Top-Down Online Handwritten Mathematical Expression Parsing with Graph Grammar

Frank Julca-Aguilar[1(✉)], Harold Mouchère[1], Christian Viard-Gaudin[1], and Nina S.T. Hirata[2]

[1] University of Nantes, Nantes, France
faguilar@ime.usp.br
[2] University of São Paulo, São Paulo, Brazil

Abstract. In recognition of online handwritten mathematical expressions, symbol segmentation and classification and recognition of relations among symbols is managed through a parsing technique. Most parsing techniques follow a bottom-up approach and adapt grammars typically used to parse strings. However, in contrast to top-down approaches, pure bottom-up approaches do not exploit grammar information to avoid parsing of invalid subexpressions. Moreover, modeling math expressions by string grammars makes difficult to extend it to include new structures. We propose a new parsing technique that models mathematical expressions as languages generated by graph grammars, and parses expressions following a top-down approach. The method is general in the sense that it can be easily extended to parse other multidimensional languages, as chemical expressions, or diagrams. We evaluate the method using the (publicly available) CROHME-2013 dataset.

Keywords: Mathematical expression recognition · Graph grammar · Top-down parsing · Bottom-up parsing

1 Introduction

An online handwritten mathematical expression consists of a sequence of strokes, usually collected using a touch screen device. For example, in Figure 1, the expression is composed of five strokes, that is $(str_1, \ldots, str_5)$, where str_i is the i^{th} stroke, considering the input order. Recognition of online handwritten mathematical expressions involves three processes: (1) symbol segmentation, (2) symbol classification and (3) structural analysis. The first process groups strokes that form a same symbol; the second identifies which mathematical symbol represents each group of strokes and the third identifies relations between symbols – as the *superscript* relation between symbols "a" and "b" in the expression "$a^b c$".

The recognition process is usually handled by a parsing technique [1,2,8,10]. In these techniques, a grammar defines the mathematical language (valid symbols and structures) to be recognized and a parse algorithm determines the structure of the expression, in accordance with the grammar. Reasons to use

A. Pardo and J. Kittler (Eds.): CIARP 2015, LNCS 9423, pp. 444–451, 2015.
DOI: 10.1007/978-3-319-25751-8_53

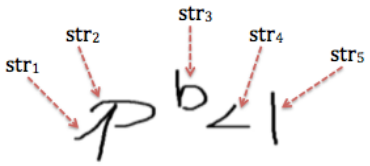

Fig. 1. Online handwritten mathematical expression composed of five strokes: $(str_1, \ldots, str_5)$.

parsing techniques include: (1) they generate an structured result that can be further processed, (2) the grammar represents our understanding or model of the object to be recognized and (3) missing information can be completed using syntactic or contextual information [3]. For example, in an expression "(1+1)", if all symbols but the closing parenthesis are already recognized, a parse algorithm can determine that the missing symbol is actually a closing parenthesis. Contextual information is useful when dealing with handwritten expressions as ambiguous recognition cases can be generated.

Parsing techniques have been successfully applied in recognition of strings, where symbols or words are arranged horizontally (there is only one relation type between symbols). Those approaches generate a parse tree as result; and according to how the parse tree is built, the techniques can be divided into two types: top-down and bottom-up. Top-down techniques determine first high level structures (subexpressions), then low level structures (symbols). The bottom-up approaches perform the inverse process. According to the literature, top-down techniques or bottom-up techniques with a top-down component are needed to build powerful parsers [3]. A main advantage of the top-down components is the fact that it avoids to parse some subexpressions that do not generate valid parsing results [3,5].

Most grammars used to represent mathematical expressions are based on grammars to parse strings [1,5,8,10]. However, as these grammars were originally designed to represent only horizontal relations between symbols, it is difficult to extend the model to represent languages with multiple relation types. About the parse algorithms, most approaches follow a pure bottom-up parsing; for instance, different adaptations of the CYK algorithm can be seen in [1,8,10].

