

System Description: A Context-Free Parser of Math Expressions

Florian Rabe

University Erlangen-Nuremberg

September 2026

Motivation

Context

UniFormal: recently developed new language

- ▶ theories+logic+computation+...
- ▶ subsume MMT $\rightarrow$ MMT 2.0?
- ▶ details still in flux
- ▶ this paper: expression parser component of UniFormal
notations, type-based disambiguation, ...

Fun but not obviously publishable

- ▶ is there anything new about a parser for math expressions?
- ▶ not much, but a few technical ideas
- ▶ reviewer: “not deeply theoretical, more like a hacker’s document”
indeed

Example

Typical phased design of formula interpretation

- ▶ lexer: identifiers (e.g., alphanumeric) and operators (e.g., symbolic)
- ▶ parser
 - ▶ spot and reduce notations
 - ▶ disambiguate using precedences or similar
- ▶ type-checker/elaborator
 - ▶ resolve identifiers
 - ▶ fill in missing parts
 - ▶ apply coercions
 - ▶ disambiguate using types, type classes, ...
 - ▶ discharge proof obligations

critical: every phase contributes to context
but already should have access to whole context

e.g., $\forall L : \text{Semilattice}, x. x \circ x = x$

note: `Semilattice` might be defined anywhere in the library!

Context-Free Parsing

Design

- ▶ split into two phases
 - ▶ context-free parsing
 - ▶ context-sensitive elaboration
- ▶ Goal: do as much as possible in the parsing phase
research question: how much?

Applications

- ▶ IDEs
 - ▶ parse on every keystroke
 - ▶ support even in ill-typed files
- ▶ Casual browsing of formal libraries e.g., on github
- ▶ LLMs reading/writing math
need to experiment with different treatment of contexts

Leveraging Unicode

Unicode Categories

Bad news and good news

- ▶ bad: context-free = not much more available than Unicode
- ▶ good: nowadays easy to enter in IDEs
 - especially when LLM-generated
- ▶ good: Unicode categories go a long way

categories	size	includes
... Letter (4x)	$\approx 131k$	letters etc.
... Symbol (4x)	$\approx 8k$	mathematical symbols
... Number (3x)	$\approx 2k$	digits
Open/Close Punctuation	75/73	brackets
Connector Punctuation	10	underscore
... Punctuation (4x)	640	other punctuation
... Separator (3x)	19	whitespace
... Mark (3x)	$\approx 3k$	diacritics

Identifiers vs. Operators

Basic idea

- ▶ identifier: letters, numbers, marks
 x' (using U+02B9 for prime), Σ via U+03A3, ...
- ▶ operators: symbols, punctuation
 $\oplus$, Σ via U+2211, ...

Identifiers vs. Operators

Basic idea

- ▶ identifier: letters, numbers, marks
 x' (using U+02B9 for prime), Σ via U+03A3, ...
- ▶ operators: symbols, punctuation
 $\oplus$, Σ via U+2211, ...

Exceptions

- ▶ exception: connector continues token
 - ▶ identifier: *commutative*_+
 - ▶ operator: +_1
- ▶ script change ends token
 - ▶ two identifiers: λx , $\sin \alpha$, $\mathcal{P}\mathbb{N}$
 ad hoc because Unicode 'script' property insufficient
- ▶ symbolic ASCII strings also operators
 $\leq$, $\neq$, ...
 sacrifice: space needed in $x = _ - y$

Nice surprise: Brackets

Unicode handles brackets once and for all

- ▶ 63 pairs of characters officially defined as brackets $\lceil$, $\lfloor$, $\lceil$, $\lfloor$, ...
ought to be enough for everybody
- ▶ to make sure: additionally allow brackets with Modifier Symbols
($\sim$, ...)

Trade-off: no support for

- ▶ unmirrored brackets: $|x|$ or $\|x\|$
- ▶ asymmetric brackets: $|\varphi\rangle$ or $]5; 7]$
- ▶ ASCII art brackets: $/ * \dots * /$
worth sacrificing in favor of context-free bracket parsing

Disappointment: Binders

Unicode does **not** handle binders well

- ▶ quasi-standard (judging by Unicode character names) **insufficient**
 - ▶ 17 NARY operators $\cup, \wedge, \dots$
 - ▶ 25 integral variants
 - ▶ 3 quantifiers $\forall, \exists, \nexists$
 - ▶ no λ (6 letters but no symbol)
- ▶ legacy binder notations very diverse $!, [], \%, \backslash, \rightarrow, \text{fun}, \dots$

Disappointment: Binders

Unicode does **not** handle binders well

- ▶ quasi-standard (judging by Unicode character names) **insufficient**
 - ▶ 17 NARY operators $\cup, \wedge, \dots$
 - ▶ 25 integral variants
 - ▶ 3 quantifiers $\forall, \exists, \nexists$
 - ▶ no λ (6 letters but no symbol)
- ▶ legacy binder notations very diverse $!, [], \%, \backslash, \rightarrow, \text{fun}, \dots$

Trade-off: **parsed with short lookahead**

- ▶ operator $\circ$ is binder if used as $\circ x : A.E$ or $\circ x.E$
- ▶ syntactic sugar for using higher-order functions as binders
e.g., identifier λ takes function $x.t$

