Using definite clause grammars to build a global system for analyzing collections of documents

Joseph Chazalon[a] and Bertrand Coüasnon[a]

[a]INSA de Rennes, Avenue des Buttes de Coësmes, F-35043 Rennes
UMR IRISA, Campus de Beaulieu, F-35042 Rennes
Université Européenne de Bretagne, France

ABSTRACT

Collections of documents are sets of heterogeneous documents, like a specific ancient book series, having proper structural and semantic properties linking them. A particular collection contains document images with specific physical layouts, like text pages or full-page illustrations, appearing in a specific order. Its contents, like journal articles, may be shared by several pages, not necessary following, producing strong dependencies between pages interpretations. In order to build an analysis system which can bring contextual information from the collection to the appropriate recognition modules for each page, we propose to express the structural and the semantic properties of a collection with a definite clause grammar. This is made possible by representing collections as streams of document images, and by using extensions to the formalism we present here. We are then able to automatically generate a parser dedicated to a collection. Beside allowing structural variations and complex information flows, we also show that this approach enables the design of analysis stages, on a document or a set of documents. The interest of context usage is illustrated with several examples and their appropriate formalization in this framework.

Keywords: document collections, historical documents, batch processing, system design, system generation, data flow, structural recognition, attribute grammars, definite clause grammars

1. INTRODUCTION

In this paper, we address several problems which raise when processing *similar* collections of documents, and more specifically collections of historical documents. We call *"collection"* an heterogeneous *set* of document images, or *"pages"*, such as:

- each page may be structured according to a particular physical layout, like multicolumn text pages, full-page pictures or domain-specific tables, and may therefore require appropriate content extraction tools;
- the nature (physical layout) and the order (or sequence) of the pages that may appear in a collection are characteristics of a kind of collection, and can be known (and described) before processing one collection of this kind;
- the content to extract is often shared by several pages, leading to a strong interdependence between the results produced by the recognition processes of each isolated page.

Further author information: (Send correspondence to J.C.)
J.C.: E-mail: joseph.chazalon@irisa.fr, Telephone: +33 (0)2 99 84 75 28
B.C.: E-mail: bertrand.couasnon@irisa.fr, Telephone: +33 (0)2 99 84 74 11

An example of a particular collection type may be a series of ancient poetry books[*] containing different types of pages in variable order, for which we try to extract a global content, like poems, which can be shared by several pages. The following sequences of pages may be belong to instances of this kind of collection:

```
cover_page, table_of_contents, poem_page, illustration_page, ...
cover_page, poem_page, poem_continued, poem_page, ..., table_of_contents, ...
```

The case of historical documents emphasizes the need for contextual information during analysis, as their alteration through time and the fact they were produced for human understanding often leads to ambiguous interpretations of isolated elements. Thus, we need to make use of *a priori* knowledge about a collection of historical documents to be able to extract its content.

Our approach aims at providing a flexible and efficient way to process various document collections, so as to be able to face structural variations in the collection (nature and order of pages), but also to allow various information flow between each image recognition process in order to use contextual information, redundant information and any available knowledge during the analysis of an image. Furthermore, to build a real analysis system, we sometimes need to specify easily how to design stages or evaluation strategies, like, for example, gathering statistical information on a subset of documents before processing the whole collection using this knowledge.

The purposes of this paper are then:

1. to propose modifications of definite clause grammars (DCGs), which can be seen as a logical implementation of attribute grammars (AGs), that allow to easily represent a collection of document images for processing, to integrate external analysis processes and to generate a collection-level driving system;

2. to show that this formal approach permits to express *a priori* knowledge about a collection, so as to model structural variations of a collection, enable intelligent information flow between analysis modules, and design analysis stages or global evaluation strategies.

Our presentation is divided as follows: first we review known approaches and evaluate their ability to face the formulated problem in Sec. 2; then we present in Sec. 3 the theoretical framework (namely attribute grammars and definite clause grammars) that we use in our approach. We are then able to introduce in Sec. 4 the extensions we propose to the base framework in order to model and process a collection of document images. After that, we detail in Sec. 5 the different aspects of a collection that our approach enables to tackle. In Sec. 6, we finally sum up our contribution and present some future work.

