The Naproche system: Proof-checking mathematical texts in controlled natural language

Marcos Cramer

University of Luxembourg

18 July 2014

The Naproche Project

- The **Naproche** project (**Na**tural language **Pro**of **Che**cking) studies the **language and reasoning of mathematics** from the perspectives of logic and linguistics.

The Naproche Project

- The **Naproche** project (**Na**tural language **Pro**of **Che**cking) studies the **language and reasoning of mathematics** from the perspectives of logic and linguistics.
- Central goals of Naproche:
  - To develop a controlled natural language (CNL) for mathematical texts.

The Naproche Project

- The **Naproche** project (**Na**tural language **Pro**of **Che**cking) studies the **language and reasoning of mathematics** from the perspectives of logic and linguistics.
- Central goals of Naproche:
  - To develop a controlled natural language (CNL) for mathematical texts.
  - To implement a system, the **Naproche system**, which can check texts written in this CNL for logical correctness.

Outline

Naproche CNL

Burali-Forti paradox in the Naproche CNL

Axiom 1: There is a set $\emptyset$ such that no y is in $\emptyset$.
Axiom 2: There is no x such that $x \in x$.

Define x to be transitive if and only if for all u, v, if $u \in v$ and $v \in x$ then $u \in x$.
Define x to be an ordinal if and only if x is transitive and for all y, if $y \in x$ then y is transitive.

Theorem: There is no x such that for all u, $u \in x$ iff u is an ordinal.
Proof:
Assume for a contradiction that there is an x such that for all u, $u \in x$ iff u is an ordinal.
Let $u \in v$ and $v \in x$. Then v is an ordinal, i.e. u is an ordinal, i.e. $u \in x$.
Thus x is transitive.
Let $v \in x$. Then v is an ordinal, i.e. v is transitive. Thus x is an ordinal.
Then $x \in x$. Contradiction by axiom 2.
Qed.

Naproche proof checking

- The proof checking algorithm keeps track of a list of first-order formulae considered true, called **premises**.

Naproche proof checking

- The proof checking algorithm keeps track of a list of first-order formulae considered true, called **premises**.
- The premise list gets continuously updated during the verification process.

Naproche proof checking

- The proof checking algorithm keeps track of a list of first-order formulae considered true, called **premises**.
- The premise list gets continuously updated during the verification process.
- Each assertion is checked by an ATP based on the currently active premises.

Example

Suppose n is even. Then there is a k such that $n = 2k$. Then $n^2 = 4k^2$, so $4|n^2$.

Naproche proof checking

- The proof checking algorithm keeps track of a list of first-order formulae considered true, called **premises**.
- The premise list gets continuously updated during the verification process.
- Each assertion is checked by an ATP based on the currently active premises.

Example

Suppose n is even. Then there is a k such that $n = 2k$. Then $n^2 = 4k^2$, so $4|n^2$.

- Sentence-by-sentence proof verification:
  - $\Gamma, even(n) \vdash^{?} \exists k\ n = 2 \cdot k$

Naproche proof checking

- The proof checking algorithm keeps track of a list of first-order formulae considered true, called **premises**.
- The premise list gets continuously updated during the verification process.
- Each assertion is checked by an ATP based on the currently active premises.

Example

Suppose n is even. Then there is a k such that $n = 2k$. Then $n^2 = 4k^2$, so $4|n^2$.

- Sentence-by-sentence proof verification:
  - $\Gamma, \mathit{even}(n) \vdash^? \exists k\ n = 2 \cdot k$
  - $\Gamma, \mathit{even}(n), n = 2 \cdot k \vdash^? n^2 = 4 \cdot k^2$

Naproche proof checking

- The proof checking algorithm keeps track of a list of first-order formulae considered true, called **premises**.
- The premise list gets continuously updated during the verification process.
- Each assertion is checked by an ATP based on the currently active premises.

Example

Suppose n is even. Then there is a k such that $n = 2k$. Then $n^2 = 4k^2$, so $4|n^2$.

- Sentence-by-sentence proof verification:
  - $\Gamma, even(n) \vdash^? \exists k\ n = 2 \cdot k$
  - $\Gamma, even(n), n = 2 \cdot k \vdash^? n^2 = 4 \cdot k^2$
  - $\Gamma, even(n), n = 2 \cdot k, n^2 = 4 \cdot k^2 \vdash^? 4|n^2$

Naproche proof checking (2)

