Mathematical Proofs in Natural Languages (MathNat)

Muhammad Humayoun

Departement of Mathematics (LAMA)
University of Savoie
mhuma@univ-savoie.fr
http://www.lama.univ-savoie.fr/ humayoun/phd/mathnat.html

May 12, 2009

Automatic formalisation of mathematical text in three steps:

1. Syntax:

- Development of a controlled language for mathematical proofs
- A type theoretic approach (Using Grammatical Framework)
- Context-free typing in GF

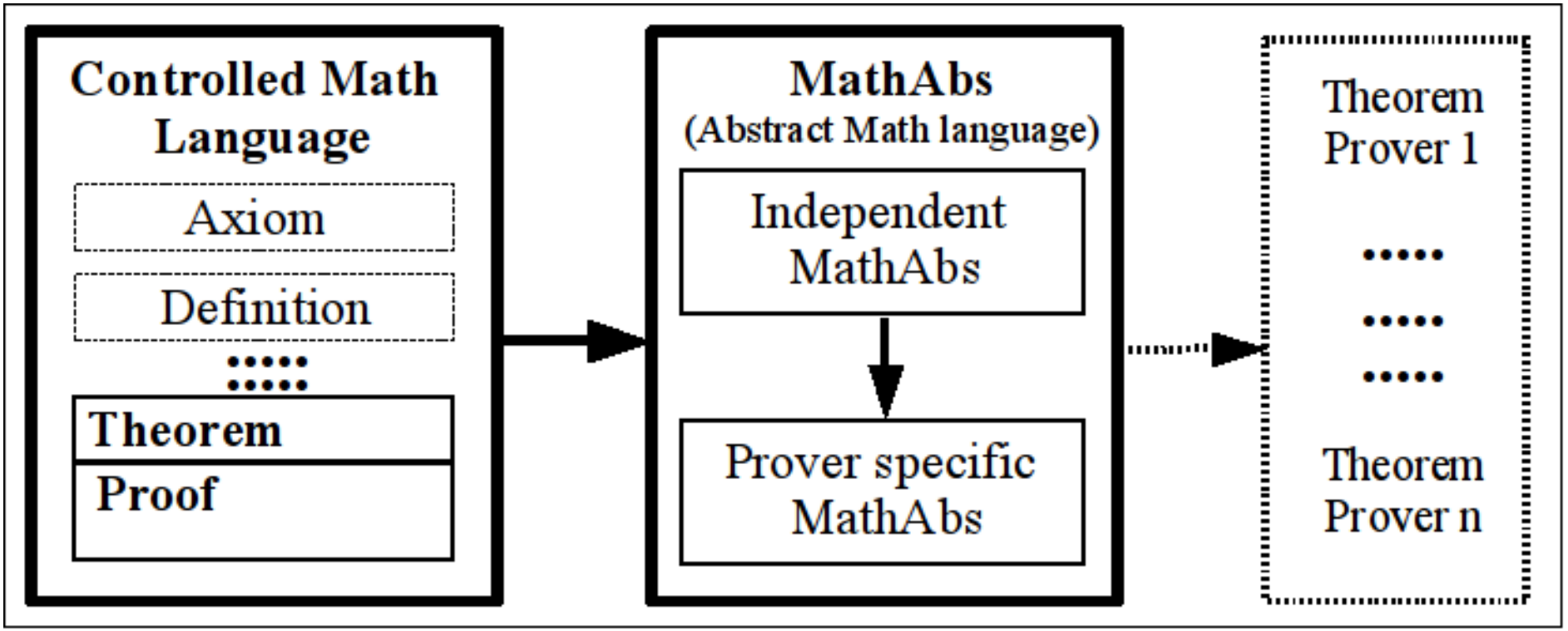
2. Semantics (The Host system of MathNat):

- Building context and basic anaphoric resolution
- Translation of controlled math proofs to an abstract mathematical language (MathAbs)

3. Proof checking:

- Translation of MathAbs into a prover specific formalism
- Not yet

Overall picture MathNat



The abstract mathematical language (MathAbs):

- A system independent formal language for mathematical proofs

Why MathAbs?