Graph grammars [7] can provide a more natural model to represent mathematical expressions: sentences (mathematical expressions) can be represented as labeled graphs, where vertices represent (terminal) symbols and edges represent relations among symbols. As arbitrary relations can be expressed as edges, the model provides a flexible representation, so that it is easy to extend a particular grammar to define new structures. On the other hand, if no strong constraints are imposed, a similar parsing technique can be used to recognize other multidimensional languages, as handwritten chemical expressions and diagrams. However, the general representation of graph grammars can generate a considerable increase on the computational cost of the parse algorithm. As an example, a tentative to use graph grammars for mathematical expressions recognition can

be found in [2], where the authors proposed a bottom-up parse algorithm; but even with a small grammar, time out failure was a problem.

In this work, we introduce a new parsing technique for recognition of online handwritten mathematical expressions. We propose a graph grammar model to represent mathematical expressions and compare this approach with grammars used in other approaches (Section 2.1). To cope with the complexity problem, we process input strokes so that symbol and relation hypotheses are pre-calculated and used to limit the subexpressions evaluated by the parse algorithm (Section 2.2). The proposed parse algorithm follows a top-down approach and can be extended to parse other multidimensional languages. To evaluate the proposed method, we used the CROHME-2013 dataset [6] (Section 3).

2 Top-Down Graph Grammar Parsing Method

The proposed method consists of three components: a context-free graph grammar, a *symbol hypotheses relational graph* (SHRG) and a top-down parse algorithm. The graph grammar defines the valid mathematical symbols, subexpressions and relations among them. The SHRG defines the groups of strokes that will be evaluated to determine if they can be interpreted as a symbol or subexpression. Finally, the top-down parse algorithm uses the grammar to determine valid interpretations of all subexpressions and symbols given by the SHRG.

2.1 Context-Free Graph Grammar

A graph grammar is defined by a tuple $M = (N, T, I, R)$, where N is a set of non-terminal symbols (or non-terminals); T is a set of terminal symbols (or terminals), such that $N \cap T = \emptyset$; I is an initial or start graph and R is a set of production or rewriting rules [7]. Figure 2 shows a graph grammar example.

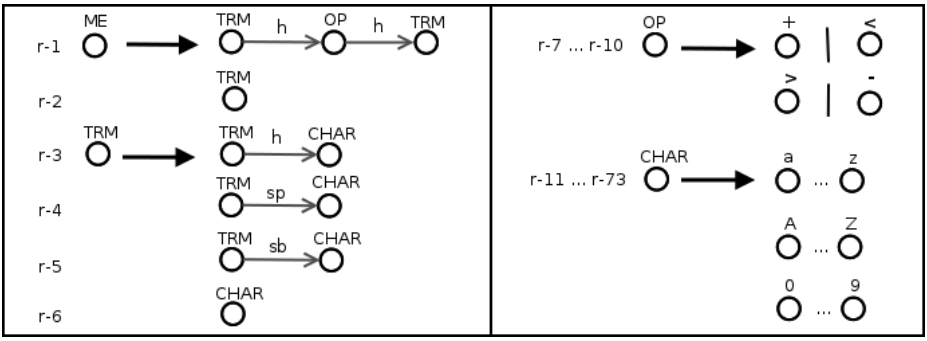

Fig. 2. Graph Grammar example: N={ME, TRM, OP, CHAR}, T = {+,-, <, >, a, ...,z, A, ..., Z, 0, ..., 9}. I corresponds to the left hand side graph of rule r-1 and R = {r-1, ..., r-73}.

We denote a rule as $r = (A, B)$ to indicate a replacement of a graph A by a graph B. In addition, we call A the left hand side (LHS) grammar graph of r ($A = LHS(r)$), and B the right hand side (RHS) graph of r ($B = RHS(r)$). Both, A and B, are digraphs. As it can be seen in Figure 2, vertices and edges of the rule graphs are labeled. For a given rule graph $G = (V, E)$, we define vertex labels by a mapping function α: $V_G \rightarrow N \cup T$, and edge labels by a mapping function β: $E_G \rightarrow SR$, where SR is a set of mathematical relation labels. In Figure 2, SR includes the following labels: "sp" (superscript), "sb" (subscript) and "h" (horizontal).

