Logical Structure Analysis and Generation for Structured Documents: A Syntactic Approach

Kyong-Ho Lee, Member, IEEE Computer Society, Yoon-Chul Choy, Member, IEEE Computer Society, and Sung-Bae Cho, Member, IEEE

Abstract—This paper presents a syntactic method for sophisticated logical structure analysis that transforms document images with multiple pages and hierarchical structure into an electronic document based on SGML/XML. To produce a logical structure more accurately and quickly than previous works of which the basic units are text lines, the proposed parsing method takes text regions with hierarchical structure as input. Furthermore, we define a document model that is able to describe geometric characteristics and logical structure information of documents efficiently and present its automated creation method. Experimental results with 372 images scanned from the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) show that the method has performed logical structure analysis successfully and generated a document model automatically. Particularly, the method generates SGML/XML documents as the result of structural analysis, so that it enhances the reusability of documents and independence of platform.

Index Terms—Logical structure analysis, document image understanding, structured documents, SGML, XML, a syntactic method.

1 INTRODUCTION

With the advance of high-performance computers and the wide spread use of high-speed networks, the transformation of a paper-based document into its electronic version is rapidly increasing in demand. The use of electronic documents increases largely due to the prevalence of the Digital Library, Internet, and EC (Electronic Commerce).

Meanwhile, the volume of paper-based documents still continues to grow in spite of the wide dissemination of electronic documents because of the human preference for paper-based documents. However, paper-based documents are less efficient than electronic documents from the various perspectives of document processing such as storage, retrieval, modification, and transfer. Therefore, automated transformation of a paper-based document into its electronic version is very important.

Document image analysis and understanding [1], [2] for transforming a document image into its electronic version consists of two phases: geometric structure analysis and logical structure analysis. Compared to geometric structure analysis, there is not so much literature for logical structure analysis. However, there is a growing interest in logical structure analysis because of the wide spread use of structured documents.

Since SGML (Standard Generalized Markup Language) [3] and XML (eXtensible Markup Language) [4] have advantages of embedding logical structure information into documents and are independent of a platform, they are widely used as standard formats for electronic documents in various fields such as Digital Library, CALS (Commerce At Light Speed), EC, and the Web.

Generally, humans identify logical components such as section headers or paragraphs from document images using geometric characteristics of the corresponding text areas. Additionally, by combining them into hierarchical structure elements such as sections, they recognize a logical structure of a document. Likewise, the logical components that are directly identifiable from geometric characteristics of the corresponding text regions are called primary structures and the ones that can be identified through grouping components together are called secondary structures [5].

For automated creation of SGML/XML documents from document images, secondary structures as well as primary structures should be extracted from multipage documents. For example, a paragraph can be placed in adjacent pages. For identifying a section, extraction of lower-level components such as a section title, paragraphs, and subsections must be preceded. However, because most previous works [6], [7], [8], [9], [10] only extract primary structures from single-page documents, they cannot create hierarchical structure information. Therefore, extending the work from single-page documents to multipage documents is required.

Generally, the logical structure of a document differs according to the document type and the geometric layout of documents with the same type may differ according to the publication. Therefore, the development of a logical structure analysis method for all document types is impossible. For this reason, logical structure analysis methods need a document model that represents explicit knowledge about the document class and publication.

For effective extraction of a logical structure from document images, a document model should be able to express various information about geometric characteristics and logical structure information. As a reasonable way to customize a system for documents with different structures,
automated creation of a document model from samples is also needed. However, most of the conventional methods provide only a simple document model and they do not create a document model automatically [6], [7], [10].

Therefore, we present a syntactic method for logical structure analysis of multipage documents with hierarchical structure. Specifically, our work concentrates on logical structure analysis of technical journal documents. Our choice is dictated by the availability of a large number of documents of this type and their explicit logical hierarchy. Normally, text regions comprise a document image function as a header or a body. For instance, a title and a paragraph correspond to a header and a body, respectively. Headers and bodies may be classified into various kinds according to their geometric characteristics. In this paper, we define headers and bodies as functional components, and a hierarchy of headers and bodies as a functional structure tree.

To improve the processing speed of structure analysis, compared with previous works of which the basic units are text lines, the proposed method takes a functional structure tree and creates a logical structure tree with labels on each node by applying a document model. For this purpose, the method merges adjacent text lines that have similar geometric characteristics into a sequence of headers and bodies, and creates a functional structure tree in a top-down manner by splitting the sequence repeatedly based on the repeating header.

This paper also presents a Document Structure Description Language (DSDL) for representing a document model efficiently. DSDL is designed to formulate not only geometric characteristics of primary structures but also structure information of secondary structures. It is also designed to incorporate a new class of logical components. In addition, the proposed creation method of a document model generates an SGML DTD (Document Type Definition) and a DSSSL (Document Style Semantics and Specification Language) [14] style sheet from logical structure information and geometric characteristics of the created document model, respectively.

Experimental results with 26 regular papers with 372 pages from the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) show that the method has performed logical structure analysis on multipage documents successfully and generated a document model automatically. Particularly, the proposed method generates SGML/XML documents as the result of structural analysis, so that it enhances the reusability of documents and independence of platform. As a result, this work is considered a valuable contribution to various fields such as Digital Library and office automation where transformation of large volumes of paper documents into electronic representations is needed.

