Speech Rule Engine: Semantic Tree Grammar

Volker Sorge

July 23, 2022

Sections 1, 2, 3 presents the types, roles and fonts used in the semantic representation. Types are used as tags in corresponding XML representation, while roles and fonts are attributes.

1 Types

Types are immutable. Their idea is to capture the basic nature of a symbol.

1.1 Primitive Types

They are assigned by default to single symbols (or combination in the case of numbers, functions, units) and never change.

`punctuation`	Punctuation like comma, dot, ellipses.
`fence`	Fence symbol.
`number`	One or several digits, plus some punctuation.
`identifier`	Single or multiple letters.
`text`	Regular text in a math expression.
`operator`	e.g. $+, *$.
`relation`	Relation symbol, e.g. equals.
`largeop`	e.g. Sum, product, integral.
`function`	Some named function.

1.2 Compound Types

They are computed once and then never change.

Compound Symbols	
`accent`	Accented symbols (*Deprecated?*)
`fenced`	Fenced expression
`fraction`	Fractions
`punctuated`	List of punctuated elements
Relations	
`relseq`	Relation sequence of a single relation.
`multirel`	Relation sequence containing at least two different relations.
Operations	
`infixop`	Infix operator like $+, \cdot$.
`prefixop`	Prefix operator like $f(x), \sin(x)$ etc.
`postfixop`	Postfix operator like `x++, y--`

Function and big operator applications

`appl`	General function application
`integral`	Integral expression
`bigop`	Big operator expression such as a sum or product.
`sqrt`	Square root expression, i.e., surds without argument or argument = 2
`root`	Root expression, i.e., surds with argument ≠ 2

Big operators or functions with limits or indices

`limupper`	Large operators or limit functions with upper limit expressions. Note the difference to the *script* and *score* types.
`limlower`	Large operators or limit functions with lower limit.
`limboth`	Large operators or limit functions with upper and lower limit.
`subscript`	Subscript expression; can have role subsup, meaning that it originated from an msubsup.
`superscript`	Superscript expression.
`underscore`	Stacked expression with underscript.
`overscore`	Stacked expression with overscript.
`tensor`	Tensor with four indices. At least one left index is present otherwise it would be sub-superscript expression.

Tables and their elements

`table`	A layout element with multiple columns and one or multipled rows. It contains rows and cells.
`multiline`	A layout element with one or multiple rows, where each row contains at most one element. It contains lines as children.
`matrix`	Fenced element with multiple columns and one or more rows. Contains rows as children and the fences as content.
`vector`	Fenced element with single column and one or more rows. Contains lines as children and the fences as content.
`cases`	A layout element starting with a single open fence. It can contain either rows and cells or lines as children. Contains the opening fence as content.
`row`	Contain cells as children.
`cell`	Represent the column element. Are always children or rows.
`line`	Lines are effectively single cell rows.

Enclosed (counterpart for menclosed)

`enclose`	Enclosed (counterpart for menclosed), its role is the type of enclosure. This is the only element that has not a standard role!

General

`unknown`	Unknown expression or symbol.
`empty`	Empty element.

2 Roles

Roles are mutable. They describe the role of a symbol in the context of the formula. Initially a symbol is assigned a default role, which can change during the course of the semantic interpretation. Therefore some roles simply mirror the type, usually until something more specific is known about the role of this particular symbol. As example consider f in the expression $f(x)$. It gets assigned type `identifier` and role `latinletter`. However, the role will eventually change to `prefix function`.