- Reasoning gaps in natural language mathematical proofs
  - obvious parts are exempted
  - many reasoning steps often performed in fewer steps
- Logical systems are not so natural for natural language proofs
  - e.g. sequent calculus, natural deduction etc
- **MathAbs allows to do many steps of reasoning in one step**
- Why not discource representation theory
  - quite general framework. perhaps an overkill for this task
  - requires extension to accomodate mathematical text[1]
  - not enough implementations around

[1] C. Zinn. 2004. Understanding Informal Mathematical Discourse. PhD thesis

The MathAbs language:

- Describes a tree of logical (meta) rules
- A quite small grammar
  - `let, assume, deduce, show, trivial`

- **Variable declaration:**
  - `let` Formula
    `let` *x*, `let` *x:Type*
- **Assuming a hypothesis:**
  - `assume` Formula (Hint)
    `assume` $x > 0$, `assume` $x > 0$ `by Form` *positive(x)*
- **Deduction:**
  - `deduce` Formula (Hint)
    `deduce` $x > 0$, `deduce` $x > 0$ `by Form` *positive(x)*
- **Proving a goal:**
  - `show` Formula (Hint)
    `show` *positive(x)*, `show` *positive(x)* `by Form` *positive*$(x) => (x > 0)$

MathNat conventions:

- Anaphoric resolution for `it` and `they`
  - We always pick the latest singular or plural object in the context

MathNat conventions:

- Anaphoric resolution for `it` and `they`
  - We always pick the latest singular or plural object in the context
- Two types of expressions:
  - An expression for equational reasoning (equality/inequality):
    - ★ we suppose that `x + y + 2 = z`. we conclude that they are equal.
      :=> `they` refers to `x`, `y` and `2`
    - ★ we assume that `x = 30`. we conclude that it is positive.
      :=> `it` refers to `x`
    - ★ let `x + y`. *not allowed*

MathNat conventions:

- Anaphoric resolution for `it` and `they`
  - We always pick the latest singular or plural object in the context
- Two types of expressions:
  - An expression for equational reasoning (equality/inequality):
    - ★ we suppose that `x + y + 2 = z`. we conclude that they are equal.
      :=> `they` refers to `x`, `y` and `2`
    - ★ we assume that `x = 30`. we conclude that it is positive.
      :=> `it` refers to `x`
    - ★ let `x + y`. *not allowed*
  - An expression appearing as a variable in declarative sentences:
    - ★ let `x` be a even integer. suppose it is positive.
      :=> `it` refers to `x`
    - ★ let `x + y` be a even integer. suppose it is positive.
      :=> `it` refers to `x+y`
    - ★ let $\sqrt{x}$ and `y` be positive. they are equal.
      :=> `they` refers to $\sqrt{x}$ and `y`
    - ★ let `x = y` be even. *not allowed*

MathNat conventions:

- Anaphoric resolution for `it` and `they`
  - We always pick the latest singular or plural object in the context
- Two types of expressions:
  - An expression for equational reasoning (equality/inequality):
    - ★ we suppose that `x + y + 2 = z`. we conclude that they are equal.
      :=> `they` refers to `x`, `y` and `2`
    - ★ we assume that `x = 30`. we conclude that it is positive.
      :=> `it` refers to `x`
    - ★ let `x + y`. *not allowed*
  - An expression appearing as a variable in declarative sentences:
    - ★ let `x` be a even integer. suppose it is positive.
      :=> `it` refers to `x`
    - ★ let `x + y` be a even integer. suppose it is positive.
      :=> `it` refers to `x+y`
    - ★ let $\sqrt{x}$ and `y` be positive. they are equal.
      :=> `they` refers to $\sqrt{x}$ and `y`
    - ★ let `x = y` be even. *not allowed*
- Repetition of type is not allowed.
  - with one exception
    "recall that `x` is an integer" *not yet*

Future work

Long way to go....

Coverage:

- Importing definitions from WebAlt project
- Solving more anaphora
- Coverage for considerable number of proofs from elementary number theory
- Perhaps those proofs which already formalised in proof assistants (collaboration?)

Proof checking:

- Proof checking in PML

Demo

Welcome to the demo