This paper is organized as follows: In Section 2, a thorough and comprehensive survey on related works is presented. DSDL for describing a document model is explained in detail in Section 3. In Section 4, the proposed method for logical structure analysis is classified into three stages: creation of a functional structure tree, creation of a logical structure tree, and generation of SGML/XML documents. A detailed explanation on each stage is given. The creation method of a document model is introduced in Section 5. In Section 6, the performance of the method is analyzed and compared to previous works through experimental results on a large volume of documents. Finally, the conclusions and future works are summarized in Section 7.

# 2 Related Works

Previous works for logical structure analysis can be broadly classified into the model-matching method based on AI and the syntactic method, and they generally use a document model or a grammar. This section, as shown in Table 1 and Table 2, classifies the conventional methods into the model-matching and syntactic methods, and describes their characteristics and problems in terms of extractable structure types, representation range of a document model, and automated creation of a document model.

For syntactic methods, Nagy et al. [6] and Krishnamoorthy et al. [7] segment and label a page image simultaneously using publication-specific knowledge in the form of page grammar. Because they only extract primary structures using simple geometric characteristics such as the length and frequency of neighboring pixels, sophisticated structure analysis is not allowed. Likewise,
Story et al. [15] define a simple grammar, which uses information about the relative positions of blocks. Hu and Ingold [16] present a syntactic method based on fuzzy logic, which allows error-tolerant parsing. When a precise parse is impossible, the method chooses the nearest match based on a predefined cost function. Conway [17] proposes a bottom-up parsing method based on a page grammar. The grammar describes spatial relations between blocks, their frequency, and geometric information such as font size and style, alignment, and indentation.

Tateisi and Itoh [18] assign one or more possible labels to a text line and form a graph structure that represents the connections between the labels. Their parsing method searches the graph path with the lowest cost by applying a grammar. The grammar includes rules with statistical values that reflect their probability according to the geometric characteristics and the order of adjacent labels.

Klein and Fankhauser [19] and Klein and Abecker [20] have developed DREAM [21], which transforms electronic documents to SGML versions. Particularly, DSD (Document Structure Description) based on an SGML DTD is defined for a transformation grammar. To identify the beginning and ending regions for a certain element as well as hierarchical structure of elements, DSD uses regular expressions to describe style information such as characters, symbols, indentation, and spacing. Furthermore, the method supports the automated creation of a grammar [22]. However, the method targets character-based electronic documents or OCR-processed documents.

On the other hand, the conventional methods based on model matching are as follows: Farrow et al. [10] and Niyogi and Srihari [9] use knowledge about the publication-specific layout of technical journal documents and newspapers, respectively. Because there is no way for a document model to describe the relations between logical components and their frequency, structured documents with hierarchy cannot be dealt with.

Dengel and Barth [12] and Dengel et al. [13] propose a geometric tree as a document model that can describe geometric characteristics and logical hierarchy of a document class. Additionally, they propose methods [23], [24] for creating a geometric tree automatically from samples. However, they do not support a document with multiple pages because they use the relative size and location of blocks to an entire page as geometric characteristics. Tsujimoto and Asada [11] define logical structure analysis as a transformation from a geometric structure tree into a logical structure tree. For this transformation, four rules are proposed. The method does not support documents with multiple pages and hierarchical structures.

Rus and Summers [25] and Summers [26] create a hierarchical tree structure in a bottom-up manner by grouping adjacent text lines together. The method takes geometric characteristics including contour similarity, vertical distance, and horizontal containment into consideration, and merges adjacent regions to create their top-level structure. Then, labeling proceeds based on the comparison of structural
nodes to predefined prototypes that describe values of attributes such as shape, context, height, symbol, font, and children. However, the method does not provide a document model that can explicitly describe structure information of a document class.

Hitz et al. [27] define generalized N-Gram, which is a statistical document model. The generalized N-Gram model represents hierarchical relationships between logical components with probabilities. In addition to an automated creation of a document model, the formalisms used to express the model into an SGML DTD and a DSSSL style sheet are proposed [28]. However, the method uses a heuristic search algorithm for choosing the most appropriate model for a document image. Thus, as the size of the document model and its structural complexity increase, the time complexity may be exponential.

Lin et al. [29] perform a logical structure analysis of a book by making use of its table of contents. The characters of headline candidates discriminated from text page analysis are matched with the headlines and page numbers drawn from the content page and, thus, a hierarchical structure is constructed by using chapter-numbers in headlines. Kochi and Saito [30] identify logical components by comparing geometric characteristics of text blocks to a document model. The proposed method cannot extract logical hierarchy because the document model expresses only geometric characteristics.

The document model proposed by Bayer and Walischewski [31] expresses geometric characteristics, spatial relations, and lexical information. They extract only the logical components that correspond to primary structures from business letters. Worring and Smeulders [32] identify logical objects based on geometric characteristics in the form of predicate logic, and a hierarchical structure is extracted by identifying the titles of chapters or sections using OCR.

Most of the previous works usually extract simple structure information from a single-page image. Therefore, a sophisticated method for documents with multiple pages and hierarchical structure is required. For this purpose, a document model should be able to express not only geometric characteristics but also logical structure information such as the ordering and spatial relationships among logical components, their frequency, and so on.

## 3 Document Model

This section describes DSDL that is proposed to define a document model in detail. For secondary structures that correspond to interior nodes in the proposed logical structure tree, DSDL expresses such information as their names and frequency, and ordering information among them in the form of regular expression. This paper defines this expression as the content model of a secondary structure. The content model defines the required and optional contents of secondary structures, specified by subelement names as shown in Fig. 1. Additionally, it defines order among the contents and their occurrence.

