Interest of Syntactic Knowledge for On-Line Flowchart Recognition

Aurélie Lemaitre[1], Harold Mouchère[2], Jean Camillerapp[3], and Bertrand Coüasnon[3]

[1] IRISA - Université de Rennes 2
Campus de Beaulieu 35042 Rennes Cedex, France
aurelie.lemaitre@irisa.fr
[2] IRCCyN/IVC - UMR CNRS 6597
Rue Christian Pauc - BP 50609 44306 Nantes Cedex 3, France
[3] IRISA - INSA
Campus de Beaulieu 35042 Rennes Cedex, France

Abstract. In this paper, we address the problem of segmentation and recognition of on-line *a posteriori* flowcharts. Flowcharts are bi-dimensional documents, in the sense that the order of writing is not defined. Some statistical approaches have been proposed in the literature to label and segment the flowcharts. However, as they are very well structured documents, we propose to introduce some structural and syntactic knowledge on flowcharts to improve their recognition. For this purpose, we have used an existing grammatical off-line method with on-line *a posteriori* signal. We apply this work on a freely available database. The results demonstrate the interest of structural knowledge on the context to improve the recognition.

Keywords: structured documents, flowcharts, symbol recognition, on-line analysis, grammatical analysis, segmentation.

1 Introduction

We work in the context of handwritten document recognition, and more particularly complex bi-dimensional documents such as schemes, plans, diagrams, flowcharts (example on figure 1).

These kinds of documents are *complex* as they are not only made of text but also of symbols, shapes, boxes... Consequently, the steps of segmentation and of structure analysis are essential before the handwriting recognition.

The *bi-dimensional* aspect of these documents is also very important for the recognition. Indeed, in these documents, the order of writing is not necessarily from left to right, nor from top to bottom. Consequently the order of reading and of analyzing the document must be adapted depending on the content of the document. For example, even if the main orientation of a flowchart is left to right or top to bottom, these diagrams will be read by following the arrows, which can have any orientation.

The final objective of flowchart recognition is to produce a semantic analysis of its content. In this paper, we propose some first experiments: we focus our analysis to label each stroke of the flowchart and to group the strokes depending on the symbol they belong to. We work on the *a posteriori* analysis of on-line signal. These tasks present several challenges. First, the strokes represent either pieces of text or pieces of symbols, and confusion can occur between some strokes of text and some strokes of symbol. For example, a circular stroke could be the letter "o" or a circle shape, or even something else. The second challenge is to deal with the fact that most of the symbols are multi-strokes, and that the strokes of one symbol have not necessarily been written successively. To sum up, we are faced with two segmentation problems for the strokes: the confusion between symbols and text and the confusion between symbols.

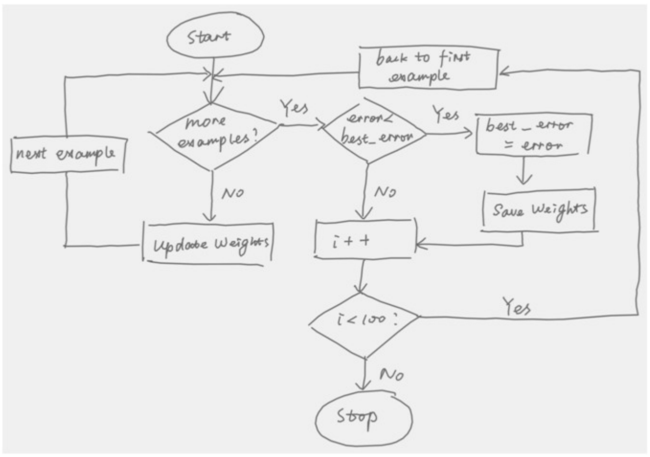


Fig. 1. Example of handwritten flowchart

Several methods have been proposed in the literature to deal with handwritten diagram recognition. However, one of the important limits of these methods is the difficulty to make the classification between strokes of text and of symbols. For example, Qi *et al.* [5] focus on diagrams that are only made of symbols. This is also a constraint mentioned by Feng *et al.* in [3]. As diagrams often contain both text and symbols, some authors assume that the user will explicitly choose a kind of writing, text or symbol, when drawing its diagram. For example, Tilak *et al.* [6] propose to add text on handwritten diagrams using a specific form. Yuan *et al.* [8] remove this constraint of separating text and symbol strokes, but impose symbols to be mono-stroke. This shows that the segmentation of strokes into text and symbols remains an open problem.

