Toward Automatic Formalization of Informal Mathematics with MathNat

MKM 2010

Muhammad Humayoun, Christophe Raffalli
(firstname.lastname@univ-savoie.fr)

Laboratoire de mathématiques, université de Savoie

July 2010
Outline
Some facts

- Mathematical English is universally accepted by all mathematicians.
- Mathematical English is mostly a sub-language of English.
- Trivial translation from formal proofs to English is easy.
Some facts

- Mathematical English is universally accepted by all mathematicians.
- Mathematical English is mostly a sub-language of English.
- Trivial translation from formal proofs to English is easy but not easily accepted by human reader.
- Parsing natural languages is easier than good text generation.
Some facts

- Mathematical English is universally accepted by all mathematicians.
- Mathematical English is mostly a sub-language of English.
- Trivial translation from formal proofs to English is easy but not easily accepted by human reader.
- Parsing natural languages is easier than good text generation but it is still very difficult.
- Even when parsing succeeds, proof-checking is still very hard.
Our goal

- Define a small subset of mathematical English with some rich linguistic features.
- A formal language MathAbs for mathematical text that can keep the structure of the natural language text.
- A parser translating the first to the second.
- A proof-checker for MathAbs.
Our goal

- Define a small subset of mathematical English with some rich linguistic features.
- A formal language MathAbs for mathematical text that can keep the structure of the natural language text.
- A parser translating the first to the second.
- A proof-checker for MathAbs.

In the current prototype the first three points are working.
Theorem

Prove that $\sqrt{2}$ is an irrational number.

Proof: suppose that $\sqrt{2}$ is a rational number. We can assume that $\sqrt{2} = a/b$ by the definition of rational number, where $a$ and $b$ are non zero integers with no common factor. Thus, $\sqrt{2} \cdot b = a$. We get $2 \cdot b^2 = a^2$ - (i) by squaring both sides. Since $b^2$ and $a^2$ are non zero integers, we conclude that $a^2$ is even. By the last deduction, $a$ is even. We can write $a = 2 \cdot c$ by the definition of even numbers, where $c$ is an integer. We get $2 \cdot b^2 = (2 \cdot c)^2 = 4 \cdot c^2$ by substituting the value of $a$ into equation (i). Dividing both sides by 2, yields $b^2 = 2 \cdot c^2$. Because $b$ is a multiple of 2, we conclude that $b^2$ is even. If $a$ and $b$ are even, then they have a common factor. It is a contradiction.
"Theorem" show sqrt (2): Irrational;
"Proof" assume sqrt (2): Rational
let a : ZZ let b : ZZ assume positive (a) and positive (b)
  assume no_cmn_factor (a, b) assume sqrt (2)= a / b
  by def Rational
deduce sqrt(2)* b = a
deduce 2 * b ^ 2 = a ^ 2 by oper
  do_take_square_from_equation (sqrt(2)* b = a)
deduce b ^ 2 : ZZ and (a ^ 2 : ZZ and
    (positive (b ^ 2) and positive (a ^ 2)))
deduce even (a ^ 2) by form b ^ 2 : ZZ and
    (a ^ 2 : ZZ and (positive (b ^ 2) and positive (a ^ 2)))
deduce even (a)by form even (a ^ 2)
let c : ZZ
assume a = 2 * c by def Even_Number
deduce 2 * b ^ 2 = (2 * c)^ 2 = 4 * c ^ 2
  by oper do_substitution_in_equation(a,2 * b ^ 2 = a ^ 2)
deduce b ^ 2 = 2 * c ^ 2 by oper
  do_divisor_equation_by(2 * b ^ 2 = (2 * c)^ 2 = 4 * c ^2,2)
deduce multiple_of ([b], 2)
deduce even (b ^ 2) by form multiple_of ([b], 2)
assume even (a) and even (b) show one_cmn_factor (a, b)
show _|_ trivial ;
MathAbs

MathAbs is parametrized by $\text{hint}$ and expression/formula.

Describe the proof by describing how the sequent changes:

\[
\text{proof} := \\
| \text{assume formula \[id\] \[hint\] proof} \\
| \text{let \[id\] : type \[hint\] proof} \\
| \text{deduce formula \[id\] \[hint\] proof} \\
| \text{show formula \[hint\] proof} \\
| \text{trivial \[hint\]} \\
| \{ \text{proof; ... } \} \\
| \text{search \[id\] : type \[hint\] proof} \\
| \text{take \[id\] = expression \[hint\] proof}
\]

Semantics: take an initial sequent and build a proof.
MathAbs: detailed examples

assume A assume B show C ...

\[ \Gamma, A, B \vdash B \]
\[ \Gamma, A, B \vdash A \rightarrow B \rightarrow C \]
\[ \Gamma, A \vdash A \rightarrow B \rightarrow C \]
\[ \Gamma \vdash A \rightarrow B \rightarrow C \]

let x : N assume A deduce B show C ...

let x : N assume A
{ show B trivial ; assume B show C ... }

\[ \Gamma, x : N, A, B \vdash C \]
\[ \Gamma, x : N, A \vdash B \]
\[ \Gamma, x : N, A, B \vdash \forall x : N (A \rightarrow C) \]
\[ \Gamma, x : N, A \vdash \forall x : N (A \rightarrow C) \]
\[ \Gamma \vdash \forall x : N (A \rightarrow C) \]
Sentence level parser written using A. Ranta’s GF

Four levels:

- symbolic expressions and formulas.
- natural language expressions and formulas.
- proofs.
- text structure.