Notations

Notation Patterns instead of General Mixfix Notations

identifier f with		
notation	type	enabled notation
nullfix $\circ$	a	$\circ$
prefix $\circ$	$a \rightarrow b$	$\circ e$
postfix $\circ$	$a \rightarrow b$	$e \circ$
infix $\circ$	$(a, b) \rightarrow c$	$d \circ e$
infixleft $\circ$	$(a, b) \rightarrow a$	$e_1 \circ \dots \circ e_n$
infixright $\circ$	$(a, b) \rightarrow b$	$e_1 \circ \dots \circ e_n$
infixassoc $\circ$	$\text{list } a \rightarrow a$	$e_1 \circ \dots \circ e_n$
bindfix $\circ$	$(a \rightarrow b) \rightarrow c$	$\circ x : A.e$
bindfixassoc $\circ$	$(a \rightarrow b) \rightarrow c$	$\circ x_1 : A_1, \dots, x_n : A_n.e$
circumfix $\langle$	$(a_1, \dots, a_n) \rightarrow b$	$\langle e_1, \dots, e_n \rangle$
circumfixflex $\langle$	$\text{list } a \rightarrow b$	$\langle e_1, \dots, e_n \rangle$
applyfix $\langle$	$(c, a_1, \dots, a_n) \rightarrow b$	$e \langle e_1, \dots, e_n \rangle$
applyfixflex $\langle$	$(c, \text{list } a) \rightarrow b$	$e \langle e_1, \dots, e_n \rangle$

notations recognized during parsing

resolved using context and type inference during checking

Parsing Conflicts

Fixity conflicts

- ▶ prefix-prefix vs. nullfix-infix: $\Box \neg x$ vs. $\emptyset \cup x$
- ▶ postfix-postfix vs. infix-nullfix: $x!*$ vs. $x \cup \emptyset$
- ▶ postfix vs. infix: $x!$ and stop parsing vs. $x+$ and keep parsing
- ▶ precedence conflict: $x + y \cdot z$

Parsing Conflicts

Fixity conflicts

- ▶ prefix-prefix vs. nullfix-infix: $\Box \neg x$ vs. $\emptyset \cup x$
- ▶ postfix-postfix vs. infix-nullfix: $x!*$ vs. $x \cup \emptyset$
- ▶ postfix vs. infix: $x!$ and stop parsing vs. $x+$ and keep parsing
- ▶ precedence conflict: $x + y \cdot z$

First idea (eventually abandoned)

- ▶ operator dictionary
 - ▶ globally assign fixity, precedence to all operators
partially done as part of MathML
- ▶ problematic for parsing
 - ▶ controversial to standardize
 - ▶ difficult to remember

MathML Operator Dictionary

MathML Core, Appendix B

- ▶ classifies > 1000 mathematical symbols
- ▶ 13 categories: pre/in/postfix, brackets, binders, ...
- ▶ 3 precedence levels for infix operators
- ▶ defines in particular
 - ▶ horizontal space
 - ▶ vertical stretch

Key application: default spacing in presentation

- ▶ symmetric space for infix
- ▶ wider space for lower precedence infix

MathML Operator Dictionary

MathML Core, Appendix B

- ▶ classifies > 1000 mathematical symbols
- ▶ 13 categories: pre/in/postfix, brackets, binders, ...
- ▶ 3 precedence levels for infix operators
- ▶ defines in particular
 - ▶ horizontal space
 - ▶ vertical stretch

Key application: default spacing in presentation

- ▶ symmetric space for infix
- ▶ wider space for lower precedence infix

crazy idea: use spacing for parsing too

Space-Based Parsing

Fixity: symmetry of space characters around operator

- ▶ more on left: prefix $x _ + _ ! y$
- ▶ more on right: postfix $x ! _ + _ y$
- ▶ same on both sides: infix $x _ ! _ y$

Precedence: number of space characters

- ▶ longer infixes bind less

$$a _ _ _ \wedge _ _ _ b _ _ = _ _ c _ + _ d * e$$

$$a _ \wedge _ b = (c _ + _ d * e)$$

Reviewer: “struck me at first as bizarre [...] thinking more, I was sold” — same for me

Space-Based Parsing

Fixity: symmetry of space characters around operator

- ▶ more on left: prefix $x _ + _ ! y$
- ▶ more on right: postfix $x ! _ + _ y$
- ▶ same on both sides: infix $x _ ! _ y$

Precedence: number of space characters

- ▶ longer infixes bind less $a _ _ _ \wedge _ _ _ b _ _ = _ _ c _ + _ d _ * _ e$
 $a _ \wedge _ b = (c _ + _ d _ * _ e)$

Reviewer: “struck me at first as bizarre [...] thinking more, I was sold” — same for me

Open issue: operator chains without any spaces

- ▶ doesn't work: $\neg T \wedge \Box \neg \forall x. B$
- ▶ works: $\neg T _ _ \wedge _ _ \Box _ \neg \forall x. B$
- ▶ would prefer: $\neg T _ \wedge _ \Box _ \neg \forall x. B$
- ▶ maybe good convention: prefix > postfix > nullfix

Conclusion

Experiments with Context-Free Parsing in UniFormal

Character level: leverage Unicode categories

- ▶ good for identifier-operator lexing
- ▶ great for bracket parsing
- ▶ bad for binders lookahead used
- ▶ 'script' property insufficient for math ad hoc definition used

AST level: count space around operators

- ▶ great for human-readable and parsable notations
- ▶ tricky when using no spaces

In practice

- ▶ my code: spaces often already used to enhance readability
- ▶ Isabelle: typically 1-whitespace around infixes, often brackets
sporadic use of `⋀` to help readability
apparently removed since I wrote my paper
- ▶ Lean: 1-whitespace around infixes, few brackets
formulas unpleasant for casual reading