What Does this Notation Mean Anyway?

BNF-Style Notation as it is Actually Used

D. A. Feller J. B. Wells S. Carlier F. Kamareddine

Heriot Watt University

July 5, 2018

What is MBNF?

Math-BNF (MBNF) is sometimes called “abstract syntax.” We avoid that name because MBNF is in fact a concrete form. It consists of production rules roughly of this form:

$$\bullet ::= \circ_1 \mid \cdots \mid \circ_n$$

What is MBNF?

Math-BNF (MBNF) is sometimes called "abstract syntax." We avoid that name because MBNF is in fact a concrete form. It consists of production rules roughly of this form:

$$\bullet ::= \circ_1 \mid \cdots \mid \circ_n$$

Unlike BNF, MBNF production rules contain chunks of mathematical text and themselves stand for abstract mathematical structures.

BNF and MBNF

- A BNF rule "$P ::= P \star P$" replaces an occurrence in a string of P by $P \star P$. The star can only be a symbol. The language of the non-terminal P is the set of non-terminal-free strings reachable from the string P by the grammar's rules.
- An MBNF rule "$P \in S ::= P \star P$ if C" requires for $P_1, P_2 \in S$ that if the condition C (which can use the full power of mathematics (WCUTFPM)) holds, then the object $P_1 \star P_2$ belongs to S. The star can be any mathematical operator (WCUTFPM), or it can form part of an arrangement. Such arrangements are identified up to user-declared equivalences (WCUTFPM). Usually, the sets declared by a MBNF grammar are the unique smallest sets satisfying the rules, if such a choice of sets exists.

What is Math-Text?

The omission of some parentheses is inherited from Math-Text.

$$(\lambda x.((x\,y)\,z)) = \lambda x.x\,y\,z$$

What is Math-Text?

The omission of some parentheses is inherited from Math-Text.

$$(\lambda x.((x\,y)\,z)) = \lambda x.x\,y\,z$$

$$x_1, x_2, \ldots, x^1, x^2, \ldots, x', x'', \ldots$$

What is Math-Text?

The omission of some parentheses is inherited from Math-Text.

$$(\lambda x.((x\,y)\,z)) = \lambda x.x\,y\,z$$

$$x_1, x_2, \ldots, x^1, x^2, \ldots, x', x'', \ldots$$

$$^{c}{\downarrow}\,\check{p}\langle v''_x \odot a^{2+1}\rangle - \overline{f^n_x + \overline{y \cdot fj}} + \sum_{i=0}^{\infty} s_i \xrightarrow{a,b} \hat{a}$$

What is Math-Text?

The omission of some parentheses is inherited from Math-Text.

$$(\lambda x.((x\,y)\,z)) = \lambda x.x\,y\,z$$

$$x_1, x_2, \ldots, x^1, x^2, \ldots, x', x'', \ldots$$

$${}^{c}{\downarrow}\, \check{p}\langle v''_x \odot a^{2+1}\rangle - \overline{f^n_x + \overline{y \cdot fj}} + \sum_{i=0}^{\infty} s_i \xrightarrow{a,b} \hat{a}$$

MBNF requires us to interpret some pieces of math text which stand essentially for themselves:

$1+3$	Stands for	4
$\lambda x.x$	Stands for	$\lambda x.x$

MBNF Allows Arbitrary Operators Inside Production Rules

Chang and Felleisen [CF12, p 134] give the following MBNF grammar :

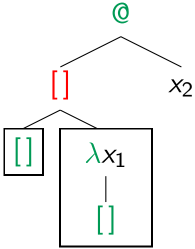
$$
\begin{aligned}
e &::= x \mid \lambda x.e \mid e\ e \\
A &::= [\,] \mid A[\lambda x.A]\ e
\end{aligned}
$$

MBNF Allows Arbitrary Operators Inside Production Rules

Chang and Felleisen [CF12, p 134] give the following MBNF grammar :

$$e \ ::= \ x \mid \lambda x.e \mid e \ e$$
$$A \ ::= \ [\,] \mid A[\lambda x.A] \ e$$

$$A = [\,]\,[\,\lambda x_1.[\,]\,]\,x_2$$

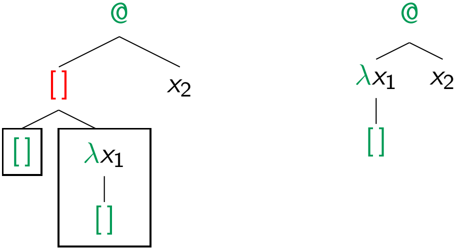


MBNF Allows Arbitrary Operators Inside Production Rules

Chang and Felleisen [CF12, p 134] give the following MBNF grammar :

$$
\begin{aligned}
e &::= x \mid \lambda x.e \mid e\ e \\
A &::= [\,] \mid A[\lambda x.A]\ e
\end{aligned}
$$

$$A = [\,]\,[\,\lambda x_1.[\,]\,]\,x_2$$



MBNF Mixes Math Stuff With BNF-Style Notation

Sometimes production rules are written in the form $v \in S ::= \cdots$

Germane and Might [GM17, pg 20] give the following MBNF grammar:

$$\begin{aligned}
u &\in \mathit{UVar} \\
k &\in \mathit{CVar} \\
\mathit{lam} \in \mathit{Lam} &= \mathit{ULam} + \mathit{CLam} \\
\mathit{ulam} \in \mathit{ULam} &::= (\lambda e(u^* k)\mathit{call}) \\
\mathit{clam} \in \mathit{CLam} &::= (\lambda_\gamma(u^*)\mathit{call}) \\
\mathit{call} \in \mathit{Call} &= \mathit{UCall} + \mathit{CCall} \\
\mathit{ucall} \in \mathit{UCall} &::= (f e^* q)_\ell
\end{aligned}$$

$$\begin{aligned}
\mathit{ucall} \in \mathit{UCall} &::= (f e^* q)_\ell \\
\mathit{ccall} \in \mathit{CCall} &::= (q\, e^*)_\gamma \\
e, f \in \mathit{UExp} &= \mathit{UVar} + \mathit{ULam} \\
q \in \mathit{CExp} &= \mathit{CVar} + \mathit{CLam} \\
\ell &\in \mathit{ULab} \\
\gamma &\in \mathit{CLab}
\end{aligned}$$

MBNF Has at Least the Power of Indexed Grammars

Inoe and Taha [IT12, pg 361] use this MBNF rule:

$$\mathcal{E}^{\ell,m} \in ECtx_{\mathbf{n}}^{\ell,m} ::= \cdots \mid \langle \mathcal{E}^{\ell+1,m} \rangle \mid \tilde{}\mathcal{E}^{\ell-1,m}[\ell > 0] \mid \cdots$$

MBNF Allows Arbitrary Side Conditions on Production Rules

Chang and Felleisen [CF12, p 134] give the following MBNF rule:

$$E = [\,] \mid E\ e \mid A[E] \mid \hat{A}[A[\lambda x.\check{A}[E[x]]]E] \qquad \text{where } \hat{A}[\check{A}] \in A$$

MBNF "Syntax" Can Contain Very Large Infinite Sets

Toronto and McCarthy [TM12, p 297] write:

$$e ::= \cdots \mid \langle t_{set}, \{e^{*\kappa}\} \rangle$$

Later they tell us $\{e^{*\kappa}\}$ denotes "sets comprised of no more than κ terms from the language of e". It seems as though κ is intended to be an inaccessible cardinal, i.e., a truly big infinity.

MBNF Allows Infinitary Operators

Fdo, Díaz and Núñez [LDN97, p 539] write an MBNF grammar with the following operator, which the authors state is infinitary:

$$P ::= \cdots \mid \prod_{i \in I} P_i \mid \cdots$$

MBNF Allows Co-Inductive Definitions

Eberhart, Hirschowitz and Seiller [EHS15, p 94] intend the following MBNF grammar to define infinite terms co-inductively:

$$
\begin{aligned}
P, Q &::= \Sigma_{i \in n} G_i \mid (P|Q) \\
G &::= \overline{a}\langle b \rangle.P \mid a(b).P \mid \nu a.P \mid \tau.P \mid \heartsuit.P
\end{aligned}
$$

Our Proposal:

Syntactic Math Text (SMT)
Plus a Definition of Production Rules

SMT: Arrangements and Objects

Example Arrangements:

O_{a*b}		$1 \odot x$	$\overline{O \cdot P}$	$\clubsuit$	O
$p \xrightarrow{q,r} M$	$p^{x,y}$	${}^{x}\uparrow \cdot y$	$\underline{\heartsuit.P}$	$\frac{a-b}{c}$	O^{a}_{b}
$\lambda x.xy$	$P \mid Q$	$\{a,b\}$	0	$\Box \cdot \Box$	$\tilde{\mathcal{E}}$

SMT: Arrangements and Objects

Example Arrangements:

O_{a*b}		$1 \odot x$	$\overline{O \cdot P}$	$\clubsuit$	O
$p \xrightarrow{q,r} M$	$p^{x,y}$	${}^{x}\uparrow \cdot y$	$\underline{\heartsuit.P}$	$\frac{a-b}{c}$	O^a_b
$\lambda x.xy$	$P \mid Q$	$\{a,b\}$	0	$\Box \cdot \Box$	$\tilde{}\mathcal{E}$

Example Objects:

$\{\lambda x.x\,y, \lambda z.z\,y, \ldots\}$ $\quad$ $\{\{a,b\}, \{b,a\}, \{a,a,b\}, \{a,b,b\}, \ldots\}$

$\{P \mid Q, P \mid Q \mid 0, Q \mid P, \ldots\}$ $\quad$ $\{\clubsuit\}$

$\Box$

Pointers to objects appear in arrangements. Objects and arrangements may be nested within one another.

Relatively Mundane Features of our Model

We define the following in what might be considered a fairly standard way:

- Context-hole filling.
- Compatible closure (congruence).
- The concept of free names.
- α-Equivalence.
- Capture avoiding substitution.