As geometric conditions that primary structures should satisfy, the model describes geometric characteristics such as column type, the number and height of text lines, the space before and after a text region, the density distribution

```
Begin Document
  Title, Author, Affili, Abstract, Keyword, Sec-Bodies
End Document

Begin Author
  (Title, Author, Affili, Abstract, Keyword, Sec-Bodies)
End Author

Begin Affili
  (Title, Author, Affili, Abstract, Keyword, Sec-Bodies)
End Affili

Begin Abstract
  (Title, Author, Affili, Abstract, Keyword, Sec-Bodies)
End Abstract

Begin Keyword
  (Title, Author, Affili, Abstract, Keyword, Sec-Bodies)
End Keyword

Begin Sec-Bodies
  (Section*, Reference*)
End Sec-Bodies

Begin Section
  (Sec-Header, Paragraph*, Sub-Section*)
End Section

Begin Sub-Section
  (Sec-Header, Paragraph*, Sub-Sub-Section*)
End Sub-Section

Begin Paragraph
  (Sec-Header, Paragraph*)
End Paragraph

Begin Sec-Header
  (Sec-Header, Ref-Item*)
End Sec-Header

Begin Sub-Sub-Header
  (Sec-Header, Reference*)
End Sub-Sub-Header

Begin Reference
  (Sec-Header, Ref-Item*)
End Reference
```

Fig. 1. An example of a document grammar.
of a black pixel, and alignment. DSDL discriminates geometric characteristics of a primary structure from the content model of a secondary structure by using the symbol "#" as shown in Fig. 1.

Particularly, DSDL has been defined by extending the element declaration method of an SGML/XML DTD, which describes a logical structure of documents, as shown in Fig. 2. We define a document model that is described by DSDL as a document grammar.

Like SGML/XML, DSDL allows only deterministic content model [33]. For example, the content model \((b; c) | (b; d)\) is nondeterministic because, with the given initial \(b\), the parser cannot know which \(b\) in the model is being matched without looking ahead to see the element following the \(b\). In this case, the two references to \(b\) can be collapsed into a single reference, changing the model into a deterministic one \((b, (c | d))\) by using the conventional method [34]. The initial \(b\) now clearly matches only a single name in the content model.

4 THE SYNTACTIC METHOD

In this section, the method proposed for logical structure analysis is described. As shown in Fig. 3, the method is composed of three phases: creation of a functional structure tree, logical structure tree creation, and generation of an SGML/XML document.

First is the creation step of a functional structure tree where text lines are merged into a sequence of headers and bodies and, by splitting the sequence repeatedly, a hierarchical functional structure tree is created. Second, the parsing method applies a document grammar on a functional structure tree and generates a logical structure tree. To improve the processing speed of structure analysis, the parsing method takes a functional structure tree as input rather than pixels or text lines. Finally, the method generates an SGML/XML document as the result of logical structure analysis by doing depth-first search on a logical structure tree. A detailed explanation on each stage is as follows.
4.1 Functional Structure Tree Creation

In this section, a method for creating a functional structure tree from text lines is described. First of all, a sequence of headers and bodies is created by merging adjacent text lines whose geometric characteristics are homogeneous.

Three principles from Gestalt psychology [35] are applied. First, according to the principle of proximity, the line spacing between text objects is larger than the one between text lines within the same object. Second, geometric characteristics of text lines belonging to the same object are similar according to the principle of similarity. Last, text lines belonging to different kinds of column areas are easily discriminated by the principle of contiguity. For example, in the TPAMI that has been used for experiments, whereas text lines within a single-column area comprise a title, an author, an abstract, and keywords, double-column areas include text lines that comprise sections with hierarchically nested structures.

There are different types of headers and bodies, which can be differentiated from each other by the geometric characteristics. For example, a title and a section title are considered to be different types of headers and an abstract, and keywords, double-column areas include text lines that comprise sections with hierarchically nested structures.

The method creates a functional structure tree from the sequence of headers and bodies based on the repetitive characteristic of a header in a document. Typically, each section structure is identified by the section title and the title of a subsection is placed after the title of the section that it is contained in.

Therefore, the proposed method creates a functional structure tree in a top-down manner by splitting a sequence repeatedly with the repetitive header as the base. Examples of a sequence of headers and bodies extracted from the section body of a sample paper of TPAMI and its partial functional structure tree are shown in Fig. 4 and Fig. 5, respectively.

As shown in Fig. 6, the method segments a sequence based on the first identified repetitive header.

This splitting process extends a tree structure in a top-down manner. For example, the segmenting result of the sequence S with the repetitive header H1 as the base, and
the corresponding tree structure are shown in Fig. 7 and
Fig. 8, respectively.
On the other hand, if the sequence contains a single
header that corresponds to the title of a lower level section,
it is split using this as the base. Fig. 9 and Fig. 10 are the
results of segmenting $S_1$ and $S_{1-3}$ with $H_2$ and $H_3$ as the
base, respectively. This process is repeatedly applied until
there is no more divisible sequence left. Therefore, the

1. Generate a header/body sequence.
   (a) Group together text lines into various kinds of header and body based on the
   principles of Gestalt psychology.
2. Split the header/body sequence into sub sequences.
   (a) Find repetitive headers that form repetitive patterns, or the header
   which corresponds to a lower level header.
   (b) Split the sequence into subsequences at the next lower level:
   the sequence becomes the parent of subsequences.
3. Repeat step 2 until no changes are generated.
proposed algorithm can extract hierarchical structure from a document composed of multilevel nested components. In particular, each leaf node comprising the created functional structure tree includes the geometric characteristics of the corresponding text region.