A lot of rephrasing is allowed ...
Sentence level parser written using A. Ranta’s GF

Four levels:

- symbolic expressions and formulas.
- natural language expressions and formulas.
- proofs.
- text structure.

A lot of rephrasing is allowed ... but far not enough! The user is almost always outside of the scope!
Haskell context and MathAbs building

- Uses a zipper to build the MathAbs proof tree.
- Using a context with all expressions, hypothesis, conclusion, ...
- To solve anaphora such as: “it”, “these integers”, “by the last hypothesis”, ...
- Distinguish collective versus distributive reading: “x and y are equal/positive”.
- Deals with implicitly structured proof by case (unfinished).
Statements - Logical formulas:

one_cmn_factor(a, b) :===> (Goal,12,DeclMan)
even(a) and even(b) :===> (Hypothesis,12,DeclMan)
even(b^2) :===> (Deduction,11,DeclMan)
multiple_of([b], 2) :===> (Justification,11,DeclMan)
multiple_of([b], 2) :===> (Deduction,11,DeclAuto)
b^2 = 2 * c^2 :===> (Deduction,10,DeclMan)
2 * b^2 = (2 * c)^2 = 4 * c^2 :===> (Deduction,9,DeclMan)
c : ZZ :===> (Hypothesis,8,DeclMan)
a = 2 * c :===> (Hypothesis,8,DeclMan)
even(a) :===> (Deduction,7,DeclMan)
even(a^2) :===> (Deduction,6,DeclMan)
b^2 : ZZ and (a^2 : ZZ) and (positive(b^2) and positive(a^2)) :===> (Justification,6,DeclMan)
b^2 : ZZ and (a^2 : ZZ and (positive(b^2) and positive(a^2))) :===> (Deduction,6,DeclAuto)
2 * b^2 = a^2 :===> (Deduction,5,DeclMan)
sqrt(2)*b = a :===> (Deduction,4,DeclMan)
a : ZZ and (b : ZZ and (positive(a) and positive(b))) and no_cmn_factor(a, b) :===> (Hypothesis,3,DeclMan)
sqrt(2) = a / b :===> (Hypothesis,3,DeclMan)
sqrt(2) : Rational :===> (Hypothesis,2,DeclMan)
“if $x = y$, then it is positive”
Here “it” means “$x$”

we deduce $x < y$, hence it is smaller than $z$ (if $y < z$).
Here “it” means “$x$”
Anaphora inside expressions

- “if $x = y$, then it is positive”
  Here “it” means “$x$”
- we deduce $x < y$, hence it is smaller than $z$ (if $y < z$).
  Here “it” means “$x$”
- we deduce $x < y$, hence it is greater than $z$ (if $z < x$).
  Here “it” should not be resolved!
Anaphora inside expressions

- “if $x = y$, then it is positive”
  Here “it” means “$x$”
- we deduce $x < y$, hence it is smaller than $z$ (if $y < z$).
  Here “it” means “$x$”
- we deduce $x < y$, hence it is greater than $z$ (if $z < x$).
  Here “it” should not be resolved!
- we deduce $x < y$, hence $z$ is smaller than this integer (if $z < x$).
  Here “this integer” could mean $y$. 
Proof Checking

A first prototype using first-order ATP ...

- Sensible to useless hypothesis.
- Sensible to useless definitions.
- Not good at equational reasoning.
- Some missing implicit hypothesis.
- Very hard to use hints.

What is possible: check the proof against a formal proof in Coq, HOL, ...

Design an incomplete ATP that solves some of the above problems.
Proof Checking

A first prototype using first-order ATP ... failed!

Reasons: ATP are
- Sensible to useless hypothesis.
- Sensible to useless definitions.
- No that good at equational reasoning.
- Some missing implicit hypothesis.
- Very hard to use hints.

What is possible: check the proof against a formal proof in Coq, HOL, ...

Design an incomplete ATP that solve some of the above problems.
Further works

- Checking MathAbs proofs against formal proofs.
- Enlarging a lot the grammar.
- Using Gf word completion (problem of rejection by Haskell).
- Dealing with meta variables.
- Really use the GF resource library.
- Merge our work with the WebAlt project (doing statement, using GF).