1) Ibison, P.; Jacquot, M.; Kam, F.; Neville, A. G.; Simpson, R. W.; Tonnelier, C.; Vinczel, T.; Johnson, A. P. Chemical literature data extraction: The CLiDE project. *J. Chem. Inf. Comput. Sci.* **1993**, *33*(3), 338−344.

(2) Ibison, P.; Kam, F.; Simpson, R. W.; Tonnelier, C.; Venczel, T.; Johnson, A. P. Chemical structure recognition and generic text interpretation in the CLiDE project. In *Proceedings on Online Information 92*; London, England, 1992.

(3) Simon, A.; Pret, J. C.; Johnson, A. P. (Chem)DeTeX automatic generation of a markup language description of (Chemical) documents from bitmap images. In *Proc. 3rd Int. Conf. on Doc. Anal. and Recogn. (ICDAR'95)*; Montreal, Canada, 1995; pp 458−461.

(4) Bayer, T. A.; Walischewski, H. Experiments on extracting structural information from paper documents using syntactic pattern analysis. In *Proc. 3rd Int. Conf. on Doc. Anal. and Recogn. (ICDAR'95)*; Montreal, Canada, 1995; pp 476−479.

(5) Niyogi, S.; Srihari, S. N. Knowledge-based derivation of document logical structure. In *Proc. 3rd Int. Conf. on Doc. Anal. and Recogn. (ICDAR'95)*; Montreal, Canada, 1995; pp 472−475.

(6) Summers, K. Near-wordless document structure classification. In *Proc. 3rd Int. Conf. on Doc. Anal. and Recogn. (ICDAR'95)*; Montreal, Canada, 1995; pp 462−465.

(7) Higashino, J.; Fujisawa, H.; Nakano, Y.; Ejiri, M. A knowledge-based segmentation method for document understanding. In *Proceedings on 8th International Conference on Pattern Recognition*; 1986; pp 745−748.

(8) Tang, Y. Y.; Yan, C. D.; Suen, C. Y. Document processing for automatic knowledge acquisition. *IEEE Trans. Knowledge Data Eng.* **1994**, *6*(1), 3−21.

(9) ISO 8613. Information processing - text and office systems - Office Document Architecture (ODA) and interchange format. *International Organization for Standardization*; 1989.

(10) Ukkonen, E. Algorithms for approximate string matching. *Inf. Control* **1985**, *64*, 100−118.

(11) Aho, A. V.; Hopcroft, J. E.; Ullman, J. D. *Data Structures and Algorithms*; Addison-Wesley Publishing Company: Reading, MA, 1983.

(12) Simon, A.; Johnson, A. P. The interpretation of generic structure diagrams in chemical document images. Submitted to *J. Chem. Inf. Comput. Sci.*

CI9601022