2.1 Symbol Roles

Type	Role	Meaning
punctuation	comma	comma characters
	dash	dash characters of differing length.
	ellipsis	unicode ellipses characters. Does not include sequences of separate full stops.
	fullstop	single period characters.
	prime	prime characters (includes multiple primes)
	openfence	an open fence, which is not used as a fence, i.e., it is solitary and has no counterpart. It is thus considered a punctuation element.
	closefence	ditto for closed fence.
	vbar	ditto for neutral fence.
	dummy	usage of invisible comma as the dummy separator for text.
	application	usage of unicode function application symbol.
	unknown	Punctuation element with unknown role.
fence	open	opening fence.
	close	closing fence.
	top	top fence (e.g., overbrace).
	bottom	bottom fence (e.g., underbrace).
	neutral	neutral fence (vertical bar, double bar, etc.)
identifier	latinletter	Latin character.
	greekletter	Greek character.
	otherletter	Character from some other alphabet. Currently Hebrew.
	unit	A unit name. Normally comes from `MathML-Unit` class attribute.
	unknown	a designated identifier (`mi`) that we could not be classified any further. Usually multi character identifier.
text	text	regular text.
	string	text has been identified as string. Usually comes from `ms` elements.
number	integer	exclusively numerical.
	float	numerical with punctuation.
	othernumber	other numbers, that might contain alpha characters, etc.
	mixed	An integer with a vulgar fraction and an implicit multiplication between the two. Note that this is actually a compound element with two children: a number with role integer and a fraction with role vulgar.
	latinletter	single character that has been explicitly designated as number by `mn`.
	greekletter	ditto.
	otherletter	ditto.
operator	addition	Addition symbol.
	multiplication	Multiplication symbol.
	subtraction	Subtraction symbol.
	division	Division symbol.
relation	equality	Equality symbol. Also equivalence, etc.
	inequality	Inequality symbol.
	arrow	Arrow symbol.
	unknown	Unknown relation.
largeop	sum	Large, sum-like operators. E.g. sum, product, co-product, multi-conjunctions.
	integral	Integral symbols.
function	limfunc	Limit functions.
	prefixfunc	Prefix functions like sin, cos, etc.
	simplefunc	A simple, onle letter function.

2.2 Roles for Compound Types

Type	Role	Meaning
punctuated	sequence	A sequence of punctuated elements. I.e., at least two non punctuation elements separated a punctuation element. It can have punctations at start or end.
	startpunct	Element with a single punctuation as the front.
	endpunct	Element with a single punctuation as the end. Note, that for elements with punctuations at front and end we get a sequence.
	text	Elements separated by one or more dummy punctuation (invisible comma).
fenced	leftright	Elements with fences at start and end.
	abovebelow	Elements with fences above and below. (*Deprecated?*)
fraction	vulgar	fraction of two integers.
	division	any other fraction.
prefixop	negative	Prefix operation with subtraction operation.
	multiop	Prefix operation with multiple operands.
postfixop	multiop	Postfix operation with multiple operands.
infixop	implicit	Infix operation with an inferred multiplication. Note if an invisible times is explicitly given, inheritance will take over. See below.

Role Inheritance from Content Elements In addition, certain compound elements can inherit roles from content elements.

Type	Roles of type	Remark
punctuated	punctuation	If element is made up exclusively of a single punctuation. I.e., no other interspersed elements.
infixop	operation	
prefixop	operation	
postfixop	operation	
relseq	relation	If all content elements are the same.
bigop	largeop	
integral	largeop	

2.3 Roles for Tables, Vectors, Matrices

Unless some special case can be identified the role will be unknown. Special roles are currently only computed for `vector` and `matrix` types.

Type	Role	Meaning
vector	binomial	Vector with exactly two line elements.
	squarematrix	Vector with exactly one element.
	determinant	Vector with exactly one element and neutral fences.
	unknown	Vector that could not be further determined.
matrix	rowvector	A single row with multiple cells. Normally a vector has line elements, this one has rows and cells, hence is classified as a matrix.
	squarematrix	Matrix with the number of rows and columns.
	determinant	Square matrix with neutral fences.
	unknown	Vector that could not be further determined.
table	unknown	
multiline	unknown	
cases	unknown	

Observe that both vectors and matrices can be determinants or square matrices. The type then determines what components they contain, either lines or rows and cells.

In general components of tabular expressions, i.e., lines, rows, cells, get the same role as the type of the expression. In the particular case, where a tabular element has a specialist role, they inherit that role instead.