4.2 Logical Structure Tree Creation

The proposed parsing method creates a logical structure tree by applying a document grammar to a functional structure tree. As shown in Fig. 2, a document grammar contains the type, the order, and the frequency of its contents for each secondary structure as well as the geometric characteristics of each primary structure. By doing depth-first search on a functional structure tree, the method tests whether the hierarchical structure of interior nodes and the geometric characteristics of leaf nodes are acceptable under the grammars of a document model.

The method responds appropriately to interior nodes including a root node according to whether the child of the node in question is a leaf node or an interior node. If the child node is a leaf node, an element name that satisfies its geometric characteristics is given as a label. Otherwise, one of the permissible element names is given and whether the corresponding content model of the element is appropriate or not is tested recursively.

If the child node is found invalid (i.e., it cannot be parsed with the assigned grammars), the proposed method backtracks to an alternative grammar for the parent node. For example, suppose that the content model of Sec-Body in Fig. 1 is applied to the child node $S_9$ of the interior node $S$ in Fig. 5. As it is possible to give Section or Reference as a label, the proposed method grants the label Section, first, then tests whether the child node of $S_9$ satisfies the content model of Section. However, the content model in question does not contain the element that satisfies the geometric characteristics of the second leaf child node, $B_2$. Therefore, the proposed method backtracks to the parent node, $S_9$, to apply the content model of Reference that is another permissible grammar.

To improve processing speed for an interior node, the proposed method reorganizes the content model in the table. First, the method converts the content model that has the connector "&," which means that the order does not matter, into the content model with explicit order. For example, a content model $(A & B)$ is transformed to $((A B)(B A))$.

The proposed method expresses content models in transition diagrams, which are later reconstructed as transition tables. Each logical component contains a starting pointer for the corresponding table. For example, the result of constructing the content model, $((A C)(B D E))$, in transition diagram and the corresponding implementation are shown in Fig. 11.

As a result, the proposed parsing method completes a logical structure tree by assigning a label to each node of a functional structure tree. For example, based on the document grammar in Fig. 1, the logical structure tree created from the functional structure tree in Fig. 5 is as shown in Fig. 12.

4.3 SGML Document Generation

This paper generates an SGML/XML document as the final result of logical structure analysis. Generally, the reading order of users from the beginning to the end of a document corresponds to the process of doing a depth-first search in a logical structure tree.

The proposed method applies different methods on leaf and interior nodes of a logical structure tree to generate an SGML/XML document. The main principle is to traverse the logical structure tree in a depth-first manner. Whenever an interior node is encountered and exited, its corresponding start and end tags are added directly to the result file.

For example, the start and end tags (i.e., <Section> and </Section>) of an element Section have to be created on node entry, 1, and node exit, 10, in Fig. 13. A leaf node has a

\[
\begin{align*}
S_1 & = \{H_1, B_1, B_1, B_1, B_1, B_1, S_1-1, S_1-2, S_1-3, S_1-4\} \\
S_1-1 & = \{H_2, B_1, B_1, B_1, B_1\}, \\
S_1-2 & = \{H_2, B_1, B_1, B_1, H_3, B_1, B_1, B_1, H_3, B_1, B_1\}, \\
S_1-3 & = \{H_2, B_1, B_1, H_3, B_1, B_1, B_1, H_3, B_1, B_1, H_3, B_1, B_1\}, \\
S_1-4 & = \{H_2, B_1, B_1, B_1\}
\end{align*}
\]

Fig. 8. Tree representation of Fig. 7.

Fig. 9. The splitting result of $S_1$ with $H_2$ as the base.

Fig. 9. The splitting result of $S_1$ with $H_2$ as the base.

Fig. 10. The splitting result of $S_1-3$ with $H_3$ as the base.
direct relation with a text region. Whenever a leaf node is encountered, its corresponding start tag is added and then the OCR result of the text region and the end tag are added. Fig. 14 is an SGML document, which is generated from the logical structure tree in Fig. 13.

Figure objects such as images and drawings are not considered as logical units of a document. They are just physical entities. Actually, as illustrated in Fig. 15, they must be defined in the corresponding DTD before the first reference to them and can be referenced in documents. In the example, the “ENTITY” keyword identifies an entity declaration [36]. The file “sample.gif” is declared as an entity and identified as a “GIF” file in the DTD. In an XML document, the entity is referenced by an attribute SRC, which has already been specified to be of type “Entity.”

On the other hand, in SGML/XML documents, a table is usually represented by elements with hierarchical structure. This paper does not deal with transformation of tables into structured representation.

In addition, it is possible to validate an SGML/XML document on an SGML/XML DTD that has been automatically generated from a document model. The method to generate an SGML/XML DTD is explained later in Section 5.3.

5 AUTOMATED CREATION OF DOCUMENT MODEL

This section describes the creation method of a document model from sample images. The proposed method is composed of four stages: creation and generalization of a functional structure tree, labeling, and generation of a document model. Additionally, to support SGML and
DSSSL, the method generates an SGML DTD and a DSSSL style sheet from logical structure information and geometric characteristics contained in a document model, respectively. Detailed explanation on each stage except the creation step of a functional structure tree, which has already been described in Section 4.1, is as follows.

