Formal Mathematics in Informal Language

Aarne Ranta

MathWiki Workshop, Edinburgh, 31 October - 1 November 2007

Mature technology?

Does type theory provide a mature technology for mathematics?

Mature technology...

... can be hidden from its users.

Cf.

- cars 100 ago vs. now
- Unix 30 years ago vs. Mac OS X and Ubuntu Linux

What is needed to hide technology

It must be **reliable** enough not to remind of itself

It must relate **ordinary practices** and solve ordinary problems

We need to make a **conscious effort** for hiding it

Ordinary practices in mathematics

Expert formal proof community:

- type theoretical formalism
- interactive proving, helped by some automation
- automatic proof checking

Undergraduate mathematics student:

- natural language with some mathematical symbolism
- automation in numeric and symbolic computation (computer algebra)
- proof checking a distant dream

Language of ordinary mathematics

Symbols for numbers, variables, arithmetic, calculus

Sentence structure (logic and predicates) mostly in natural language

Proof structure mostly in natural language

Controlled text structure: definitions, theorems

Diagrams for geometry, sets, category theory

State of the art in hiding technology

Symbols

- numbers, variables, arithmetic: FORTRAN
- calculus: computer algebras

Sentence structure: mostly requires system-dependent formalism

Proof structure: natural language structural words, or system-dependent unreadable formalism, or proof not shown

Controlled text structure: Mizar

Diagrams?

A realistic goal

Natural language representation of formalized mathematics with the look and feel of informal mathematics

- definitions and theorems
- controlled text structure

Proofs in natural language?

- still a research problem
- largely the same as problems in intelligible formal proofs
- but: if this is solved, we already know how to select structural words

Some approaches to natural language mathematics

de Bruijn: Mathematical Vernacular (1980's)

Coscoy, Kahn, and Théry: explaining proofs in Coq (1995)

Fiedler, Horacek and Siekmann: explaining proofs in Omega (2000)

Ranta: GF, Grammatical Framework (1998-)

GF from the LF point of view

GF = Logical Framework + concrete syntax

Every **category** (basic type) is equipped with a **linearization type**:

```
cat Set ;
lincat Set = {s : Number => Str}
```

Every **function** (object former) is equipped with a **linearization**:

```
fun Integer = Set ;
lin Integer = {s = table {Sg => "integer" ; Pl => "integers"}} ;
```

Forms of judgement in GF

Abstract syntax

form	**reading**
`cat` *C G*	*C* is a category in context *G*
`fun` *f* `:` *A*	*f* is a function of type *A*

Concrete syntax

form	**reading**
`lincat` *C* `=` *T*	category *C* has linearization type *T*
`lin` *f* `=` *t*	function *f* has linearization *t*
`param` *P* `=` *C1* | ... | *Cn*	parameter type P has constructors *C1...Cn*
`oper` *h* `:` *T* `=` *t*	operation *h* of type *T* is defined as *t*

Multilinguality

Multilingual grammar: one abstract syntax + several concrete syntaxes

```
lincat Set = {s : Number => Str}                        -- Eng
lin Integer = {s = table {Sg => "integer" ; Pl => "integers"}} ;

lincat Set = {s : Number => Str ; g : Gender}           -- Fre
lin Integer = {s = table {Sg => "entier" ; Pl => "entiers"} ; g = Masc} ;
param Gender = Masc | Fem ;

lincat Set = {s : Species => Number => Str ; Gender} -- Swe
lin Integer = {
  s = table {
    Indef => table {_ => "heltal"} ;
    Def   => table {Sg => "heltalet" ; Pl => "heltalen"}} ;
  g = Neutr
  } ;
param Gender = Utr | Neutr ;
param Species = Indef | Def ;
```

Resource grammar libraries

Hiding linguistic knowledge from the authors of **application grammars**.

```
lincat Set = N ; -- functor over resource grammar interface

lin Integer = mkN "integer" ; -- Eng

lin Integer = mkN "entier" ;  -- Fre

lin Integer = mkN "heltal" "heltal" ; -- Swe
```

The current library covers 10-12 languages.

