

From notation to semantics: there and back again

Luca Padovani

`padovani@sti.uniurb.it`

University of Urbino

Stefano Zacchiroli

`zacchiro@cs.unibo.it`

University of Bologna

August 12, 2006

MKM 2006

Wokingham, UK

Levels of formula encoding

Most MKM software applications have to deal with mathematical formulae and work on some internal encoding of them.

Notational level (aka presentation level) formulae are encoded on the basis of their (visual) rendering. **Interaction with the user** usually happens on formulae at this encoding level.

Semantic level domain and application specific level, formulae are encoded so that the application has the deepest understanding on them. **Computations** usually happen here.

Content level intermediate level, encodes the structure of formulae (and, to a limited extent, the semantics). Is meant as a vehicle of **interoperability** among applications with different foundations.

The role of mathematical notation

Mathematical notation is an open, structured, and ambiguous language, agreed upon by mathematicians. Offering it has a user language is a desirable feature of all MKM software user interfaces.

E.g.:

- proof assistants (like other applications producing mathematical content) should let authors define their own notation for freshly formalized concepts;
- search engines (like other apps consuming mathematical content) should let users search using the notation they are familiar with.

The need of an unifying **notational framework** for designing and discussing notation support in MKM apps is the thrust of this work.

Outline

1. introduction

- encoding of formulae in MKM applications
- notational support in MKM applications

2. a generic framework for meaningful notation

- architecture, some details of the formalization
- Matita-specific features and implementation

3. exploiting meaningful notation in Matita

- hypertextual browsing, semantic selection & direct manipulation

4. wrap up

Outline

1. introduction

- encoding of formulae in MKM applications
- notational support in MKM applications

2. a generic framework for meaningful notation

- architecture, some details of the formalization
- Matita-specific features and implementation

3. exploiting meaningful notation in Matita

- hypertextual browsing, semantic selection & direct manipulation

4. wrap up

“Meaningful” mathematical notation

A natural architecture for a notational framework is layered according to the encoding levels and keep synchronized formulae across them.

Features we require from such a framework:

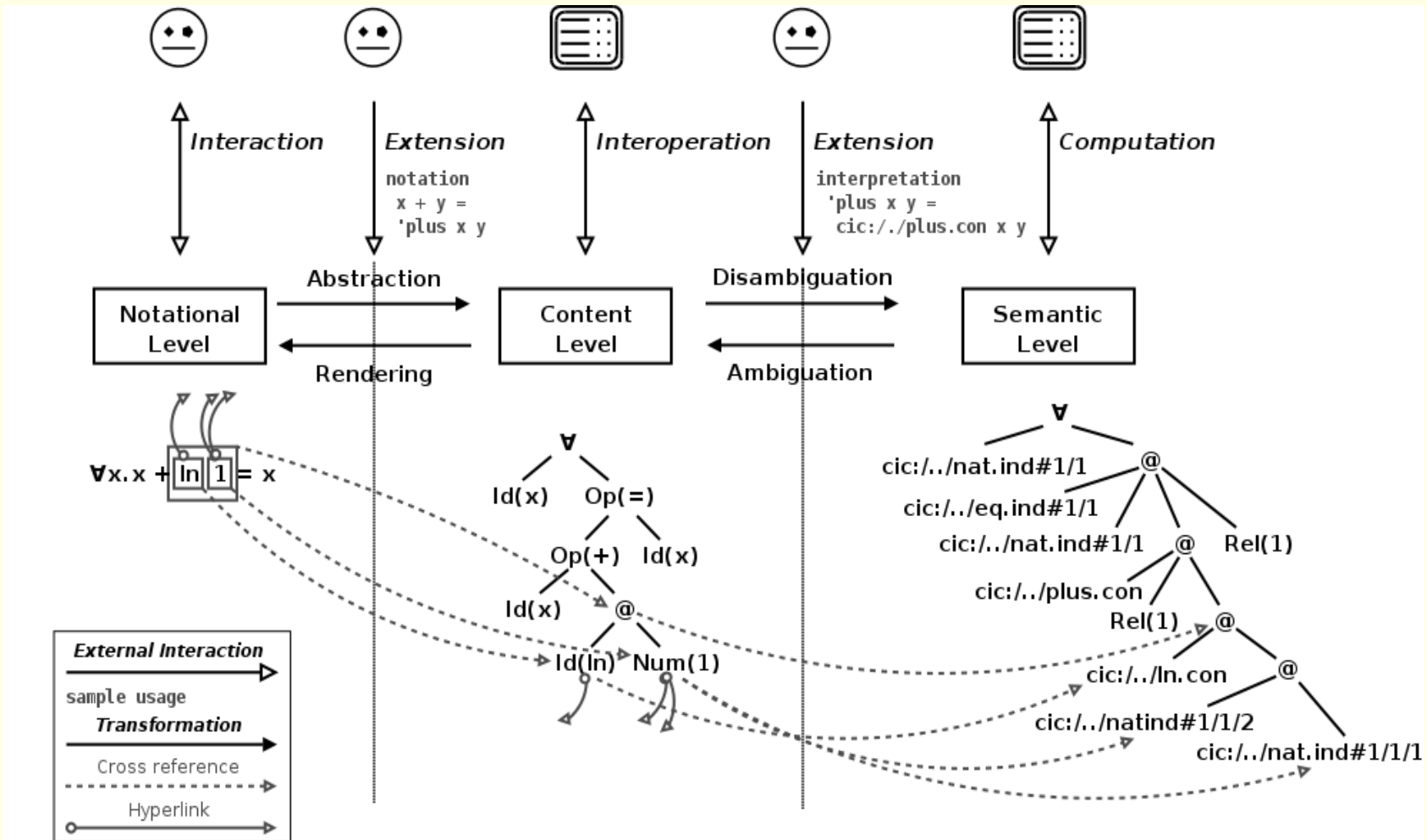
extensibility it should be possible to incrementally defines transformation rules, building on top of previously defined rules;