### 5.1 Generalization of Functional Structure Tree

This method transforms a functional structure tree illustrated in Fig. 5 into its generalized version as shown in Fig. 16. The main principle lies in the repetitive characteristic of structures in a document.

For a technical journal document, a section body is normally composed of multiple sections with a hierarchically nested structure. For each interior node of a functional structure tree, the proposed method identifies repetitive common structures from the sequence of its child nodes and introduces Kleene stars “∗” to produce a regular expression that generalizes the sequence.

Section structures with the same level are located in the same level of a functional structure tree. In the example of Fig. 5, sections are located in the second level. The proposed method applies the bottom-up generalization to each level.
of a functional structure tree and extracts a regular expression of substructure that a higher node in the tree can contain. The generalization algorithm is as shown in Fig. 17.

For example, the generalization result of the lowest level in Fig. 4 and Fig. 5 is as shown in Fig. 18. In the paper, the result of representing a group of repetitive nodes with Kleene star is defined as a generalized pattern. For example, by replacing \((B_1, B_1, B_1)\) with \((B_1^*)\) from \(S_1 - 2 - 1 = (H_3, B_1, B_1, B_1)\), a generalized pattern \((H_3, B_1^*)\) is created. Actually, a generalized pattern corresponds to a regular expression for the type, order, and frequency of children that a parent node can contain. To extract a generalized pattern effectively, the method expresses parent nodes that have the same generalized pattern using an auxiliary symbol with the same name. Especially, two patterns, \((a, b)\) and \((a, b^*)\), which start with the same header and body, are considered as the same form.

As a result, both of the generalized patterns, \((H_3, B_1^*)\) and \((H_3, B_1)\) are replaced with an auxiliary symbol, A1, as shown in Fig. 19. Afterwards, repetitive patterns, \((B_1 \ldots B_1)\) and \((A1 \ldots A1)\), are identified from node sequences that are part of level 4 and they are replaced with \((B_1^*)\) and \((A1^*)\), respectively, to extract generalized patterns as shown in Fig. 20.

Similarly, level 3 is generalized from the result of level 4. For example, the generalization process of

\[
S_1 = (H_1, B_1, B_1, B_1, B_1, B_1, A2, A3, A3, A2)
\]

in Fig. 21 is as follows: As mentioned before, the method starts to identify repetitive patterns \((B_1 \ldots B_1)\) and \((A3 \ldots A3)\) and then generates a regular expression, \(S_1 = (H_1, B_1^*, A2, A3^*, A2)\), as shown in Fig. 22.

Sections in a document may include multilevel subsections. Therefore, the two nodes A and B, which have \((S, C)\) and \((S, C)\), respectively, as patterns of children, are generalized as \((B^*)\). Here, \(S\) is a single header or a sequence composed of a header and one or more body of the same type, and \(C\) is an auxiliary symbol that represents a subsection. For example, \(A2\) and \(A3\) are generalized patterns of which both contain a prefix \(A2\). The regular expression \((A2, A3, A2)\) is further generalized into \((A3^*)\) to extract \((H_1, B_1^*, A3^*)\) as the generalized pattern of \(S_1\) as shown in Fig. 23. The generalization result of a functional structure tree in Fig. 19 to level 3 is as shown in Fig. 24.

Finally, the generalization process for the children, \((S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9)\), of node \(S\) in Fig. 24 is as follows: Generalized patterns,

\[
S_1 = S_2 = S_3 = S_5 = (H_1, B_1^*, A3^*)
\]

\[
S_4 = S_6 = S_7 = (H_1, B_1^*)
\]

and \(S_8 = (H_1, B_1^*)\), and \(S_9 = (H_1, B_2^*)\), are replaced with auxiliary symbols \(A4, A5, A6\), and \(A6\), respectively. Afterwards, repetitive patterns \((A4, A4, A4)\) and \((A5, A5, A5)\) are identified and replaced with \((A4^*)\) and \((A5^*)\), respectively. As a result, generalized pattern \((A4^*, A5^*, A4^*, A5^*, A4^*, A6)\) is created. By applying 1(b) of the generalization algorithm, \((A4^* A6)\) is created as the final pattern as shown in Fig. 16.
5.2 Labeling

To give logical meanings to nodes of a generalized tree, the proposed method receives an appropriate label for each node from the user. The labeling is the one and only one manual process in the document model creation. This is because various kinds of names can be used to represent a logical component, for example, `section-title`, `Sec-Title`, or `SectionTitle`. As a result, the document model creation is not fully but mostly automated. For example, the result of labeling the generalized tree of Fig. 16 is as shown in Fig. 25.

5.3 Document Model Generation

As mentioned before, interior nodes and leaf nodes in a functional structure tree correspond to secondary structures and primary structures, respectively. Therefore, for each interior node, the proposed method creates its content model by using the corresponding generalized pattern for its children. Likewise, the geometric conditions of primary structures are generated from leaf nodes. Fig. 26 illustrates a partial document model extracted by applying breadth-first traversal on the labeled generalized tree in Fig. 25.