2. KNOWN APPROACHES

It is important to note that implementing from scratch the analysis of a given collection with scripts is not reasonable when considering real cases. The complex sequence of calls to document page analyzers and the information flow between them may be impossible to express in a maintainable solution. Thus, we review in this section different approaches related to the analysis of collections of documents, and in particular those which deal with how to efficiently build a system dedicated to the processing of a specific type of collection.

Unluckily, the most relevant approaches disclose little or no information about their inner machinery. This is the case of *docWORKS[e]*, a commercial product proposed by the European project *Meta-e*[†], which was used to extract content from German books dating from the 18th century using a rule-based system. Even if Le Bourgeois points out in Ref. 1, p.154, that is may be hard to adapt this system to other collections, it shows that a high-level specification may be an interesting way to face the problem of document collection analysis.

Some other systems address related problems, but may show some limitations, mainly because they cannot deal with various physical structures or limit their information flow. Lin,[2] first, proposed a content extraction

[*]In this paper, we will often refer to simple examples with books so as to avoid describing complicated document sets.
[†]`http://meta-e.aib.uni-linz.ac.at/`

system for books with very structured results, by making use of relationships between table of contents pages and book content pages. However, this system is entirely dedicated to books and cannot be easily adapted to other document collections. On the opposite, some systems allow many variations, like mass or large-scale digitization projects competing for the *ICDAR 2009 Book Structure Extraction Competition*.[3] They are working on building generic systems able to extract structure information from any digitized book. Nevertheless, as they rarely make assumptions about the kind of element which is processed, and then do not make use of *a priori* knowledge about a collection, they only extract limited information with little structure.[4]

Finally, some approaches like smartFIX,[5] a mail-routing product which processes important amounts of document pages from incoming and outgoing communication in a company to automatically perform business tasks, have some qualities. They can face various document types, provide interesting tools to adapt the system and have efficient workflows that integrate human verification and system training. However, their on-the-fly recognition constraints limit information flow patterns between page analysis to a local merging of information and prevent from managing relationships between distant document images, like a table of contents page and a common text page in a book. As a result, this kind of system does not appear fitted to collections with strong dependencies between distant document pages.

3. THEORETICAL FOUNDATIONS

This section quickly describes the basis over which our contribution is built. We need to introduce the attribute grammar (AG) formalism and the definite clause grammars (DCGs) formalism before extending them, in Sec. 4, for the purpose of document collection analysis. AGs can be seen as a structuring framework for the description of collections of document images; and DCGs can be considered as a compilable expression of AGs in a logical programming model.

3.1 Attribute Grammars Formalism

AGs were initially introduced by Knuth[6] for *"specifying and implementing the (static) semantic aspects of programming languages."* Since then, this formalism was successfully used in various fields. In particular, the work of Tsai and Fu[7] in the field of pattern recognition showed that an AG can be a powerful mean to combine statistical and structural tools in a recognition system. Paakki explains[8] that their versatility enables AGs to *"be used in many other fields where relations among structured information play a central role."*

AGs were extensively described in many works,[6,8,9] which clearly defined this formalism, and we encourage the reader to refer to the given references for a complete overview. What follows is a summary of the notations and the vocabulary we will use in this article.

We will note $AG = <G, A, R>$ an attribute grammar where:

G	is a context-free grammar;
A	is a set of attributes;
R	is a set of semantic rules.

The context-free grammar G will be noted $G = <N, T, P, S>$ where:

N	is a set of non-terminal symbols;
T	is a set of terminal symbols;
P	is a set of production (or "derivation") rules;
S	is a starting symbol (or "axiom") belonging to N.

We will consider, as usual, that each non-terminal can have inherited and synthesized attributes. Nevertheless, in our case, we will require that both the start symbol and the terminal symbols can also have inherited and synthesized attributes, which is usually not necessary.[8]

3.2 Definite Clause Grammars as an Implementation Language

DCGs are a formalism proposed by Pereira and Warren.[10] They are defined by Paakki[8] as a *"a logic counterpart to context-free grammars such that non-terminals can be augmented with arguments, and productions can be augmented with arbitrary code in Prolog."* Most of the logical programming languages now provide a DCG library and can produce a parser after a compilation stage. We recommend the section 3.1 of Paakki's attribute grammar paradigms presentation[8] for further details on DCGs.