Concerning the flowcharts in particular, the existing methods are only based on statistical approaches. Thus, Yuan *et al.* [8] use a hybrid SVM-HMM for sketch recognition. Awal *et al.* [1] have also presented some work for flowchart recognition. Concerning the recognition of symbols, they apply two different methods: the separation of text and graphic symbols using a method based on entropy, and the recognition of symbols using TDNN or SVM after a step of stroke re-ordering. They also propose a global learning/recognition approach using dynamic programming and TDNN. However, these authors conclude that their statistical approach obtains limited results, due to the instability of stroke signal. Thus the introduction of structural knowledge could improve the recognition. This conclusion joins the work of Mace *et al.* [4] who use grammatical descriptions for the recognition of complex documents, such as electrical schemes.

In this paper, we propose to show that a structural method is particularly convenient for bi-dimensional complex handwritten documents. Thus, we present syntactic knowledge to describe the flowcharts is section 2. Then, we present how we have implemented this structural approach into an existing structural method (section 3). At last, we present in section 4 our experiments on a freely available database and demonstrate that our structural approach increases the recognition rates, by comparison with the statistical methods.

2 Syntactic Knowledge on Flowcharts

We first present the different symbols that compose a flowchart[1]. Then, we propose to express some syntactic knowledge on the organisation of the flowcharts, using a grammatical description. We conclude by the structural flexibility that is required by the analysis.

2.1 Existing Symbols

The flowcharts are used to describe algorithms or processes. They are made of different symbols such as circles, rectangles... Arrows are used to represent the control flow. Some text is present inside the symbols or close to the arrows. The figure 2 synthesizes the different kinds of symbols that can be found in our flowcharts.

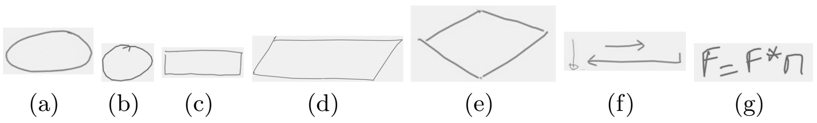


Fig. 2. Existing symbols on flowcharts: terminator, connection, process, data, decision, arrow, text

The terminator and the connection can be described as circular shapes, and more particularly oval or circles. The process, the data and the decision are described as specific quadrilaterals: rectangle, parallelogram and diamond. The arrows are made of a succession of line segments, possibly ended by a pointer.

[1] This work is applied to the database presented in [1].

2.2 Syntactic Rules

In this work, we propose some syntactic rules that enable the grammatical analysis of flowcharts. As a flowchart always begins by a connection or a terminator, we propose two ways to `StartDiagram`:

```
StartDiagram :: terminator, arrow, RestOfDiagram.
StartDiagram :: connection, arrow, RestOfDiagram.
```

These two rules call the analysis of the following symbols of the diagram, using `RestOfDiagram`.

Three kinds of symbols can continue a diagram: process, data, decision. They are followed by one or two arrows (for decision). So, these three rules recursively call `RestOfDiagram`.

```
RestOfDiagram :: process, arrow, RestOfDiagram.
RestOfDiagram :: data, arrow, RestOfDiagram.
RestOfDiagram :: decision,
                 arrow1, RestOfDiagram1,
                 arrow2, RestOfDiagram2.
```

The `RestOfDiagram` can also be the end of the diagram if we meet a terminator, a connection, or an element that has been seen before in the analysis, in the case of a loop.

```
RestOfDiagram :: terminator.
RestOfDiagram :: connection.
RestOfDiagram :: seenBeforeElement.
```

These rules that we have proposed simply describe the syntax of our flowcharts. We have realised a global description that ensures a global consistence in the recognition of the whole flowchart. We now detail some structural aspects of the analysis.

2.3 Structural Aspects

The application of the syntactic knowledge is based on the structural analysis of primitives: the strokes that are contained inside of the signal. However, even if the syntactic rules expressed below are very stable, the way to draw the symbols can vary a lot.