A context-free graph grammar is a graph grammar such that for each rule (A, B), A is a single vertex graph – as in the case of Figure 2. To clarify the further explanations, we assume that a grammar is defined only with two types of rules: *terminal* and *non-terminal*. In a terminal rule the RHS graph is a single vertex graph , whose vertex is labeled with a terminal symbol – rules from r-7 to r-73 of Figure 2 for instance. A non-terminal rule refers to a rule whose RHS graph contains one or more vertices, labeled only with non-terminal symbols – as r-1 of Figure 2. In addition, we will refer to the non-terminal of the start graph as *start symbol*.

As mentioned above, most approaches for recognition of mathematical expressions extend the grammars used for strings. Figure 3 shows a comparison of a grammar model proposed in [5] and one used in our approach. The grammar proposed in [5] defines production rules of the form: $A \xrightarrow{r} A_1 A_2 \ldots A_k$, where r indicates a relation between adjacent elements in the RHS. As the model defines a unique relation type between consecutive elements, rules that include different relation types must be split into several rules (as shown in Figure 3(a)). Further, when a CYK parse algorithm is used (as in [1,8,10]), the grammar needs to be transformed to a Chomsky Normal Form, which requires the grammar to have no more than two elements the RHS. As a result, grammar rules with more than two elements in the RHS must be split, incrementing the total number of rules. These restrictions make difficult to extend string grammars to model multidimensional languages.

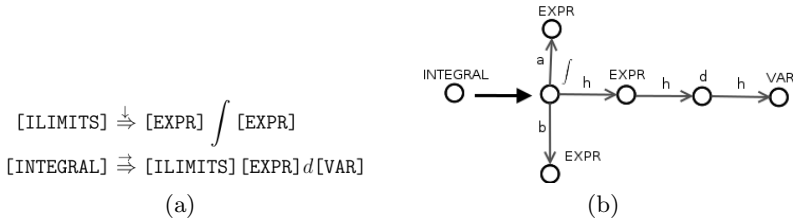

Fig. 3. Integration rule example used in [5] (a) and its corresponding representation using graph grammar (b). "a" and "b" edge labels indicate above and below relations respectively.eps

2.2 Symbol Hypotheses Relational Graph

A symbol hypothesis is a set of strokes that can be interpreted as a terminal symbol. Each symbol hypothesis is assigned a label set defined by a function $\gamma : SH \rightarrow P(L)$, where SH is a set of symbol hypotheses, L is a set of symbol labels, and $P(L)$ is the power set of L.

A Symbol hypothesis relational graph is a digraph (V, E), where V is a set of symbol hypotheses and E is a set of spatial relation hypotheses, defined over pairs of *compatible* symbol hypotheses [1]. As in the case of symbol hypotheses, we define a function $\delta : E \rightarrow P(SR)$, where SR is a set of relation labels (as defined in Section 2.1), and $P(SR)$ is the power set of SR. Figure 4 shows an example of a SHRG.

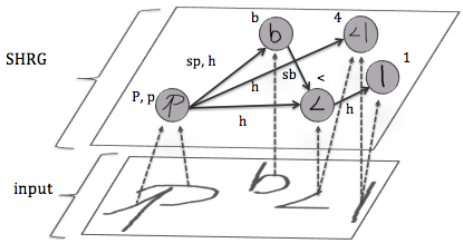

Fig. 4. SHRG example. Edge labels indicate relation types: sp = *superscript*, h = *horizontal*.

To compute symbol hypotheses, we built a 3-nearest neighbor graph from a graph with the bounding box center of the input strokes as vertices and euclidean distances as edge weights. Then, for each stroke, we generate all combinations of the stroke with its neighbors. Each combination defines a symbol hypothesis. For each symbol hypothesis, its corresponding labels are calculated by a neural network classifier that uses shape context and online features, as defined in [4].

To calculate relation hypotheses, we evaluate all pairs of compatible symbol hypotheses and use a neural network classifier, with shape features, to determine the most probable relations. Both symbol and relation classifiers were trained to reject false symbol and relation hypotheses – by including wrong stroke combinations and pairs of symbols in the symbols and relations training sets respectively [4].