In our presentation, we will use the following notations (alternatives will be expressed with multiples derivations for a same left-hand part):

`LEFT --> RIGHT`	represents the classical derivation operator (`LEFT` generates `RIGHT`)
`E1 & E2`	represents the concatenation of `E1` and `E2` (generates `E1` then `E2`)
`$ TERMINAL`	represents the consumption of a terminal (generates `TERMINAL`)
`EPSILON`	represents the generation of an empty element in a rule
`{PROLOG_CODE}`	represents some embedded Prolog code in a rule
`non_term`	non terminals are written in lower case
`SomeAttribute`	attributes (logical variables) start with an upper case letter for each word
`non_term Arg1 ... ArgN`	optional arguments follow the symbol name and are separated with spaces

DCGs can be used to easily specify, with a succinct and appropriate language, the key elements for the analysis of a collection: the definition of non-terminals (N), production rules (P), and the start symbol (S) are quite straight-forward. The main originality comes from the fact that attributes (A) are logical variables and semantic rules (R) are logical relations (predicates). We will show in Sec. 4.2 what are the terminals (T) suitable for a collection of document images and how to use them. The recognition automaton which is generated is a *top-down* $LL(k)$ parser, and it enables to evaluate the attributes while parsing the input, blurring the traditional distinction between syntactical and semantic analysis, as both are performed at the same time.

Furthermore, this formalism can be readily extended in to increase its expressive power, thanks to the strong integration of the logical language in the production rules, and therefore directly lead to an improved associated parser.

4. EXTENSIONS FOR DOCUMENT COLLECTION PARSING

This section describes the extensions we propose to AGs and DCGs in order to be able to use them in the context of the analysis of collections of document images. We first present the interests of AGs when trying to describe a collection of document images. Then, we show that 3 key extensions enable to use DCGs to express a global analysis process dedicated to a collection of document images. Those extensions are:

1. the use of *special input stream* for the parser;
2. the definition of a *set of external image analyzers*;
3. a *special operator* which enables to use the modified input stream and the external analyzers.

This permits to automatically generate a global process as a composition of image analyzers, capable of calling them with an appropriate context.

4.1 Expressive Power of Attribute Grammars for Document Collections

In the case of the analysis of collections of heterogeneous or interdependent documents, AGs can be used to express the structures of the collections and the dependencies between each document image. The *production rules*, first, enable to adjust the description grain and to propose several abstraction levels (like book, chapter and page) where it is possible to merge enough information for decision making by delaying local decisions. (This is a first way to use context, a second way is to bring information from a global environment to a local decision process, and will be presented in Sec. 4.3.) *Alternatives* in production rules enable to express the structure of the collection (the nature and the order of the images it contains) and its variations, transforming structural variability into a syntactical problem. Beside this, *attributes and semantic relations* can model links between data used or produced by any analysis modules, and permit, among others, to specify how contextual information flows.

Unlike an XML schema which may express the valid logical structures for a collection, an AG gives, in a formal way, a full mapping between a physical structure and a logical structure, while limiting the valid configurations for each. We can then consider that AGs give enough information to try to solve the inverse (as opposed to synthesis) problem of the recognition of images collections.

We can therefore see that, in order to use the AGs modeling framework for the analysis of a collection of document images, we first need to describe the structure of our collection with a context-free grammar (G), which requires to decompose the global collection (S) into appropriate abstraction levels (N) clearly organized (P). We will suggest in Sec. 4.2 some appropriate terminal symbols (T). Secondly, we also need to describe what information will be exchanged during this process (A) and how its components are related (R). Finally, the actual recognition system needs to be implemented and run. DCGs can easily be used to define most of those elements, as shown in Sec. 3.2.

4.2 Document Collection as a Stream of Images

During common text parsing, the source file (a stream of characters) is processed though several well-defined stages. First, the lexical analyzer segments it into *lexemes* (some blocks of text), and categorizes them according to a function, giving them meaning. The resulting *tokens* are grouped in a single stream, and can, in the second stage, be analyzed by a parser. The terminal symbols (T) accepted by this parser are the token values it can take as an input.