Firstly, we can mention the case of the arrows, presented in Figure 2(f). Indeed, the number of edges is not defined (often one, two or three . . .). Moreover, the end pointer of the arrow has a varying shape, or can be nonexistent. Consequently the structural analysis must be flexible enough to deal with all these cases.

Secondly, we are faced with the fact that the diagram is not built from left to right nor from top to bottom. Consequently, when looking for the `RestOfDiagram`, we have to study the four directions after a symbol: right, left, top, bottom.

Thirdly, we must notice that each symbol can be composed of a varying number of strokes. For example, figure 3 shows that a rectangle can be made of 1, 3, 5 strokes. Here again, the structural analysis must be flexible enough to deal with all these cases.

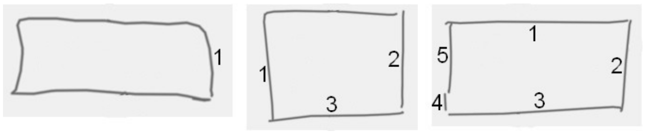


Fig. 3. Variable number of strokes for a rectangle: respectively 1,3,5

At last, the structural analysis must deal with the presence of text inside of the symbols. As presented on figure 4, the presence of text is not mandatory. The text can be on one or several lines and can overflow out of the symbols.

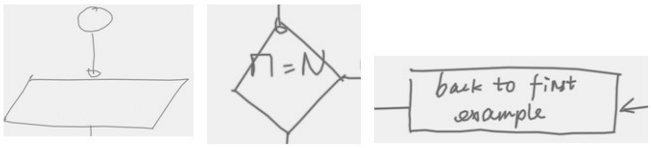


Fig. 4. Variable configurations of text inside of the symbols

As a conclusion, we can notice that the primitives that could be useful for symbol recognition (for example sides of a quadrilateral) are not always directly present in the on-line signal, and that the order of the strokes is not always relevant. Thanks to our grammatical structural approach, we will solve these local problems by using the relative positioning of the strokes in the image and ensure a global consistence of the recognition.

3 Implementation with a Grammar-Based Method

We have implemented our syntactic and structural description of flowcharts using an existing grammar-based method: DMOS.

3.1 Existing DMOS Method

The DMOS (Description and MOdification of the Segmentation) method [2] is a grammatical method for structured documents recognition. It is based on a grammatical language, EPF (Enhanced Position Formalism) that enables a syntactic and structural description of the content of a document. Once the description has been realized in EPF for a kind of document, the associated analyzer is automatically produced by a compilation step.

This method has been applied for the analysis of various kinds of documents: tabular, archive documents, mathematical formulae... and at a large scale (> 600,000 images). However, it had never been applied to on-line signal.

The use of DMOS method is particularly adapted to our problem. Indeed, DMOS enables to deal with the need of flexibility that we have exposed on section 2.3. Thus, as the method is based on logical programming, it naturally offers the possibility to express different ways for a given rule to succeed. It also offers the ability to backtrack, which is very convenient to deal with the various possible configurations of the diagram, and ensures that the final result of the recognition is globally consistent.

3.2 Adaptation to On-line Signal

Our first contribution has consisted in improving the existing DMOS method to make it treat on-line signal.

Thus, in the previous version of the method, the input signal was scanned images. We were extracting the connected components and the line segments that were present in the off-line analyzed image. These connected components and line segments were used as primitive for the grammatical description.

In our work, the input signal is on-line signal. Our contribution enables its interpretation. The strokes are represented by their bounding boxes and stands for the components. We extract a polygonal approximation from the on-line signal to compute the line segments, using the classical Ramer-Douglas-Peucker algorithm [7]. Consequently, our experience on scanned images with DMOS method can be used for on-line signal as the analysis is based on an homogeneous set of primitives.

