Incremental Interpretation of On-Line Hand-Drawn Structured Documents

Sébastien Macé and Eric Anquetil
IRISA - INSA
Campus Universitaire de Beaulieu
35042, Rennes Cedex, France
{sebastien.mace,eric.anquetil}@irisa.fr

Abstract

This paper deals with the incremental interpretation of on-line hand-drawn structured documents. We recall the bases of the CD-CMG visual grammar we have presented previously, but we mainly focus on its associated parser. The formalism context-sensitiveness allows to exploit a parser with predictive capacities which couples bottom-up and top-down strategies. It also helps targeting consistent candidate symbols, which significantly reduces the combinatorial. We use the fuzzy set framework to model the adequacy measure of each possible interpretation and demonstrate how this permits to detect both uncertainty and the need for segmentation. Finally, in order to detect the necessity to wait before making a decision, we exploit partial interpretation evaluation. We present experiments highlighting the quickness of the analysis process, which makes it usable in real-time applications. This technology has been used to design the “Script&Go Electrical Sketches” software that is already on the market.

Keywords: on-line structured document, hand-drawn shapes, incremental interpretation, fuzzy set framework

1. Introduction

This paper deals with the interpretation of on-line hand-drawn structured documents, such as electric diagrams, musical scores or mathematical formulas. This is a complex problem of pattern recognition. Moreover, ink parsing, i.e. the task of grouping strokes constituting a same symbol, must be considered; if no constraint is imposed to the user, one stroke can even contain (part of) several symbols. Finally, a stroke drawn in such a document can have various different interpretations depending on its context (structural and/or temporal) which must therefore be taken into account. As a consequence, structured document interpretation requires complex analysis and decision making processes.

In this article, we focus more specifically on one interpretation paradigm, which is incremental interpretation. As far as structured documents are concerned, incremental interpretation consists in trying to understand the structure of the document as well as its elements during its composition, i.e. after each input stroke. We believe that it should imply a feedback showing to the user how the system has interpreted his strokes; it can be done either directly in the document [3, 6] or in a separated location [5]. This offers many benefits from a human-computer interaction point of view: it allows immediate interactive error correction and implicit validation by the user, who becomes an actor of the analysis process. He also knows how his document has been understood (e.g. he does not have to check all his document at the end of the composition to be sure that it has been correctly interpreted).

In spite of many advantages, incremental interpretation still remains a complex and open problem. On the one hand, the analysis process should be efficient enough to keep the user's pace and not make him wait for a feedback. On the other hand, this process should be able to decide whether or not it has enough information to make a decision: it is sometimes necessary to wait for forthcoming strokes to interpret another one (e.g. with multi-stroke symbols or with one symbol that can be the prefix of another one). Moreover, it is sometimes difficult, or even impossible, to interpret a stroke, because it either corresponds to none of the domain symbols (outlier stroke), or can correspond to more than one (ambiguous stroke).

We have introduced in [6] the DALI methodology which is used in the “Script&Go Electrical Sketches” software that is already on the market (see figure 1, videos are available at [4]). It is based on a context-sensitive visual grammar (extension of Constraint Multiset Grammars [8]) modeling the way structured documents are composed (cf. section 3). This paper aims at highlighting how this knowledge is used to interpret each input stroke as efficiently as possible. The context modeling makes it possible to exploit a parser with predictive capacities which couples bottom-up and top-down strategies. Context also helps targeting candidate symbols, which significantly reduces the combinatorial. We use the fuzzy set framework
to model the adequacy measure of each possible interpretation and demonstrate how this permits to detect both uncertainty and the need for segmentation. Finally, in order to detect the necessity to wait before making a decision, we exploit partial interpretation evaluation.

This paper is organized as follows. Next section gives a brief review of the state of the art in the domain of incremental analysis of on-line documents. Section 3 reminds the bases of the generic method we have designed for the modeling of on-line structured document composition. In the following section, we introduce the general analysis process to incrementally interpret user strokes and focus more specifically on ambiguity management. Section 5 deals with the postponement of the decision making. Then, we present experimental results and conclude.

2. Related works

In this section, we review existing generic approaches for stroke interpretation and ambiguity management in structured document analysis; due to space limitations, we do not focus on domain-specific approaches.

