ANALYSIS OF EQUATION STRUCTURE USING LEAST COST PARSING

R. Nigel Horspool **John Aycock**

Department of Computer Science, University of Victoria
P.O. Box 3055, Victoria, BC, Canada V8W 3P6

{nigelh,aycock}@csr.uvic.ca

Abstract

Mathematical equations in LaTeX are composed with tags that express formatting as opposed to structure. For conversion from LaTeX to other word-processing systems, the structure of each equation must be inferred. We show how a form of least cost parsing used with a very general and ambiguous grammar may be used to select an appropriate structure for a LaTeX equation. MathML provides another application for the same technology; it has two alternative tagging schemes – presentation tags to specify formatting and content tags to specify structure. While conversion from content tagging to presentation tagging is straightforward, the converse is not. Our implementation of least cost parsing is based on Earley's algorithm.

1 Introduction

LaTeX [1] is a widely used system for typesetting where the user embeds markup tags in a text document to control the formatting. Some tags are *structure tags* because they denote the logical structure of some part of the document (e.g. \section or \footnote are structure tags). Others are *presentation tags* because they specify formatting without necessarily imparting any information about the document structure. For example, \it and \tiny are presentation tags. HTML is another example of a markup language where structure tags (e.g. <ul> and <h2>) can be mixed with presentation tags (e.g. <font> and <center>) in the same document.

For reformatting a document according to new style guidelines or for importing into some word processing systems, it is necessary to know the full structure of the document. Conversion of a LaTeX document into an Adobe FrameMaker document is an example where full structural information is needed. We have developed such a conversion tool, a FrameMaker import filter named *laimport*. However, conversion of equations in LaTeX documents has been problematic and motivated the research reported here. The research also has applications to MathML [2] where both presentation and content tagging schemes are provided.

2 A 'Cost-Based' Grammar for LaTeX Equation Notation

The grammar must be able to accept any sequence of mathematical symbols, even nonsensical ones. But when a conventional structure exists, it should be inferred. We accomplish this goal by attaching cost expressions to grammar rules, where high costs are associated with rules when parentheses do not balance ,operator symbols lack operands, and so on. The grammar also takes account of hints in the form of spacing – operators surrounded by white spaceare assumed to have lower precedence than operators with less surrounding white space. A grammar for a subset of LaTeX equations is shown in Figure 1. In this grammar, '~' represents an inserted space. Each non-terminal has a synthesized cost attribute, computed by the cost expression attached to the grammar rule. These costs increase if parentheses do not balance or if '*' is used with lower precedence than '+'.

With the example grammar, the LaTeX equation \$a+b~*~c\$ has parses equivalent to \${a+{b~*~c}}\$ with cost 14780 and \${{a+b}~*~c}\$ with cost 9873.5. The whitespace has overridden the usual operator precedences, as is desirable when handling operators whose meanings are unknown to the translator. Similarly, the grammar provides a parse for \$((a\$ while rejecting a parse of \$(a+b)\$ as equivalent to \${(a}+{b)}\$.

S	→	E	%cost{$1}
E	→	E OP E	%cost{$2*($1 + 1.5*$3)}
E	→	(E)	%cost{$2}
E	→	(E	%cost{$2*2.0}
E	→	E)	%cost{$1*2.0}
E	→	symbol	%cost{1.0}
E	→	OP	%cost(200.0)
OP	→	~ OP	%cost{0.7*$2}
OP	→	OP ~	%cost{0.7*$1}
OP	→	+	%cost{80.0}
OP	→	*	%cost{100.0}

Figure 1: A Partial Grammar with Cost Expressions

3 Extending Earley's Parsing Algorithm

The problem of finding a least cost parse is similar to that of finding the most probable parse. Our implementation is based upon Stolcke's extension of Earley's algorithm to probabilistic parsing [3]. It is necessary to restrict our cost expressions to be non-negative monotonically non-decreasing functions of their inputs; this permits a pruning strategy that removes items from the parser's states.

The essence of the parsing method is to extend each item in an Earley parser state with two extra components.An item, therefore, has this structure:

$[\ A \rightarrow \alpha \bullet \beta\ ;\ @n\ ;\ \overline{C}\ ;\ @r\ ]$

The *n* component refers back to the ancestor state where the parser started matching the rule $A \rightarrow \alpha\beta$. The $\overline{C}$ component is a vector of costs, one per symbol in α. The *r* component is called the *least cost predecessor*. It records the identity of a predecessor item *in the same state* which generated this particular item.

The completer step of the parser is responsible for computing the cost of the completed rule based on the input values in $\overline{C}$. It must then consider whether to add a new item to the current state where the marker has been advanced past the non-terminal A. If a similar item (one with the same rule, same dot position and same ancestor state) already exists in the current state, the monotonicity property of the ancestor rule's cost expression can frequently be used to choose only one of the two similar items to retain. The least cost predecessor in the new item is linked to the item containing the completed rule, and may be used to construct the least cost parse after all the input has been processed.

The worst-case time complexity of the parsing method is exponential. We note, however, that if the grammar can be transformed so that each righthand side contains no more than two non-terminals, the pruning strategy would be totally effective and the parsing efficiency will be the same as Earley's method. Such a transformation would come at the cost of reducing the expressivity of the cost formulae however.

Acknowledgements

Financial support was received from the Natural Sciences and Engineering Research Council of Canada.

References

[1] L. Lamport. *LATEX: A Document Preparation System*. 2nd Edition. Addison-Wesley, 1994.

[2] Mathematical Markup Language (MathML™) 1.01 Specification, 7 July 1999. URL: `http://www.w3.org/TR/REC-MathML`.

[3] A. Stolcke. An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities. *Computational Linguistics* 21, 2 (1995), pp. 165-201.