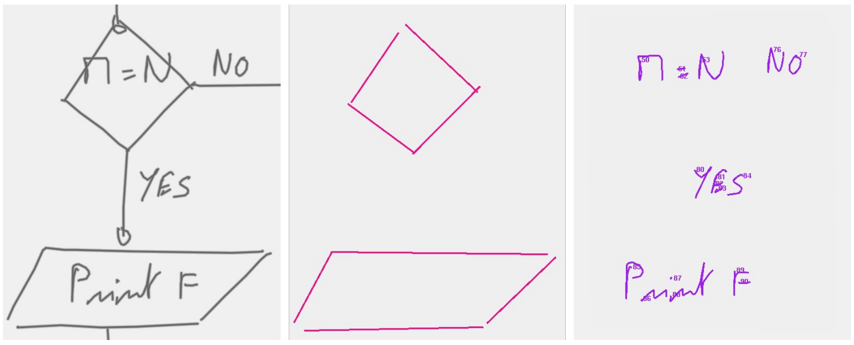
3.3 Interest of Two Sets of Primitives

As our grammatical description is based on the combination of two kinds of primitives (strokes and line segments) (figure 5), it enables to deal with some difficulties previously presented. For example, the line segments are convenient to detect the sides of the quadrilaterals: the figure 5(b) shows that the rectangles are composed of four line segments, whatever the number of strokes that compose them. On the opposite, the segmentation using strokes (figure 5(c)) enables an easy description of the text, even when it overlaps the sides of the symbols.

3.4 Grammatical Description

Our work has consisted in expressing our syntactic and structural rules, presented on section 2, using the two sets of primitives. This has been realized thanks to the grammatical EPF language.

The rules presented on section 2 have been expressed using EPF formalism. For example, we present the translation of the first `RestOfDiagram` rule with EPF syntax.



(a) Initial image (b) Line segments (for quadrilateral sides) (c) Strokes (for text)

Fig. 5. Primitives used for the grammatical analysis

```
restOfDiagram::=
     process P &&
     AT(aroundSymbol P) &&
     arrow A &&
     AT(otherArrowEnd A) &&
     restOfDiagram.
```

In EPF language, `&&` is the concatenation operator. `AT` is the position operator. It indicates where to find the next item for the following analysis. For example, once the process `P` is found, we search around `P` for an arrow. The rules `process` and `arrow` are non-terminals.

We now detail the analysis of the process rule. It is made of a rectangle with text inside.

```
process P::=
     rectangle R &&
     AT(insideSymbole R) &&
     text T.
```

As we said above (see figure 5), we combine two kind of primitives for the description of a process. The rule rectangle is based on finding four consecutive sides `S1, S2, S3, S4`. We use the `TERM_SEG` operator to search terminals in line segments. On the opposite, the text uses the strokes (stored as connected components). Consequently, the `text` rule uses the `TERM_CMP` operator to find a terminal `TxtStroke` among all the stroke primitives.

```
rectangle R::=
     TERM_SEG noCondS S1 &&
     AT(end S1) && TERM_SEG S2 &&
     AT(end S2) && TERM_SEG S3 &&
     AT(end S3) && TERM_SEG S4.
```

```
text T::=
     TERM_CMP TxtStroke &&
     AT(around TxtStroke) &&
     endOfText.
```

3.5 Parser

Once all the rules of section 2 are expressed in EPF language, we used the DMOS method to automatically generate the associated parser. Thanks to our grammatical approach, the obtained parser realises *in the same time* the three following tasks:

- structural analysis and segmentation,
- syntactic recognition of the document,
- labelling of strokes.

The analysis only succeed if we can find a global consistence for these three tasks. Consequently, the final structural segmentation is totally linked to the syntactic description.

This implementation enables to evaluate our approach on an existing database, and to show the interest of the syntactic knowledge for flowchart recognition.

4 Experimental Results

4.1 Database and Metrics

In order to evaluate our method, we have worked on the freely available database presented by Awal *et al.* in [1]. This database is made of handwritten flowcharts (example on figure 1), of various complexity (different patterns), that have been written by 31 writers using the Anoto pen technology. The table 1 summarizes the properties of the database. A ground truth is provided for each flowchart, containing the label for each stroke (one of the 7 classes presented on figure 2) and the segmentation of strokes into symbols. We notice that each stroke owns exactly one label.