Context-free (or isolated) approaches aim at recognizing the document elements independently one from another [5, 9, 11]. Such approaches do not have the ability to use the context in which a shape appears to help interpreting it. On the contrary, context-sensitive approaches can model elements relatively to others. We believe this is very adapted to the management of highly structured documents, in which symbols have strong links. The approach we present in this paper is context-sensitive. Hammond and Davis propose in [3] a language to describe drawing, display and editing of shapes drawn in the context of documents. They show how this knowledge is used in the recognition process, but do not specify the decision making process in case of ambiguity. Alvarado and Davis extend this approach by coupling it with dynamically constructed bayes networks [1]. They use a coupling of bottom-up and top-down strategies to question stroke interpretation based on its drawing context. Nevertheless, their analysis processing times and recognition rates do not enable using it in real-time applications. We also use a coupling of bottom-up and top-down strategies but exploit both the interaction with the user and the modeling of the context to significantly reduce the combinatorial. Chok and Marriott [2] present an approach based on constraint multiset grammars (which are context-sensitive) and their associated incremental parser but do not focus on hand-drawing interpretation nor ambiguity management. We extend this by enriching the context modeling and taking the imprecision of hand-drawing into account thanks to the fuzzy set framework.

3. Context-Driven CMG

We introduced in [6] a new visual grammar for the modeling of structured document composition, called the Context-Driven Constraint Multiset Grammars (CD-CMG). In this section, we remind the most important concepts of CD-CMG and highlight their last evolutions.

CD-CMG are an extension of Constraint Multiset Grammars (CMG) [8], which are well adapted to the handling of two-dimensional elements: they model the replacement of a multiset of document elements $\beta$ into another one $\alpha$ if some constraints are satisfied.

The originality of CD-CMG is to externalize and enrich contextual knowledge. They model not only the symbols that must already exist in order to recognize another one (classically expressed with the “∃” operator), but also which symbols can exist in consequence of its addition; these informations are formalized respectively as preconditions and postconditions of CD-CMG productions (which is an evolution of the works presented in [6]), and concretely model composition conventions about the document, i.e. how the user will draw it.

A CD-CMG $G$ is a quadruplet $G = (V_N, V_T, P, S)$ where $V_N$ is a set of non-terminal symbols, $V_T$ a set of terminal symbols, $P$ a set of productions and $S$ the starting symbol. A production can have the following form:

$\alpha \rightarrow \beta$

contextual

We use CD-CMG with only one terminal symbol (i.e. one primitive), which is the hand-drawn stroke. This is to avoid a systematic segmentation process which turns out to be a major cause of interpretation errors. We have also noticed that users raise the pen between the major part of the symbols, which does not encourage systematic stroke segmentation. Of course, our choice do not forbid to segment a stroke when necessary (cf. section 4.5).

Figure 2 presents a CD-CMG production dedicated to the interpretation of the Switch electric component. In the next subsections, we present the role of each part of a production and illustrate them thanks to this example. For practical reasons, we first present the role of postconditions, and then the role of preconditions.
3.3. Constraints

Constraints model a local vision of the \( \beta \) elements. It is possible to exploit not only structural recognition (which is well adapted to the representation of the relative arrangement of symbols), but also statistical recognition (which is well adapted to the management of imprecise data such as hand-drawn strokes).

In the switch production, constraints model, on the one hand, that the second stroke should be below the first one, and, on the other, that a classifier able to discriminate between electric components should interpret both of them as a switch.

In the following section, we show how this knowledge is exploited to incrementally interpret the user strokes.

Figure 2. CD-CMG production modeling the interpretation of a switch (a) and examples of hand-drawn and recognized switches (b).

4. Analysis Process

In this section, we focus on the parser associated to CD-CMG and highlight its originalities. We first present the general process and show how it uses the formalized knowledge to both couple bottom-up and top-down strategies and significantly reduce the combinatorial. Then we show the way we exploit the fuzzy set framework to model the adequacy measure of interpretations, discover ambiguities and detect when segmentation is required.

4.1. General process

The CD-CMG parsing process is performed after each input stroke. It is based on the CMG parser [2]: both consist in trying to repeatedly replace multisets of document elements into other multisets thanks to the reduction of productions of which constraints are satisfied. Parsing stops when stability is reached, i.e. when no further reduction can occur. The main originality of our CD-CMG parser is its predictive capacities, based on the document global vision modeling; this permits to combine bottom-up (“guided by the elements”) and top-down (“guided by the contexts”) analysis processes. In particular, CD-CMG consume and produce not only multisets of elements, but also structural contexts.

Parsing CD-CMG is based on three main principles that help reducing the search space. The first one consists in pruning the number of document elements and structural contexts to analyze. We require that when trying to apply new reductions, at least one of the \( \beta \) elements or precondition contexts be a “new” one (i.e. created since the last analysis). Informally, this is because if some reductions could have occurred with only “old” ones, they would have been previously.

The second principle consists in pruning the number of productions and associated \( \beta \) elements and precondition contexts to analyze, by the following process:

- when analyzing a new element (e.g. a stroke), the parser looks for the structural contexts in which it is located;
- when analyzing a new structural context, the parser looks for the elements that are located in it.