remote control interactions at the notational level should be able to be reflected at content and semantic levels;

ambiguity being common in mathematical artifacts, ambiguity should be accounted for;

interoperability communication with other applications should not be hindered.

Architecture



Some details: notational level

$E^p ::=$		(presentation expressions)
	x	(identifier)
	$l@H$	(literal)
	$A\{E^p\}$	(annotation)
	$L[E_1^p, \dots, E_n^p]$	(layout)
	$B[E_1^p \dots E_n^p]$	(box)
	α	(variable)

Literals l (atomic notational elements)

Layout schemata L (constructors of mathematical notation)

Box schemata B (line breaking hints)

Some details: content level

E^c	$::=$		(content expressions)
		x	(identifier)
		$s@H$	(symbol)
		$A\{E^c\}$	(references)
		$C[E_1^c, \dots, E_n^c]$	(constructor)
		α	(variable)

Constructors C (compound objects builders)

Some details: notational equations

A well-formed **presentation pattern** is a content expression without identifiers, hyperlinks and cross references, s.t. any variable occurs exactly once. Well formed content pattern are similarly defined.

A **notational equation** is a pair of well-formed patterns:

$$P^p \iff P^c$$

that simultaneously augment abstraction and rendering rules (more details on this in the paper).

E.g.

$$\begin{aligned} hvbox[\alpha \text{ break} = \beta] &\iff \text{apply}[\text{eq}, \alpha, \beta] \\ hvbox[\alpha \text{ break} \neq \beta] &\iff \text{apply}[\text{not}, \alpha = \beta] \end{aligned}$$

Some details: ambiguity in Matita

We instantiated the framework to the Matita proof assistant (and to a minor extent to the Whelp search engine).

SPAM: try Matita today at <http://matita.cs.unibo.it>

The semantic level is (an XML encoding of) terms of the Calculus of (Co)Inductive Constructions (CIC for short).

A (symbol) **interpretation** is a pair:

$$s \alpha_1 \cdots \alpha_n \iff t[\alpha_1, \dots, \alpha_n]$$

where s is a content symbol of arity $n \geq 0$ and $t[\alpha_1, \dots, \alpha_n]$ is a CIC term with n holes labelled $\alpha_1, \dots, \alpha_n$. Other support source of ambiguities are literal numbers and unbound identifiers.

Implementation

The framework instantiation to Matita consists of:

1. an implementation of a well-known pattern matching algorithm (+ backtracking to deal with meta-operators);
2. additional commands in the proof language which let the user provide notational equations and interpretations;
3. mapping from notational equations to API invocations of an extensible parser (CamLP4 in our case);
4. mapping from notational equations and interpretation to rows in the pattern matching matrix.

Full scale example: \exists notation in Matita

Notational equation and interpretation from the script
library/logic/connectives.ma:

```
notation "hvbox(\exists ident i opt (: ty) break . p)"
  right associative with precedence 20
for @{ 'exists ${ default
  @{ \lambda ${ident i} : $ty. $p }
  @{ \lambda ${ident i} . $p }
}}.
```

```
interpretation "exists" 'exists \eta.x =
  (cic:/matita/logic/connectives/ex.ind#xpointer(1/1) _ x).
```

Highlights: meta-operators, implicit variables, η -expansion.