It is important to note that, instead of a single label, a symbol and relation hypothesis may have multiple labels. Thus, this configuration keeps several possible interpretations to cope with ambiguous recognition cases. The selection of the most probable labels is based on recognition scores calculated by the classifiers. Given a hypothesis h (symbol or relation) and a set of probable labels $(l_1, \ldots, l_m)$, sorted in descending order by their likelihood score, $score(l_i), for\, i = 1, \ldots, m$,

[1] Two symbol hypotheses sh_i, sh_j are compatible if $sh_i \cap sh_j = \emptyset$.

we select k labels, such that: $\sum_{i=1}^{k} score(l_i) > tr$, where tr is a threshold defined experimentally.

2.3 Top-Down Parse Algorithm

The proposed algorithm considers that the input to the algorithm is a set of strokes, denoted as $str = \{str_1, \ldots, str_n\}$ [2]. Given str, a graph grammar M, and a SHRG G, the algorithm calculates recursive partitions of str according to the rules of M, starting from the rules derived by the start symbol of G, until generating partitions derived by terminal rules. For a given rule r, we consider only partitions of str derived by minors of G[3] that are isomorphic to RHS(r).

Figure 5 illustrates the parsing process of the input expression and SHRG of Figure 4 and graph grammar of Figure 2. The process starts by determining partitions of the complete set of strokes (top of the image), according to the rules derived by the non-terminal **ME** (the start symbol of the grammar). Two partition candidates are found, one using rule r-1 and the other using rule r-2. Each partition is actually a minor graph isomorphic to the RHS graph of the rule that generates the partition. A graph that defines a partition is called **instantiated graph**. The strokes of each vertex of each instantiated graph are further partitioned using the rules derived by their corresponding non-terminals. For example, in the instantiated graph derived by rule r-1, the strokes of the subexpression "p^b" are partitioned according to the rules derived by the non-terminal **TRM**.

As it is shown in Figure 5, for a given set of strokes str and a non-terminal NT, several instantiated graphs may be generated. Those results are recorded in a table denoted as T, where $T(str = \{str_1, \ldots, str_n\}, NT) = \{(g_1, r_1), \ldots, (g_q, r_q)\}$, g_i is an instantiated graph and r_i is the rule that "generated" g_i. The use of the parse table T is usually used in strings parsing [3] to calculate parse results only once (memoization).

The partial results recorded in table T define a parse forest– set of different interpretations of the input. A tree of the parse forest represents a particular interpretation of the input and is defined by a sequence of partitions calculated from the start symbol to terminals. For instance, Figure 5 shows a total of 8 interpretations or parse trees and one of those may be composed by the partitions indicated with red arrows. The tree corresponds to the interpretation "$P^b 4$".

Once the parse forest is built, the final step consists on extracting a tree that better represents the input, according to a given measure. To do that, we defined a ranking function $p : t \rightarrow \mathbb{R}$, where t is a parse tree. Currently, we calculate

[2] We denote a set as a braced list of elements, that is $set = \{element_1, \ldots, element_n\}$ and a sequence (ordered set) as a bracketed list of elements, that is $sequence = (element_1, \ldots, element_n)$, where $element_i$ is the i^{th} element considering a particular order.

[3] A graph H is a *minor* of a graph G if H can be obtained from a **subgraph** of G by contracting edges [9].

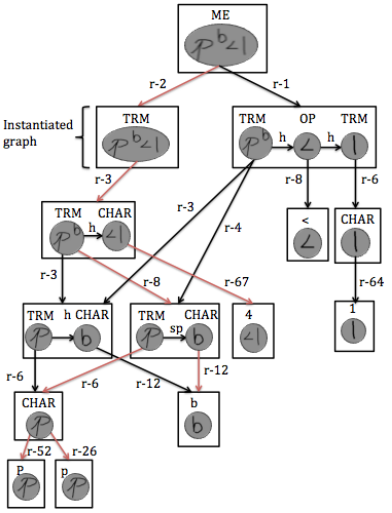

Fig. 5. Recursive partitions calculated by the top-down parse algorithm.eps

p(t) with the geometric mean of the relations and classification scores given by the symbol and relations classifiers to the partitions that compose t.