We define name groups as an equivalence relation on the set of objects, which we write $\sim$. This relation can be extended as an author requires.

SMT: Primitive Constructor Decomposition

Primitive constructors:

$$\langle\Box\rangle \qquad !\Box \qquad \Box \to \Box \qquad \underline{\Box} \qquad \lambda\Box.\Box$$

SMT: Primitive Constructor Decomposition

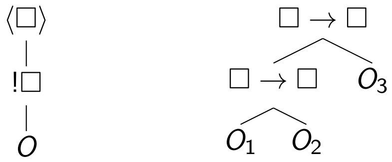
Primitive constructors:

$$\langle \Box \rangle \qquad !\Box \qquad \Box \to \Box \qquad \underline{\Box} \qquad \lambda \Box . \Box$$

Primitive constructor decompositions:

$$\langle (!O) \rangle = \langle \Box \rangle [!\Box[O]]$$

$$(O_1 \to O_2) \to O_3 = (\Box \to \Box)[(\Box \to \Box)[O_1, O_2], O_3]$$



Production Rules for Defining Syntactic Sets

$$v_1, ..., v_n \in S ::= e_1 \text{ if } c_1 \mid \cdots \mid e_m \text{ if } c_m$$

$v_1, ... v_n$ are metavariables ranging over S.
S is the name of the subset of object being defined.
Each of the expressions, $e_1, ..., e_m$, is either an object level variable, a primitive constructor which is allowed to have metavariables in the place of holes.
Each optional side condition, $c_1, ..., c_m$, is a formula with expressions in the place of holes.

An Example Almost Exactly as Normal

Our model does not require that authors adjust their practices too much. For example, here is the λ-calculus:

$$e \in \mathsf{Exp} ::= v \mid \lambda v.e \mid e\, e$$

$\lambda\Box.\Box$ binds any name placed in its first hole in both its holes.

We are working modulo α-equivalence.

Our rewriting rules are the Exp-compatible closure of the following relations:

$$(\lambda v.e_1)e_2 \xrightarrow{\beta} e_1[v := e_2]$$

$$\lambda v.e_1 v \xrightarrow{\eta} e_1$$

Challenges we Encountered

- Authors generally define equivalences in whatever way they please.
- Authors want to examine sub-trees and perform calculations on them while retaining the full power of whatever equivalences they defined in their grammar.
- Authors extend and alter their grammars on the fly.
- Authors are rarely specific about the requirements of a grammar and often don't acknowledge when it is doing something interesting.
- Our definition had to mesh with existing mathematical language.
- Our definition could not just give a mathematical structure, it had to give a clear way of matching it to a concrete syntactic structure.
- The machinery we employ must remain largely invisible.
- We had to give a structure appropriate for working with inductively.
- The representation we provide must remain close to what the authors had in mind.
- Even partial descriptions of how this notation works are spread very widely and sparsely throughout the literature.

Conclusions

- MBNF is distinct from BNF in non-trivial ways.
- We should be documenting the more interesting examples of this notation.
- MBNF continues to be used in novel ways.
- We need a semi-formal definition of how both MBNF and the surrounding syntactic metalanguage define mathematical entities that is aimed at human readers.
- A fairly large cross section of MBNF and much of the surrounding metalanguage has a model in ZFC.
- We need to determine what the limitations of this notation are and clearly define the conditions under which it can be used

References I

- Stephen Chang and Matthias Felleisen, *The call-by-need lambda calculus, revisited*, in Seidl [Sei12], pp. 128–147.
- Clovis Eberhart, Tom Hirschowitz, and Thomas Seiller, *An Intensionally Fully-abstract Sheaf Model for π^**, 6th Conference on Algebra and Coalgebra in Computer Science (CALCO 2015) (Dagstuhl, Germany) (Lawrence S. Moss and Pawel Sobocinski, eds.), Leibniz International Proceedings in Informatics (LIPIcs), vol. 35, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 86–100.
- Kimball Germane and Matthew Might, *A posteriori environment analysis with pushdown delta cfa*, Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (New York, NY, USA) (Matthew Fluet, ed.), ACM, 2017.
- Jun Inoue and Walid Taha, *Reasoning about multi-stage programs*, in Seidl [Sei12].

References II

Luis Fdo. Llana Díaz and Manuel Núñez, *Testing semantics for unbounded nondeterminism*, Euro-Par'97 Parallel Processing (Berlin, Heidelberg) (Christian Lengauer, Martin Griebl, and Sergei Gorlatch, eds.), Springer Berlin Heidelberg, 1997, pp. 538–545.

Helmut Seidl (ed.), *Programming languages and systems*, Springer, 2012.

Neil Toronto and Jay McCarthy, *Computing in cantor's paradise with λ zfc*, Functional and Logic Programming (Berlin, Heidelberg) (Tom Schrijvers and Peter Thiemann, eds.), Springer Berlin Heidelberg, 2012, pp. 290–306.

Questions