For a collection of document images, things are a bit different: first, data is already segmented (images are well separated), and second, external information is often needed when analyzing a document image. This external information can be the result of the same analysis process on a previous page, some important parameter for an algorithm, or even a dictionary for word recognition. It is therefore important to be able to bring information from the global collection level to the isolated document image analysis level, and we need to be able to delay the categorization of each image until the global processing of the collection.

As a result, it is necessary for a parser working on a collection of document images to be able to accept *a stream of separated images* as an input, and lazily call appropriate analysis tool on each of them, with the appropriate context, when information contained in an image is needed. As no prior categorization is possible for any image, the only terminal we will use in our system will be a single image, and the input stream of some generated parser will be a stream of document images, appearing in the same order as how documents are physically organized in the collection. This first modification can be summarized as:

$$T = \text{document images}$$

The implementation of such a parser is possible thanks to 3 key points:

1. due to the flexibility of the DCGs model, we can define the type of terminal symbols we want the parser to work on (streams of characters are just one option);
2. so as not to store every image data in memory, we can simply keep a reference to it in our stream;
3. by adding a new operator (presented in Sec. 4.3) to our DCG description, we can cope with those references to the images, and call an appropriate contextual recognition on an image when needed.

When calling the collection parser, we will only need to give it a stream of references to images data.

4.3 Integrating External Analysis Processes

We have seen how to build a parser that accepts a stream of images as an input, but we still need to explain how to trigger an (external) analysis process for a given image. What we propose is to add a set (E) of external analyzers to our AG formalism which can be used in our description, and to add a new operator (`ANALYZE`) to the presented DCG formalism which will be able to call those analyzers and cope with the special input stream.

4.3.1 Adding an external analyzers set to AGs

The set E of usable external analyzers is a set of definitions very similar to predicate definitions. For each external analyzer, we store:

- the name of the analyzer (its identifier in the grammar) and a reference to this external module (like a file name);
- the set of attributes (or parameters) it accepts and their type.

The AG definition then becomes an extended attribute grammar definition, which we note $AG_{\text{ext}} = < G_{\text{col}}, A, R, E >$ where:

A and R	are left unchanged;
E	is the set of external analyzers;
G_{col}	is a collection grammar.

The collection grammar is defined as $G_{\text{col}} = < N, T_{\text{img}}, P, S >$ where:

N	is a set of abstract levels in the collection;
S	is its top-level element;
P	describes the structure of the collection;
T_{img}	is a single element set containing a raw image reference prototype.

4.3.2 Adding a new operator to DCGs

The new operator, which we will note `ANALYZE`, enables to dynamically choose which analysis module (from E) has to be called for each document image, and to give it as much information as it needs. It also enables to extract specific information from each document image we encounter. From the collection point of view, it mainly permits to bring contextual information from the global process to a local document analysis module, in order to help it analyzing the current document image; and to retrieve information produced, in a prediction-verification fashion. This is a new terminal consuming operator which triggers a given "lexical and contextual" analysis on an image. The syntax of this operator is:

```
ANALYZE(external_analyzer_ident Param1 Param2 ... ParamN)
```

It works in three steps:

1. the parser consumes the current terminal (the document image under its head) and extract the reference to its content;
2. the external analysis module is invoked: thanks to a serialization mechanism, the external tool is provided the reference to the image and to any content given as input parameter to the operator (like inherited attributes);
3. when the external module terminates, its status and results are sent back to the global collection parser: if the external module failed recognizing the image, the global parser will backtrack to the previous alternative and explore it, otherwise the results sent in output parameters are used like synthesized attributes.

From the external module point of view, parameters allow information passing, and the only constraint each analysis module has is to be able to process an image and send a failure notification when it cannot accomplish its task. We then rely on a standard interface and a common serialization method between external modules and the global collection parser.

If an external analyzer is also based on a grammatical description, then we are able to express constraints over elements across document image boundaries, and we can use different representation space for the image analysis level (bi-dimensional for example) and the collection space (generally mono-dimensional). It shows that this approach is not a simple decomposition of a single grammar over different modules, because the input stream of the global collection analyzer is different from the one of the image processing modules. Furthermore, this integration of external modules only relies on a standard interface, allowing any kind of image processing tool which conforms to the later to be part of the global process.