Table 1. Properties of the database

	Writers	Patterns	Flowcharts	Strokes	Symbols
Validation set	31	14	248	23359	5541
Test set	15	14 other	171	15696	3792

The authors have identified two tasks: the labeling of each stroke and the correct segmentation and recognition of the symbols. A stroke is correctly labeled if the result label corresponds to the ground truth. A symbol is correctly segmented and recognized if the set of strokes corresponding to the symbol is exactly the same in the result and in the ground truth, and that the label of the symbol is correct.

4.2 First Results and Discussion

We have used the validation set to validate our grammatical description. We present the first results that we obtain on the test set in table 2.

Table 2. Our first results on the test set

Class	Correct stroke labeling	Correct symbol segmentation and recognition
Connection	80.0%	81.4%
Terminator	58.9%	70.3%
Data	84.7%	80.4%
Decision	84.0%	66.5%
Arrow	79.6%	68.9%
Process	85.7%	81.3%
Text	97.8%	71.7%
Total	**91.1**%	**72.4**%

We obtain a good recognition rate, 91.1% for the individual strokes. Indeed, thanks to our grammar, we are able to easily classify each stroke, depending on their position and their context. However, these first results show that the recognition at symbol level is weaker. Indeed, the metric is very strict and considers that a symbol is not recognized if only one small stroke is not joined to the segmentation. This is often the case in our analysis. Thus, we believe that, for this kind of documents, we should set up a metric that consider the syntax of the recognized symbols ("here is a connection") more than the presence of all the strokes.

We have also compared our results with the ones proposed by Awal *et al.* in [1], who use a statistical approach. The results in table 3 demonstrate the interest of the syntactic and structural knowledge for flowchart recognition. Thus, our method enables a big increase of non-text stroke and symbol recognition, thanks to the presence of context.

Table 3. Interest of our structural method for recognition

	Stroke labeling		Symbol recognition	
Method	Text	Non-text	Text	Non-text
Statistical [1]	73.9%	39.8%	71.9%	29.6%
Structural (ours)	**97.8%**	**80.6%**	**71.7%**	**72.8%**

5 Conclusion

In this paper, we have presented a way to use some syntactic knowledge for handwritten on-line flowchart recognition. We have proposed a grammatical description of this kind of document. This is particularly adapted to the bi-dimensional and structured properties of the flowcharts.

In our implementation, we have shown that the existing off-line method, DMOS, could be used for the analysis of on-line documents. We have used the EPF language to express the grammtical description of flowcharts. The ability to back-track, provided by DMOS method, and the verification of the global consistence provided by the structural description of the wall document enable to solve the local problems of segmentation, that are often met in the literature.

We have validated our work on an open database. Our first results show that our structural approach really increases the recognition rates of flowcharts, compared to the statistical methods (up to 50% of increase). In a future work, we are planning to mix our structural approach with the results obtained by the statistical methods in order to improve the recognition rate.

References

1. Awal, A.-M., Feng, G., Mouchère, H., Gaudin, C.V.: First Experiments on a new Online Handwritten Flowchart Database. In: Document Recognition and Retrieval XVIII, vol. 01, United States (2011)
2. 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, IJDAR 8(2), 111–122 (2006)
3. Feng, G.H., Viard Gaudin, C., Sun, Z.X.: On-line hand-drawn electric circuit diagram recognition using 2D dynamic programming. Pattern Recognition 42(12), 3215–3223 (2009)
4. Macé, S., Anquetil, E.: Eager interpretation of on-line hand-drawn structured documents: The dali methodology. Pattern Recognition 42(12), 3202–3214 (2009)
5. Qi, Y., Szummer, M., Minka, T.P.: Diagram structure recognition by bayesian conditional random fields. In: CVPR, pp. II:191–II:196 (2005)
6. Tilak, G., Ananthakrishnan, K.: Sketchuml – sketch based approach to class diagrams. In: Proceedings of IUI (2009)
7. Urs, Ramer: An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing 1(3), 244–256 (1972)
8. Yuan, Z., Pan, H., Zhang, L.: A Novel Pen-Based Flowchart Recognition System for Programming Teaching. In: Leung, E.W.C., Wang, F.L., Miao, L., Zhao, J., Kleinberg, R.D. (eds.) WBL 2008. LNCS, vol. 5328, pp. 55–64. Springer, Heidelberg (2008)