Generally, a document model is learned from a multitude of sample data. For this purpose, the proposed method merges possible content models of each secondary structure. Specifically, the factoring process is introduced to extract a more simplified form of a regular expression. For example, for content models `(ac)`, `(ad)`, `(bc)`, and `(bd)`, the factoring step would generate the factored form `((a|b)|((c|d)))`. This paper factorizes candidate expressions by using the algorithm proposed by Wang [37] from the literature of logic optimization. Additionally, the proposed method can reflect new documents, which belong to the same document class because it can easily merge the generalized result of the new document with an existing grammar.

Most applications dealing with SGML/XML documents use DTDs and style sheets that describe the logical schema.
and typographic style of documents, respectively. The proposed document model maintains not only geometric characteristics but also logical schema. For interfacing with the conventional document processing applications, the method generates an SGML DTD and a DSSSL style sheet from a document model.

The method generates an SGML DTD by applying a breadth-first search to the labeled generalized functional structure tree. Furthermore, a DSSSL style sheet is generated from geometric characteristics of leaf nodes. For example, the DTD and style sheet generated from the tree in Fig. 25 are as shown in Fig. 27 and Fig. 28, respectively.

6 EXPERIMENTAL RESULTS

For a fair evaluation, it is desirable to use standard databases of document images or the same data with previous works. However, the conventional databases do not contain many complete multipage documents and there is a lack of literature that has presented experiments on logical structure analysis. To evaluate the performance of the proposed method, we have experimented with 26 regular papers of TPAMI from January to June of 1999, which consist of a total of 372 page images. The method takes a set of text lines comprising each multipage paper as input.

For the identification of header and body, in advance, we have applied our recent work for geometric structure analysis [38] on each paper to extract a set of text lines and prepared test data set with ground truth of headers and bodies that are created from the extracted text lines by human experts. Table 3 shows the result of quantitative analysis for the performance of the proposed method.
6.1 Performance Analysis

This section describes the performance evaluation of the proposed method in terms of the accuracy and the processing speed. The proposed method for structure analysis consists of two main stages: creation of a functional structure tree and its parsing. The creation of a functional structure tree is also composed of two stages: identification of a header and a body and creation of hierarchical structure. Specifically, for the evaluation of the accuracy, three measures are proposed: the identification rate of headers and bodies, the accuracy of hierarchy, and the identification rate of logical objects.

In the case of the identification rate of headers and bodies, Table 3 shows that the total number of functional components is 3,043 and the number of failed components is 31, resulting in an accuracy of 98.9 percent in average. Meanwhile, Summers [26] presents experimental results on logical structure analysis with text lines from in-house 13 technical reports, comprising about 360 pages. Although the work of Summers was not based on the same data, it is the most comparable work with ours among the previous works that take text lines as inputs.

As a result, the proposed method has performed better than the work of Summers with an accuracy of 90.1 percent. This is because the proposed method considers adjacent
Fig. 25. The labeling result of the generalized functional structure tree in Fig. 16.

```xml
<!ELEMENT Sec-Body (Section*, Reference)>  
<!ELEMENT Section (Sec-Header, Paragraph*, Sub-Section*)>  
<!ELEMENT Sub-Section (Sub-Sec-Header, Paragraph*, Sub-Sub-Section*)>  
<!ELEMENT Sub-Sub-Section (Sub-Sub-Sec-Header, Paragraph*)>  
<!ELEMENT Reference (Sec-Header, Ref-Item*)>
```

Fig. 26. A document model extracted from the generalized tree in Fig. 25.

```xml
<!ELEMENT Sec-Body - - (Section*, Reference)>  
<!ELEMENT Section - - (Sec-Header, Paragraph*, Sub-Section*)>  
<!ELEMENT Sub-Section - - (Sub-Sec-Header, Paragraph*, Sub-Sub-Section*)>  
<!ELEMENT Sub-Sub-Section - - (Sub-Sub-Sec-Header, Paragraph*)>  
<!ELEMENT Reference - - (Sec-Header, Ref-Item*)>  
<!ELEMENT Sec-Header - - (#PCDATA)>  
<!ELEMENT Sub-Sec-Header - - (#PCDATA)>  
<!ELEMENT Sub-Sub-Sec-Header - - (#PCDATA)>  
<!ELEMENT Paragraph - - (#PCDATA)>  
<!ELEMENT Ref-Item - - (#PCDATA)>  
```

Fig. 27. An SGML DTD generated from the logical structure of the document model in Fig. 26.

```xml
(element Sec-Header (make paragraph font-size: 24pt  
font-weight: 'bold  
quadding: 'left  
line-spacing: 24pt  
space-before: 12pt  
space-after: 12pt ))

(element Paragraph (make paragraph  
font-size: 24pt  
font-weight: 'medium  
quadding: 'left  
first-line-start-indent: (if first-sibling?)  
0pt  
Paragraph-Indent)  
  space-before: 0pt  
  space-after: 0pt  
  color: (RGB-COLOR 0. 0. 0.)))
```

Fig. 28. A partial DSSSL style sheet.
Failure cases and their errors are as follows: Errors can be largely classified into the merging and splitting of a body and the insertion and deletion of a header. First, most of the body merging is due to the case where a paragraph, which contains a start line with irregular alignment, lists, or equations, has been merged with the previous one. On the other hand, most of the splitting errors are cases where some equations have been incorrectly identified as the start line of a paragraph. Other cases are when a body has been merged or split due to editing error and a part of text lines of a paragraph was incorrectly identified as a caption of an adjacent nontext object. All the errors for deletion and insertion of headers are cases where section titles with the third level have been incorrectly identified as some text lines of body or vice verse.