5. DESCRIBING A COLLECTION WITH AN EXTENDED DCG

This section aims at illustrating how to describe the different aspects of a collection of document images, in order to cope with the various difficulties is contains, using the approach we presented. We present here examples which, while being simple, show how to express each of those aspects in a concise way. In a first time, we explain how describing the structural variation of the collection enables to dynamically call the appropriate analyzer on a given image, like in a syntactical problem. In a second time, we focus on the description of information dependencies between document images, and on how to express them using semantic construct, so as to enable an automated information flow. Finally, we show that the proposed formalization allows to describe analysis stages in order to build complex global systems.

5.1 Describing the Structural Variation of the Collection

We have identified two kinds of structural variations in a collection of document images. The first is a variation in the *nature* of the document image. The collection can contain documents with different physical structures, requiring different analysis methods. By "different documents", we can both consider effectively different documents, like a blank pages and text pages, or a similar documents which were altered and for which the primitive extraction produces an unpredictable result: we then have damaged and understandable documents. The following code excerpt belongs to a simplified description of a digitized book, showing how to cope with this kind of variation. The first rule, `book`, shows how to combine elements; the second, `some_blank_pages`, shows how to call an external analyzer and accept alternatives.

```
book             --> cover_page & some_blank_pages & table_of_contents &
                     some_blank_pages & book_content ...

some_blank_pages --> ANALYZE(blank_page_verifier) & some_blank_page
some_blank_pages --> EPSILON
...
```

The second variation kind lies in the *order* of the elements in the collection, like, for example, when a table of contents can be found at the beginning or at the end of a book, in a unpredictable way for each book of a series. It requires some more advanced techniques to be able to explore the collection of document in a more flexible order. The *Enhanced Position Formalism* we proposed[11] contains several operators that can be adapted for our collection recognizer. Even if those operators were initially designed for a bi-dimensional analysis, their logic remains valid in our case. The generated parser may require some transformations, but we recommend Ref. 11 for further details. The two useful operators are:

- `FIND(rule) UNTIL (stop_condition)` which searches for the closest element from the current position that permits to apply production `rule`, unless the production `stop_condition` can be applied before;
- `AT(position)` which modifies the current position of the parser's head, and then allows a flexible navigation in the stream to analyze, with jumps, absolute positioning, or direction change.

In the following example, we show how to accept book structures which can contain a table of contents at the beginning or at the end of the collection of document images.

```
book     --> find_toc & some_blank_pages & chapters

find_toc --> AT(collection_start) &
             FIND(table_of_contents) UNTIL (tenth_page_from_start)
find_toc --> AT(collection_end) &
             FIND(table_of_contents) UNTIL (tenth_page_from_end)
...
```

The `book` rule describes the structure of the book with a list of the elements to recognize. The `find_toc` rule describes the search for a table of contents from the start or from the end of the collection, considering only ten pages before giving up. For sake of simplicity, the next examples will not use those extensions.

5.2 Describing Information Dependencies in the Collection

The ability to control the information flow going to and coming from analysis components is as important aspect of document collection recognition. We may need to make information flow in various ways, so as to express contextual relationships between documents. We could want to share page numbering information between neighbor pages. We could also want to bring table of contents data from the beginning of a book to the first page of each chapter to check chapters' titles. Furthermore, as we have shown how to accept variations in collection structure, we cannot predict when some piece of information will be produced nor used. We may then need some "automatic plumbing" mechanism which takes care about the activation and the storage of any piece of information.

Luckily, the unification process and the logical variables provided by a Prolog implementation enable a seamless flow of information between the rules of a DCG. In the following example, we transformed a bit the previous example by adding an attribute `TocData` that will enable table of contents information to reach the analyzer of a chapter's title page.

```
book             --> table_of_contents TocData & some_blank_pages & chapters TocData
...
a_chapter TocData --> ANALYZE(chapter_title_page TocData) & other_chapter_pages
...
```

Information produced in some way by the analysis of a table of contents section is stored in the logical variable `TocData` which is later passed to the `chapters` rule. It will be passed to each chapter, and the `a_chapter` rule accepts a parameter which is transmitted to the `chapter_title_page` analyzer. This information flow remains valid even if we change the inner behavior of the recognition modules, or if the information produced changes. We may also add various attributes to our grammar to enable the exchange of many other elements, between any recognition module.