Type	Role	Meaning
line	vector	Line in some vector.
	binomial	Line in a binomial vector.
	squarematrix	Line in one by one vector.
	determinant	Line in one by one determinant.
	multiline	Line in a multiline equation.
	cases	Line in a case statement.
row	matrix	Row in some matrix.
	rowvector	Row in a row vector. That means it will only contain one cell.
	squarematrix	Row in a square matrix.
	determinant	Row in a determinant.
	multiline	Row in a multiline statement.
	cases	Row in a case statement.
cell	matrix	Cell in some matrix.
	rowvector	Cell in a row vector.
	squarematrix	Cell in a square matrix.
	determinant	Cell in a determinant.
	multiline	Cell in a multiline statement.
	cases	Cell in a case statement.

2.4 Roles for Sub-elements of Particular Types

These are roles that indicate that an element belongs to a particular parent element. They can be assigned to an arbitrary child or to special type of children.

Parent Type	Role	Child#	Meaning
tensor	leftsub	2	Left subscript of a tensor.
	leftsuper	3	Left superscript of a tensor.
	rightsub	4	Right subscript of a tensor.
	rightsuper	5	Right superscript of a tensor.

Note that these roles are given to the element in the respective slot, regardless of the nature of that expression. Also note, that sequences of indices are modelled as a `punctuated sequence` with `dummy punctuation` element.

Parent Type	Role	Child#	Meaning
overscore	overaccent	2	Characters that can be considered an accent.
underscore	underaccent	2	Characters that can be considered an accent.

2.5 Role Inheritance from Children

Roles can be inherited from children to propagate semantic meaning of compound elements. Those roles are therefore not restricted to the roles of a particular type but only depend on the role of child at a particular position.

Type	Child#	Meaning
limlower	1	Role of inner operator or function.
limupper	1	Role of inner operator or function.
limboth	1	Role of inner operator or function.
subscript	1	Role of base element.
superscript	1	Role of base element.
overscore	1	Role of base element.
underscore	1	Role of base element.
tensor	1	Role of base element.

2.6 Specialist roles

subsup: Assigned to elements of type `superscript`, if they were generated from a `msubsup` element. This indicates that the first child is of type `subscript`.

unit: In addition to being the role of an identifier, `unit` can be propagated to more complex structures. In particular, to types `subscript, superscript, fraction` and `infixop` with roles `multiplication, implicit`.

3 Fonts

Some symbols also have a font element attached. Font elements can be extended to entire expressions if they all share a common font. Font values are:

`bold`	`double-struck`	`monospace`	`sans-serif-italic`
`bold-fraktur`	`double-struck-italic`	`normal`	`sans-serif-bold`
`bold-italic`	`fraktur`	`script`	`sans-serif-bold-italic`
`bold-script`	`italic`	`sans-serif`	`unknown`

4 Branching Nodes Overview

Branch nodes have both children and content.

Node types containing no content are:

root, sqrt, table, row, cell, enclose, number (with role mixed), subscript, superscript, underscore, overscore, bigop, integral, fraction, tensor

Node types containing content:

infix, prefix, postfix, relseq, multirel, fenced, punctuated, appl

4.1 Nodes without content

These are usually quite straightforward and often mirror the corresponding MathML element. There are some exception however:

Number only with role mixed, where the two children are an integer and a vulgar fraction.

Bigop consists of big operator and it's arguments. E.g. $\sum f(x) + b$ would yield a big operator with children $\sum, f(x)$. $+b$ would be considered not to be part of the sum.

Integral always has three children: Integral symbol, Integrand and Integral variable. Note that the last two can be both empty. This allows for $\int$, $\int dx$, and $\int f$.

Tensor is similar to mmultiscripts but contains always all four indices. Some might of course be empty.