In each case, the productions associated to the elements and structural contexts are triggered. All their precondition contexts are used to perform element parsing to find pertinently the other \( \beta \) elements: instead of looking for any document element, we look for document elements in a given location. Thus, only element combinations that are valid from a document point of view are really tested, which significantly reduces the combinatorial. A good consequence is that only contextually consistent recognition systems will be exploited.

The third principle is determinism: once a sequence of productions has been applied, i.e. shown to the user, it can not be questioned later. Indeed, the user is integrated
in the analysis process, by implicitly validating or explicitly rejecting its answers. Thus, the parser must not reconsider productions that have been validated by the user, but should rather lean on them as much as possible to analyze new strokes more efficiently.

Next sections focus on the decision-making process.

4.2. Evaluating a production

Using crisp constraints is not well adapted to the management of imprecise data such as hand-drawn strokes. Therefore, we exploit the fuzzy set framework in addition to CD-CMG to avoid making an “all-or-nothing” decision. We can now see a CD-CMG production as:

\[
\alpha \models_\rho \beta \ldots
\]

where \(\rho \in [0, 1]\) models the adequacy measure for \(\beta\) to be reduced into \(\alpha\), i.e. the degree to which the production is satisfied. It is necessary to define \(\rho\) for each production. Actually, in our work, \(\rho\) is not defined a priori but depends on the involved \(\beta\) elements and structural contexts. As introduced in [6], \(\rho\) is a fuzzy combination of \(\rho_{\text{preconditions}}\) and \(\rho_{\text{constraints}}\), which model respectively the preconditions and the constraint (both structural and statistical) verification degrees. Note that using the fuzzy set theory does not have any impact on the writing of the formalism; nevertheless, it is necessary to define fuzzy constraints.

A production adequacy measure is used to decide if an interpretation is valid: we use a threshold \(T\) below which it is not pertinent to reduce a production (outlier reject [10]). This permits to detect outlier strokes, which are strokes that no production can reduce.

4.3. Evaluating a sequence of productions

As explained previously, if a production can be reduced, we test if others can be in consequence of this. In order to not consider productions on a isolated way, it is necessary to evaluate the compound result of productions sequences. For such purpose, we define \(\rho_{PS}\) as the adequacy measure of a production sequence \(PS\):

\[
\rho_{PS} = \rho_{P_1 \ldots P_n} = t - \text{norm}(\rho_{P_1}, \ldots, \rho_{P_n})
\]

We normalize \(\rho_{PS}\) by the number of productions in the sequence \(\rho_{PS}^{\text{norm}}\), in order not avoid giving an advantage to sequences with fewer productions.

The same way as for production evaluation, a production sequence is considered inconsistent if its adequacy measure is below the threshold \(T\).

4.4. Making a decision

At the end of an analysis process, several production sequences with associated adequacy measures are computed: the one that should be applied is the one with the higher value. Nevertheless, several consistent production sequences can have very close values of \(\rho\), which implies an ambiguity between several interpretations. For that purpose, we exploit the ambiguity reject theory [10] and compare the degrees \(\rho_{PS_1}^{\text{norm}}\) and \(\rho_{PS_2}^{\text{norm}}\) of the two best sequences \(PS_1\) and \(PS_2\), such that \(\rho_{PS_1}^{\text{norm}} \geq \rho_{PS_2}^{\text{norm}} > 0\). Then we define a reliability function \(\psi_{PS_1,PS_2}\) as:

\[
\psi_{PS_1,PS_2} = \frac{\rho_{PS_1}^{\text{norm}} - \rho_{PS_2}^{\text{norm}}}{\rho_{PS_1}^{\text{norm}}}
\]

Finally, we use a threshold for \(\psi_{PS_1,PS_2}\) above which it is not pertinent to make a decision. Note that the detection of such an ambiguity can have various impacts on the interface, depending on the designer’s will.

4.5. Contextual segmentation

As emphasized in section 3, we do not use a systematic segmentation process. Nevertheless, it is too constraining to ask users to always raise the pen between two symbols (especially, in electric diagram composition, when connections are concerned). To deal with this problem, we use contextual segmentation, which consists in trying to segment a stroke only if it makes sense to do so. In the domains that we have worked on, we have found it sufficient to limit segmentation candidates to outlier strokes.