The previous information flow shows how to broadcast information produced during the analysis of one or more images (a table of content) to other images (chapter pages) where it can be useful, even if it needs to be stored for later use because it cannot be transmitted immediately : here the recognition of blank pages delays the routing of the content summary. Furthermore, various other schemes of information flow could easily be designed, in order to distribute, to merge or even to accumulate information.

Finally, as any kind of information can be exchanged with this approach, we are not limited to results of content recognition. We can, for example, bring some threshold value or some dictionary to each image analyzer. We can also transform information between each image analyzer call, and then propose a more complete integration of the various processes which can be useful in the analysis of a collection of document images. As a result, using semantic constructs of a DCG enable to exchange, along the global recognition process, any kind of information that can be used to validate or guide image analysis work.

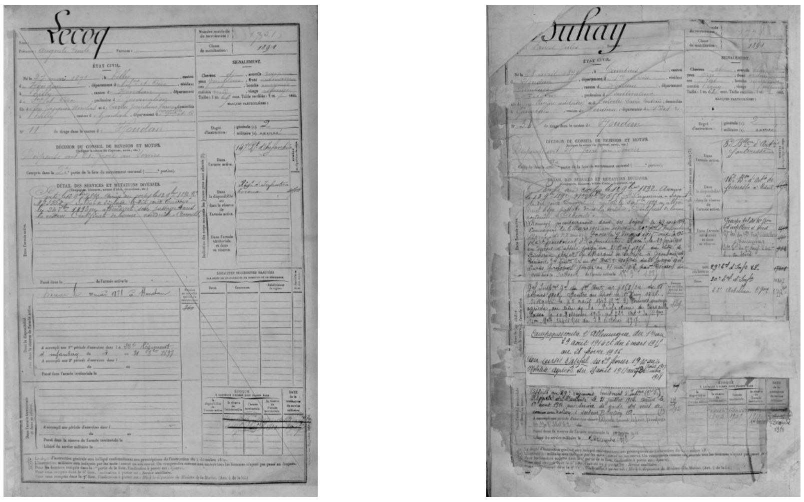

Figure 1. A clean page (left) producing clear rulings, and a damaged one (right), from the same register.

5.3 Composing Analysis Stages in a Global Process

A global content extraction process may also require to specify more precisely stages in the analysis, like, for example, alternation between recognition, validation and training processes. Those analysis stages can be expressed as *rules*, working on a set of document images, and *predicates*, transforming information, that a global description combines in a consistent way.

We are therefore capable of expressing a full specification of information generation, transformation and sharing (and forgetting) over a collection of document images. It is particularly interesting when we want to use the result of the recognition of some element to help the recognition of another element after some transformation.

In our last example, we consider a practical case of historical documents in order to show how our modeling approach enables to express the global composition of several complicated processes. In this case we want to localize specific content areas, separated by rulings (represented by line segments), in military registers dating from the early 20^{th} century. Each register is composed of several pages with the same visual structure (dimensions of content areas are stable) but between registers this structure can change. Inside a register, we have to distinguish two cases for each page:

1. the page is well conserved and line detection produces clear results;
2. the page is damaged and we cannot base our analysis only on line extraction.

Figure 1 shows a clean page image and a damaged one belonging to the same register. In order to cope with damaged document, we perform a two pass analysis on each register.

The first pass tries to locate ruling positions on clean images. Then, a machine learning stage (not detailed here) uses collected information to produce a statistical layout (cell dimensions) model for the content areas of a register. Finally, the second pass tries to locate some damaged separators using the statistical layout information: we use the redundancy of structural information in the register to create a contextual information for each page image. The following example shows how to collect separator positions for a register. The first derivation tries to extract positions with an external analyzer (`find_clean_separators`), the second ignores unreadable pages (using the terminal consuming operator `$`), and the third builds an empty result when no more document is available.

```
locate_content_separators Separators -->
       ANALYZE(find_clean_separators PageResults) &
       locate_content_separators OtherPagesResults &
       {merge_result PageResults OtherPagesResults Separators}
locate_content_separators Separators -->
       $ UnreadablePage &
       locate_content_separators Separators
locate_content_separators Separators -->
       EPSILON &
       {build_empty_result Separators}
```