The resource grammar library API

Overloaded syntax and inflection functions

```
pred : V  -> NP -> Cl         -- x converges
pred : V2 -> NP -> NP -> Cl   -- x intersects y
pred : A  -> NP -> Cl         -- x is even
pred : A2 -> NP -> NP -> Cl   -- x is divisible by y

mkN  : Str -> N               -- integer
mkN  : Str -> Str -> V        -- radius, radii
```

GF as proof editor

Abstract syntax trees: dependently typed second-order beta-eta normal terms (cf. Plotkin's framework 2007)

User-extensible definitional equality (similar to ALF)

This machinery is mainly used for enforcing well-typedness:

```
fun request : (k : DeviceKind) -> Action k -> Device k -> Request
```

It can also be used for "proof-carrying documents":

```
fun Connect : (x,y,z : City) ->
      (u : Flight x y) -> (v : Flight y z) ->
        IsPossible x y z u v -> Flight x z ;
```

Two ways of using GF for mathematics: 1

As a proof editor using its own type system and syntax editor

- not very developed for building proofs
- buggy constraint solving
- has only been used for toy examples

Two ways of using GF for mathematics: 2

As an **annotation language** for another proof system

- context-free typing in GF, semantic control from host system
- has been used for many systems:
  - Alfa (Hallgren & Ranta, LPAR 2000)
  - OCL (Hähnle, Johannisson & Ranta, ETAPS 2002)
  - MathML + Maple (WebALT project, 2005-2006)

– PhoX (Humayoun & Raffalli, 2007-)

– Wikicoqweb (Pottier 2007)

A WebALT example

(From Ng'ang'a, Laine and Carlson, "Multilingual Generation of Live Math Problems in WebALT", 2005)

Formal representation:

```
Attrib([nlg:mood nlg:imperative nlg:tense nlg:present,
  nlg:directive nlg:determine],plangeo1:are_on_line(A,B,C))
```

Linearizations:

Determine if A, B and C are collinear.

Määritä ovatko A, B ja C suoralla.

Determina si A, B y C son colineales.

Déterminer si A, B et C sont sur une droite.

Determina se A, B e C sono su una linea.

Bestäm om A, B och C är på en linje.

The translation problem

The translation must be semantically accurate - render exactly the same mathematical problem.

But the syntactic structures may differ:

- some languages use the imperative (*determina*), some the infinitive (*déterminer*)

- some language use an adjective (*colineales*), some an adverbial phrase (*sur une droite*)

What has become of WebALT

WebALT Inc. was founded before the project ended in 2006

JEM, "Join Educational Mathematics", EU network started in September 2006

WebALT-EU27 summer school planned

WebALT Africa?

Wanjiku Ng'ang'a, "Multilingual content development for eLearning in Africa". eLearning Africa: 1st Pan-African Conference on ICT for

Development, Education and Training. 24-26 May 2006, Addis Ababa, Ethiopia.

Demos: `webalt.math.helsinki.fi/PublicFiles/CD/Screencast/`

GF as annotation language

The GF grammar covers the host language, and defines

- the host language notation itself
- translations to and from other languages
- a syntax editor for all these languages

The grammar consists of

- **ground grammar**: the framework-level concepts of the host language

- **extensions**: user-defined annotations for user-defined concepts

Example: GF Alfa, `www.cs.chalmers.se/~hallgren/Alfa/Tutorial/GFplugin.html`

GF and web technology

A multilingual GF grammar can be compiled to a JavaScript program (Bringert 2006), which implements

- abstract syntax editing
- linearization
- random generation
- (parsing is forthcoming)

This has been extended to syntax editors embedded in Wiki pages (Bringert and Meza Moreno 2007):

- content displayed in any language
- content created by syntax editor in any language
- abstract syntax stored in server
- concrete syntax created in web browser

Demo: `csmisc14.cs.chalmers.se/~meza/restWiki/wiki.cgi`

A possible project task

Wiki for creating definitions and theorems

Creation and display in multiple languages

- natural languages
- different proof editor formalisms

Proofs would probably not be displayed (too difficult), but proof status and links to formal proofs would be given

This would be useful for

- disseminating results outside the expert community
- communicating between experts working in different formalisms