3 Experimentation

We evaluated the proposed methods using the CROHME 2013 competition dataset [6]. The dataset includes 101 symbol classes and 6 relation types: superscript, subscript, horizontal, above (for example, between a fraction bar and its numerator), below (fraction bar with denominator), and inside (between the radical symbol and its radicand). Using the training part of the dataset, the threshold tr, was fixed in 0.98 for the symbol classifier and 0.95 for the relations classifier.

Table 1 shows our results compared to those of the two best systems (out of five) that used only the CROHME dataset for training (the systems are identified by numbers, as in [6]). The used metrics are: recognition rate (percentage of mathematical expressions correctly recognized), symbol segmentation and symbol and relation recognition rates. The tree rel. metric measures the percentage

Table 1. Comparison of our method with results of CROHME-2013 competition

	recognition	segmentation		classification		tree rel.	
System	rate	recall	precision	recall	precision	recall	precision
IV	23.40	84.97	87.4	73.94	75.77	49.73	51.48
II	19.97	80.70	86.35	66.41	71.06	22.44	27.00
ours	**21.61**	**75.70**	**83.41**	**62.63**	**69.00**	**44.67**	**49.73**

of pairs of correctly segmented and classified symbols from the total number of relations between adjacent symbols. While our system obtained comparable results in terms of recognition rate, it obtained better results in terms of relations detection than in symbols detection.

4 Conclusions and Further Work

In this paper, we describe a new parsing technique for recognition of online handwritten mathematical expressions. The proposed method provides comparable results to those of the best systems of the CROHME-2013 competition. Results show that the use of symbol and relation hypotheses to define valid subexpressions is an effective method to reduce the parsing complexity. In addition, the graph grammar modeling of the proposed method provides a general framework to parse other multidimensional languages, as chemical expressions, or diagrams.

Future work includes the optimization of the symbol and relation classification modules. New features should be explored and evaluated in the context of the complete system performance.

References

1. Álvaro, F., Sánchez, J.A., Benedí, J.M.: Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models. Pattern Recognition Letters (2012)
2. Celik, M., Yanikoglu, B.: Probabilistic mathematical formula recognition using a 2d context-free graph grammar. In: 2011 International Conference on Document Analysis and Recognition (ICDAR), pp. 161–166, September 2011
3. Grune, D., Jacobs, C.J.H.: Parsing Techniques: A Practical Guide, 2nd edn. Springer (2008)
4. Julca-Aguilar, F., Viard-Gaudin, C., Mouchère, H., Medjkoune, S., Hirata, N.: Mathematical symbol hypothesis recognition with rejection option. In: 14th International Conference on Frontiers in Handwriting Recognition (2014)
5. MacLean, S., Labahn, G.: A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets. International Journal on Document Analysis and Recognition (IJDAR) **16**(2), 139–163 (2013)
6. Mouchère, H., Viard-Gaudin, C., Garain, U., Kim, D.H., Kim, J.H., Zanibbi, R.: Icdar 2013 crohme: Competition on recognition of online handwritten mathematical expressions @ONLINE, April 2013
7. Pflatz, J., Rosenfeld, A.: Web grammars. In: Proc. First International Joint Conference on Artificial Intelligence, pp. 193–220 (1969)
8. Simistira, F., Katsouros, V., Carayannis, G.: Recognition of online handwritten mathematical formulas using probabilistic SVMs and stochastic context free grammars. Pattern Recognition Letters **53**, 85–92 (2015)
9. Wagner, K.: Über eine eigenschaft der ebenen komplexe. Mathematische Annalen **114**(1), 570–590 (1937). http://dx.doi.org/10.1007/BF01594196
10. Yamamoto, R., Sako, S., Nishimoto, T., Sagayama, S.: On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. In: International Workshop on Frontiers in Handwriting Recognition (2006)