The global process can then be expressed in a few lines, in order to combine separator localization, knowledge production (the "layout", storing cell dimensions) and enhanced localization using this statistical information. We use here a `SELECT(rule)` operator, proposed in Ref. 11, which restores the parser's head position after the recognition of some rule. It enables us to go back to the first document image, while conserving collected information.

```
process --> SELECT(locate_content_separators SomePositions) &
            {learn_register_layout SomePositions Layout} &
            enhanced_content_locator Layout
```

6. CONCLUSION

In this paper, we showed that DCGs can be a powerful yet simple formalism to design and generate a global recognition system dedicated to a specific type of collection of heterogeneous documents sharing some content, or having interdependent interpretation.

We proposed 3 extensions to AGs and DCGs: we suggested to consider images as terminals in order to easily model the collection; we used this representation to introduce a set of external analyzers working on images; and finally we added a new operator to our DCG parser to enable the use of this new input stream and of those external analyzers.

Using those extensions (and some we previously proposed[11]), we showed it was possible to dynamically choose which analyzer use on a given image, giving it all the context it needs thanks to a seamless flow of information between the collection level and the document image level. This was made possible through the usage of the syntactical and the semantic components of an AG to describe the different aspects of a collection: we expressed its structural variation using non-terminals and production rules, while we expressed information dependencies between processes with attributes and semantic rules. We also showed that by combining those description tools, we were able to define analysis stages in at the global collection level and to integrate various information-centered processes into a common collection analyzer which can be automatically generated.

Our further work includes the exploration of the integration of interaction stages with a human operator; and a better separation of the different aspects of the system, in particular information flow patterns, in order to simplify their specification.

ACKNOWLEDGMENTS

This work has been done in cooperation with the *Archives départementales des Yvelines* in France, with the support of the *Conseil Général des Yvelines*.

REFERENCES

[1] Le Bourgeois, F., [*Les documents écrits*], ch. Reconnaissance des structures, 87–178, Informatique et Systèmes d'Information, Hermes Lavoisier (2006). 2-7462-1143-2.

[2] Lin, C. C., Niwa, Y., and Narita, S., "Logical structure analysis of book document images using contents information," in [*Proc. Fourth International Conference on Document Analysis and Recognition (ICDAR)*], **2**, 1048–1054 (Aug. 1997).

[3] Doucet, A., Kazai, G., Dresevic, B., Uzelac, A., Radakovic, B., and Todic, N., "ICDAR 2009 Book Structure Extraction Competition," in [*Proc. Tenth International Conference on Document Analysis and Recognition (ICDAR)*], (2009).

[4] Coyle, K., "Mass digitization of books," *The Journal of Academic Librarianship* **32**(6), 641–645 (2006).

[5] Klein, B., Dengel, A., and Fordan, A., [*Reading and Learning*], vol. 2956/2004 of *Lecture Notes in Computer Science*, ch. smartFIX: An Adaptive System for Document Analysis and Understanding, 166–186, Springer Berlin / Heidelberg (2004).

[6] Knuth, D. E., "Semantics of Context-Free Languages," *Mathematical Systems Theory* **2**(2), 127–145 (1968).

[7] Tsai, W.-H. and Fu, K.-S., "Attributed grammar - tool for combining syntactic and statistical approaches to pattern recognition," *Systems, Man and Cybernetics, IEEE Transactions on* **10**, 873–885 (Dec. 1980).

[8] Paakki, J., "Attribute grammar paradigms—a high-level methodology in language implementation," *ACM Computing Surveys* **27**(2), 196–255 (1995).

[9] Deransart, P., Jourdan, M., and Lorho, B., [*Attribute grammars: definitions, systems and bibliography*], Springer-Verlag New York, Inc., New York, NY, USA (1988).

[10] Pereira, F. C. N. and Warren, D. H. D., "Definite clause grammars for language analysis–a survey of the formalism and a comparison with augmented transition networks," *Artificial Intelligence* **13**(3), 231–278 (1980).

[11] Coüasnon, B., "DMOS, a generic document recognition method: application to table structure analysis in a general and in a specific way," *International Journal on Document Analysis and Recognition* **8**, 111–122 (June 2006).