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

© Springer International Publishing Switzerland 2015

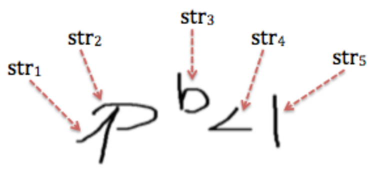

Fig. 1. Online handwritten mathematical expression composed of five strokes: $(str_1, \ldots, str_5)$.

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