Another measure for the accuracy of the proposed method can be found by computing differences between the generated hierarchy and the correct hierarchy. For this purpose, the conventional tree edit of Zhang and Shasha [39] has been applied to this problem. Specifically, this paper considers interior nodes in the same level that have the same label and all costs of edit operations are 1 to maintain simplicity.

The difference as raw integer figures provides very little information. Hence, the difference given in this paper is normalized by the sum of the sizes of the correct hierarchy and the derived hierarchy. As shown in Fig. 29, the experimental result has yielded an overall difference of 0.01.

On the other hand, since the method uses the repetitive characteristic of headers to extract hierarchical structure, it can be assumed that accurate identification of headers is more important than that of bodies. As
illustrated in Fig. 30, suppose four kinds of erroneous sequences,
\{H1, B1, B1, B1\}, \{H1, B1, H2, B1\},
\{H1, B1, H2, B1, H3, B1\},
and \{H1, B1, H2, B1, B1, B1\} are extracted from experimental images that correspond to a sequence, 
\(S = \{H1, B1, H2, B1, B1\}\). The four sequences are cases where a header \(H2\) and a body \(B1\) have not been identified and a header \(H3\) and a body \(B1\) have been misidentified additionally, respectively.

If the Zhang’s and Shasha’s method is applied to compute the difference between hierarchical structures, the differences between \(S\) and the erroneous sequences can be expressed as \{deletion of \(S1\), deletion of \(H2\), deletion of \(B1\), \}[insertion of \(S2\), insertion of \(H3\)], and \}[insertion of \(B1\), respectively. As a result, the identification error of headers introduces more structural differences. Therefore, for an accurate creation of the hierarchical structure, the relevant identification of headers is more important than that of bodies. Particularly, we will develop a method for solving the identification error of headers and bodies as a future work.

The final measure for the accuracy is the identification rate of logical objects. The proposed system has performed labeling without parsing error for all experimental data. Experimental results show that all of the labeling errors are due to misidentification of headers or bodies and there is no case where a wrong label is given to the correctly identified header or body.

For the improvement in processing speed, the parsing method takes a functional structure tree rather than text lines as the basic units. Normally, because a syntactic method introduces frequent backtracking, most of the processing time is spent in the parsing process. We have examined the frequency rate of the text lines and functional components out of the total experimental data. As a result of examination, we have found that the total numbers of text lines and functional components are 24,123 and 3,043, respectively, which shows 1: 0.126 ratio.

Therefore, the proposed structure analysis method has faster processing time than the conventional methods of which the basic units for parsing are either pixels or text lines. On the other hand, because the algorithm for creating a functional structure tree from a sequence of headers and bodies is based on repeating headers, it runs in time \(O(mn)\), where \(m\) and \(n\) denote the numbers of a header and a sequence, respectively.

On the other hand, the document image analysis and understanding is normally divided into geometric structure analysis and logical structure analysis. Geometric structure analysis generally depends on the physical layout of a specific publication. On the contrary, logical structure analysis is relatively independent of specific publications. This is because, for example, publications, which fall within the technical journal class, have the same or almost identical logical structure. Therefore, we can say, in this sense, that logical analysis methods are more general than geometric analysis ones in terms of application domains. Particularly, because the proposed document model can represent logical schema of arbitrary structured documents, our work is useful to many practically relevant application areas with logical hierarchy.

6.2 Comparison with Related Works

This paper presents the syntactic method for logical structure analysis and the creation method of a document model targeting multipage documents. As mentioned before, most of the previous works [6], [7], [9], [10], [11], [15], [17], [18], [29], [30], [31], [32] target single-page documents. Furthermore, because they do not support the automated creation of a document model, the design and testing process of a document model should be repeated many times to reflect error images.

The proposed method extracts primary structures based on various types of geometric characteristics such as column type, height, alignment, density, and line spacing. Additionally, for the identification of secondary structures, the method uses their type and order and the frequency of their children. On the contrary, most of the conventional methods support limited amount of structure analysis because they only consider simple information such as the length and frequency of pixels [6], [7], the relative location of text regions [12], [13], [21], and the location and size of white spaces [25], [26].

Normally, more than 70 percent of the processing time is spent on recursive parsing in syntactic structure analysis [6], [7]. Therefore, for faster parsing, less grammar and input are required. Because the related works of Nagy et al. and Krishnamoorthy et al. take a pixel as the basic unit for parsing, a large number of grammars are necessary. Conway, who has presented the method based on text lines, has suggested that text blocks, which can be formed by merging adjacent text lines, would be appropriated for basic units of parsing for the improvement in speed [17]. Therefore, the proposed method is faster than the previous works whose input unit is either a pixel or a text line because it is based on functional components with hierarchical structure.

On the other hand, Klein’s and Fankhauser’s method and Klein’s and Abecker’s method provide syntactic structure analysis of multipage documents and automated creation of a document model. However, their character-based parsing
is not for document images based on geometric characteristics but for electronic documents with texts.

7 CONCLUSIONS

Since SGML and XML are good tools for embedding logical structure information into documents and independent of platform, they are widely accepted as a standard format for structured documents in various fields. Therefore, in this paper, we have presented the syntactic logical structure analysis method for creating an SGML/XML document from paper-based documents with multiple pages and hierarchical structure.