Outline

1. introduction

- encoding of formulae in MKM applications
- notational support in MKM applications

2. a generic framework for meaningful notation

- architecture, some details of the formalization
- Matita-specific features and implementation

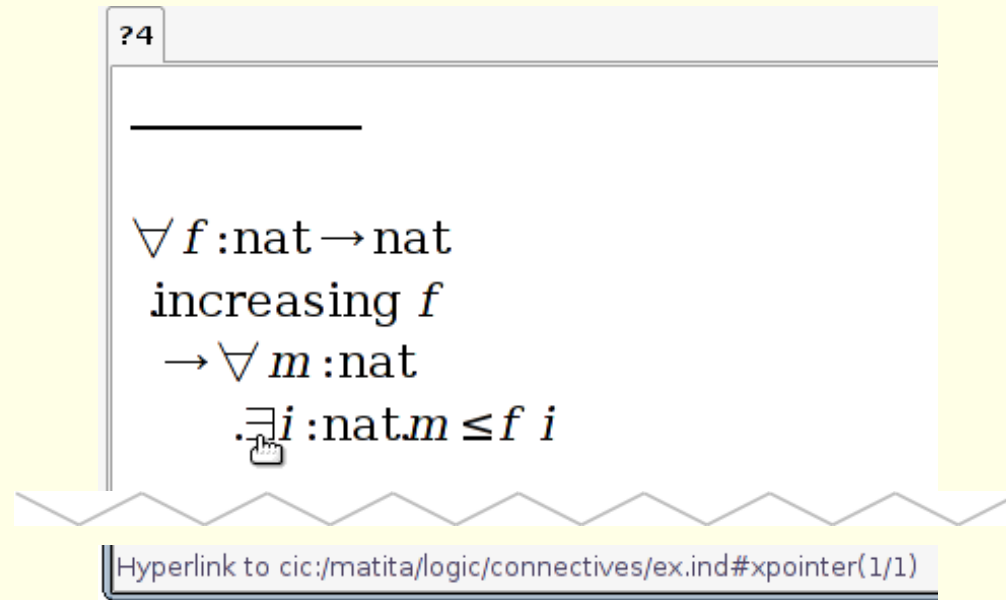
3. exploiting meaningful notation in Matita

- hypertextual browsing, semantic selection & direct manipulation

4. wrap up

Exploiting remote control in Matita

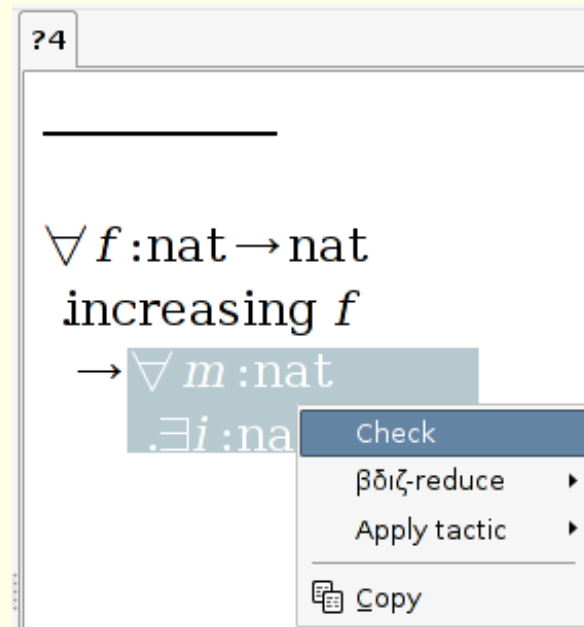
[1/2]



Hyperlinks are exploited to implement **hypertextual browsing** of the proof assistant knowledge base.

Exploiting remote control in Matita

[2/2]



Cross references are exploited to implement **semantic selection**: visual selection at the notational level is constrained to well-formed CIC terms at the semantic level. **Direct manipulation** of CIC terms is also possible directly on the notational level.

Outline

1. introduction

- encoding of formulae in MKM applications
- notational support in MKM applications

2. a generic framework for meaningful notation

- architecture, some details of the formalization
- Matita-specific features and implementation

3. exploiting meaningful notation in Matita

- hypertextual browsing, semantic selection & direct manipulation

4. wrap up

Future work

Issues with CamlP4 non-standard grammar productions, no support for multiple parse trees (ambiguous parsing) → we are investigating the implementation of an ambiguous parsing algorithm (like J. Rekers').

Support for numbers letting the user describe common number encodings as a set of simple regular rules.

Local notation currently the rules governing level boundaries are global, in practice it is often useful to limit notation effectiveness to binder scopes (as in **let \odot be a binary operation over ...**).

Conclusion

We identified meaningful mathematical notation as a common need of several MKM applications. We identified a set of requirements characterizing it.

We designed an architecture for and formalized (part of) a framework fulfilling those requirements.

We instantiated the framework to a specific semantic setting (CIC) in the Matita proof assistant.

We hope other assessments of the framework will be provided by its instantiations in other systems.

Questions?