4.2 Nodes with content

Type	Content	Children	Mixed Element	Example		
infixop	Operators	Operands	unique operator	$a+b+c$	$\longrightarrow$	[+, +][a, b, c]
prefixop	Operators	Operand	concatenated ops	$++a$	$\longrightarrow$	[+, +][a]“++”
postfixop	Operators	Operand	concatenated ops	$a--$	$\longrightarrow$	[-, -][a]“–”
relseq	Relations	Operands	unique relation	$a=b=c$	$\longrightarrow$	[=, =][a,b,c]“=
multirel	Relations	Operands	None	$a=b<c$	$\longrightarrow$	[=, <][a,b,c]
fenced	Fences	Content	None	$(a+c)$	$\longrightarrow$	[(,)][a+c]
punctuated	Punctuations	Full content	None	$a;b;c$	$\longrightarrow$	[;,;][a,b,c]
appl	Appl function, function symbol	Function, arguments	None	$f(x)$	$\longrightarrow$	[@, f][f, (x)]
bigop	large op symbol	operator, arguments	None	$\sum_{i=0} ni$	$\longrightarrow$	[$\sum$, f][$\sum_{i=0} n$,
integral	integral symbol	integral, integrand, variable	None	$\int_0 nx\,dx$	$\longrightarrow$	[($\int$, f][$\int_0 n$, x,
matrix	Fences	Table	None	(a)	$\longrightarrow$	[(,)][a]
vector	Fences	Lines	None	(a)	$\longrightarrow$	[(,)][a]
cases	Opening Fence	Lines or Table	None	$\{a$	$\longrightarrow$	[{][a]

Interesting special cases are implicit infix operators, separated text, and function applications as they introduce elements that do not exist in the original MathML expression. The latter is already given in the table above. The former two correspond to the following two cases:

1. A sequence of separated identifiers is translated into an infixop with role implicit, where the operator is the invisible times. The operator is added as the mixed element but only once to the content. Moreover, it does not correspond to an existing MathML element.

2. A sequence dominated by mtext elements is translated into a punctuated list with role text, where the punctuation is the invisible comma. Similar to the previous case the invisible comma is added as mixed element but only once to the content.

4.3 Types, Children and Content

An overview of arity and meaning of children and content.

	Children		Content	
Compound Symbols				
`accent`	2	letter, accent		
`fenced`	1	fenced expression	2	opening fence, closing fence
`fraction`	2	denominator, enumerator		
`punctuated`	n	expression including puncutation	m	contained punctuation elements
Relations				
`relseq`	n	operands	$n-1$	relation symbols
`multirel`	n	operands	$n-1$	relation symbols
Operations				
`infixop`	n	operands	$n-1$	operators
`prefixop`	1	operand	n	prefix operators
`postfixop`	1	operand	n	postfix operators

	Children		Content	
Function and big operator applications				
`appl`	2	function, application	2	invisible application, function symbol
`integral`	3	integral, integrand, integration var	1	integration symbol
`bigop`	2	operator, operand	1	large operator symbol
`sqrt`	1	content		
`root`	2	arity, content		
Big operators or functions with limits or indices				
`limupper`	2	center, upper		
`limlower`	2	center, lower		
`limboth`	3	center, lower, upper (check this!)		
`subscript`	2	base, sub		
`superscript`	2	base, super		
`underscore`	2	base, under		
`overscore`	2	base, over		
`tensor`	5	base, left sub, left super, right sub, right super		
Tables and their elements				
`table`	n	rows of type `row`		
`multiline`	n	lines of type `line`		
`matrix`	n	rows of type `row`	2	left fence, right fence
`vector`	n	lines of type `line`	2	left fence, right fence
`cases`	n	lines/rows of type `line/row`	1	left fence
`row`	n	cells of type `cell`		
`cell`	1	cell content		
`line`	1	line content		
Enclosed (counterpart for menclosed)				
`enclose`	1	enclosed expression		
General				
`unknown`	1	whatever we could not interpret/process		
`empty`	0			

Note, this is an evolving document. At the time of reading this grammar is probably incomplete.