To improve the processing speed, the proposed parsing method is based on a functional structure tree. The principles of Gestalt psychology are used to extract a sequence of functional components. A functional structure tree is created by splitting the sequence based on repetitive headers.
For accuracy of structure analysis, DSDL is proposed for an effective representation of a document model. DSDL is able to describe not only geometric characteristics of primary structures but also hierarchical structure of secondary structures. As a reasonable way to customize the system for documents with different structures, the creation method of a document model has been proposed. An SGML DTD and a DSSSL style sheet are produced from the logical and geometric information in a document model, respectively.

The parsing method creates a logical structure tree by applying a document grammar described in DSDL on a functional structure tree and generates an SGML/XML document as the result of structure analysis by doing a depth-first search on a logical structure tree. Experimental results with a multitude of documents show that the method has performed logical structure analysis on documents with multiple page successfully and generated a document model automatically. The method generates SGML/XML documents as the result of structural analysis, so that it enhances the reusability of documents and independence of platform.

On the other hand, for proving the method on various kinds of document domains, our future works include experimenting with a larger set of documents. Furthermore, we will present the properties and criteria for systematic performance comparison with the conventional methods for logical structure analysis as future works. For example, to evaluate an automated creation method of a document model, the properties and criteria for comparing automatically created document models with reference models prepared by human experts are required. The creation method of a document model identifies repetitive common patterns from the functional structure tree and introduces Kleene star “*” to produce a regular expression that generalizes the pattern. To express in more elaborate grammar, our future work will use “+?” and “?” additionally.

ACKNOWLEDGMENTS
This work was supported in part by the Yonsei University Research Fund of 1999. The third author was supported by Korean Research Foundation Grant (KRF-2002-005-H2002). The authors would like to thank the reviewers and the associate editor for their helpful and constructive comments.

REFERENCES


Kyong-Ho Lee received the BS, MS, and PhD degrees in computer science from Yonsei University, Seoul, Korea, in 1995, 1997, and 2001, respectively. Previously, he worked as a guest researcher in the IT Laboratories at NIST (National Institute of Standards and Technology), Maryland. Currently, he is an assistant professor in the Department of Computer Science at Yonsei University. His research interests include document engineering, knowledge and data engineering, pattern matching, markup languages, and multimedia. He is a member of the Korea Information Science Society, the Korea Information Processing Society, the Korea Multimedia Society, the ACM, and the IEEE Computer Society.

Yoon-Chul Choy received the BS degree from Seoul National University, Korea, in 1973, and the MS and PhD degrees from the University of California, Berkeley, in 1976 and 1979, respectively. He is a professor of computer science at Yonsei University in Korea. After working as a research staff member for Lockheed Corp. and Rockwell International, he joined Yonsei University in 1984. Currently, he leads the Multimedia & Graphics Laboratory at Yonsei University. He is the executive vice president of the Korea Multimedia Society and has served as the chairman of Korea eBook Standard Committee. He was also a visiting professor at the University of Massachusetts and the Keio University (SFC) in Japan. His main areas of interests are multimedia document processing, 3D user interface in cyberspace, virtual reality, and eLearning. He is a member of the IEEE Computer Society.

Sung-Bae Cho received the BS degree in computer science from Yonsei University, Seoul, Korea, in 1988 and the MS and PhD degrees in computer science from KAIST (Korea Advanced Institute of Science and Technology), Taejeon, Korea, in 1990 and 1993, respectively. He worked as a member of the research staff at the Center for Artificial Intelligence Research at KAIST from 1991 to 1993. He was an invited researcher of Human Information Processing Research Laboratories at ATR (Advanced Telecommunications Research) Institute, Kyoto, Japan, from 1993 to 1995. In 1996, he was invited as a visiting scholar at the University of New South Wales, Canberra, Australia. Since 1995, he has been an associate professor in the Department of Computer Science, Yonsei University. His research interests include neural networks, pattern recognition, intelligent man-machine interfaces, evolutionary computation, and artificial life. Dr. Cho was awarded outstanding paper prizes from the IEEE Korea Section in 1989 and 1992, and another one from the Korea Information Science Society in 1990. He was also the recipient of the Richard E. Merwin prize from the IEEE Computer Society in 1993. He was listed in Who’s Who in Pattern Recognition from the International Association for Pattern Recognition in 1994, and received the best paper awards at the International Conference on Soft Computing in 1996 and 1998. He is a member of the Korea Information Science Society, INNS, the IEEE, and the IEEE Systems, Man, and Cybernetics Society.

Dr. Cho received the BS degree from Seoul National University, Korea, in 1973, and the MS and PhD degrees from the University of California, Berkeley, in 1976 and 1979, respectively. He is a professor of computer science at Yonsei University in Korea. After working as a research staff member for Lockheed Corp. and Rockwell International, he joined Yonsei University in 1984. Currently, he leads the Multimedia & Graphics Laboratory at Yonsei University. He is the executive vice president of the Korea Multimedia Society and has served as the chairman of Korea eBook Standard Committee. He was also a visiting professor at the University of Massachusetts and the Keio University (SFC) in Japan. His main areas of interests are multimedia document processing, 3D user interface in cyberspace, virtual reality, and eLearning. He is a member of the IEEE Computer Society.

For more information on this or any computing topic, please visit our Digital Library at http://computer.org/publications/dlib.