Our segmentation process consists in finding the stroke angles. All the possible combinations are then tested, but only those of which most of the segments are understood are considered as valid. Figure 3 presents a contextual segmentation example. The user draws a stroke which can not be interpreted as a symbol by the analyzer. Our segmenter finds three possible segmentations, one of which leads to the interpretation of three connections.

```
Figure 3. Contextual segmentation: an outlier stroke is interpreted as three connections.
```

5. Postponing the decision-making

One main requirement of incremental analysis is to be able to know if there are enough available information to make a decision. Figure 4 illustrates different situations. In figure 4.a, the user draws a stroke which can not be interpreted as a connection, because it is “too below” the existing one. Nevertheless, it can be the first stroke of a multi-stroke receiver (cf. figure 4.c and 4.d). In figure 4.b,
the user draws a stroke that can be interpreted as a connection. However, it still can be the first stroke of a receiver: a connexion can be the prefix of a receiver. In both situations, it is necessary to wait for forthcoming strokes before making a decision; for that purpose, we propose a solution based on partial production verification, which helps dealing with multi-stroke symbols and symbols that are prefix from others.

5.1. Partial productions

The aim of partial productions is to be able to test a production with only a subset of its elements in order to evaluate if, “so far”, it is satisfied, i.e., if a stroke can be part of a multi-stroke symbol. Works presented in [1] also exploit partial productions. We go further by evaluating their adequacy measure and use it to prune interpretations; this significantly reduces the combinatorial. For that purpose, we evaluate all the structural contexts and constraints which can be without the missing elements. A partial production adequacy measure is computed as the maximum degree that the corresponding complete production could reach, by setting all the not-yet-computable degrees to 1. Note that the choice of the constraints in a production has a major impact on this process: the more numerous and precise they are, the sooner inaccurate productions will be rejected.

Thus, when a stroke is drawn, we not only test the “complete” CD-CMG productions, but also partial multi-stroke productions. If a stroke is the subject of an outlier reject, it can not be interpreted as a symbol nor as the beginning of a symbol (and then the segmentation process will take place). If a complete production is in conflict with some partial productions, it is not yet possible to make a decision (the stroke is either a symbol or the prefix of another one). Figure 5 exemplifies this when a user draws a stroke which can be either a connexion or the beginning of a receiver (more precisely the first or the third $\beta$ element of the production). Next strokes are decisive, because if they can not be interpreted as the continuation of a receiver (i.e. if they can not continue the production by “filling the blanks”), this hypothesis is rejected (our parser imposes to finish one symbol before drawing another); otherwise it is recognized.

5.2. Other properties of partial productions

Partial production evaluation offers many advantages from both pattern recognition and human-computer interaction points of view. Firstly, we prune as soon as possible the possible interpretations. This decreases significantly the combinatorial because, as illustrated in figure 5, only missing $\beta$ elements are tested. Secondly, it allows automatic completion: for instance, although many electric symbols share a subpart of the same shape corresponding to the second stroke (the diagonal and vertical segments, see figure 2.b), the first one is in fact discriminative. We can then choose to automatic replace the stroke by the complete symbol, allowing the user to save time.

6. Evaluation

In this section, we present an evaluation of the analysis process we present in this paper. For that purpose, we used the “Script&Go Electrical Sketches” software [4] that exploits it. We have already presented in [6] some recognition and reject performances of a previous version of the system: we obtained an average of 2.4% error rate and 9.3% reject rate on tests performed over 9 writers. Here we focus on the performances of the analysis process. The same way as Alvarado et al. in [1], we evaluated the time needed to process each input stroke. For that purpose, we just asked 10 users to draw a voluminous electric diagram without giving them more constraints. Figure 9 presents the results (the vertical bars show the standard deviation in processing time). Several reports can be made. Firstly, the
processing time is almost always far less than 1 second, which makes the system usable in real-time applications. Peaks can be observed when segmentation produces numerous combinations. Secondly, the median processing time is quite constant in function of the number of strokes previously drawn, which proves that the methods scales well to complex documents. Note that although our results are far better than the ones in [1], the systems are not the same, because we prune possible interpretations very early and do not reconsider them, whereas these authors can always change a decision made previously.

Thus, many processes can be set up, and for instance Mankoff’s mediation techniques presented in [7]. Nevertheless, in the context of the “Script&Go Electrical Sketches” software, we have made the following choices in consequence of discussions with future users of the system: if the system has enough information to interpret a stroke, it is changed into the corresponding document symbol. If the system is not yet able to make a decision and decides to wait for forthcoming strokes, it remains under its hand-drawn form. On the contrary, a stroke that is rejected is automatically deleted.

Future work will focus on improving the exploitation of this analysis process. So far we impose to finish one symbol before starting another one; using partial productions could make it possible to “pause” a symbol to draw another one and come back to finish it later.

8. Acknowledgment

The authors would like to thank Guy Lorette, Professor at the University de Rennes 1, and everyone who took part in the experiments for their contribution to this work. This work is partially founded by the Brittany Region.

References