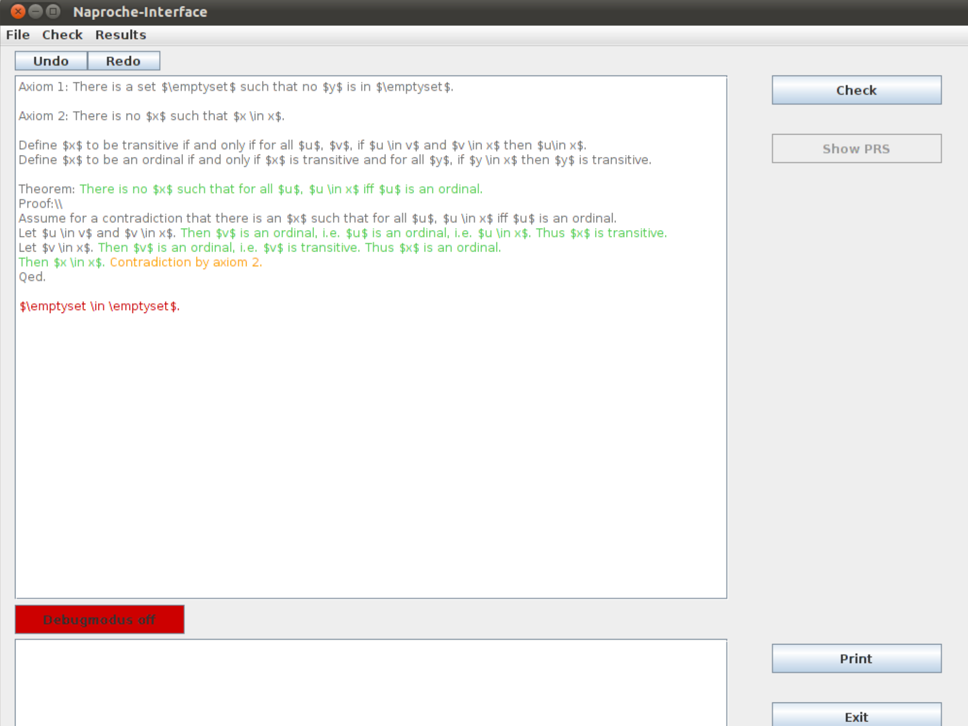
- An assumption is processed in **No-Check Mode**.

Naproche proof checking (2)

- An assumption is processed in **No-Check Mode**.
- The No-Check Mode is also used for φ and ψ in $\neg\varphi$, $\exists x\ \varphi$, $\varphi \vee \psi$ and $\varphi \rightarrow \chi$.

Naproche proof checking (2)

- An assumption is processed in **No-Check Mode**.
- The No-Check Mode is also used for φ and ψ in $\neg\varphi$, $\exists x\ \varphi$, $\varphi \vee \psi$ and $\varphi \rightarrow \chi$.
- We have proved soundness and completeness theorems for the proof checking algorithm.


Naproche-Interface
File Check Results
Undo
Redo
Axiom 1: There is a set $\emptyset$ such that no y is in $\emptyset$.
Axiom 2: There is no x such that $x \in x$.
Define x to be transitive if and only if for all u, v, if $u \in v$ and $v \in x$ then $u\in x$.
Define x to be an ordinal if and only if x is transitive and for all y, if $y \in x$ then y is transitive.
Theorem: There is no x such that for all u, $u \in x$ iff u is an ordinal.
Proof:\\
Assume for a contradiction that there is an x such that for all u, $u \in x$ iff u is an ordinal.
Let $u \in v$ and $v \in x$. Then v is an ordinal, i.e. u is an ordinal, i.e. $u \in x$. Thus x is transitive.
Let $v \in x$. Then v is an ordinal, i.e. v is transitive. Thus x is an ordinal.
Then $x \in x$. Contradiction by axiom 2.
Qed.
$\emptyset \in \emptyset$.
Debugmodus off
Check
Show PRS
Print
Exit

Outline

Dynamic Quantification

Example

Suppose n is even. Then there is a k such that $n = 2k$. Then $n^2 = 4k^2$, so $4|n^2$.

Dynamic Quantification

Example

Suppose n is even. Then there is a k such that $n = 2k$. Then $n^2 = 4k^2$, so $4|n^2$.

Example

If a space X retracts onto a subspace A, then the homomorphism $i_* : \pi_1(A, x_0) \to \pi_1(X, x_0)$ induced by the inclusion $i : A \hookrightarrow X$ is injective.

A. Hatcher: Algebraic topology (2002)

Dynamic Quantification (2)

Solution: **Dynamic Predicate Logic** (**DPL**) by Groenendijk and Stokhof

Dynamic Quantification (2)

Solution: **Dynamic Predicate Logic** (**DPL**) by Groenendijk and Stokhof

Example

If a farmer owns a donkey, he beats it.

PL: $\forall x \forall y\ (\mathit{farmer}(x) \land \mathit{donkey}(y) \land \mathit{owns}(x, y) \rightarrow \mathit{beats}(x, y))$

DPL: $\exists x\ (\mathit{farmer}(x) \land \exists y\ (\mathit{donkey}(y) \land \mathit{owns}(x, y))) \rightarrow \mathit{beats}(x, y)$

Implicit dynamic function introduction

Suppose that, for each vertex v of K, there is a vertex $g(v)$ of L such that $f(st_K(v)) \subset st_L(g(v))$. Then g is a simplicial map $V(K) \to V(L)$, and $|g| \simeq f$.

M. Lackenby: Topology and groups (2008)

Implicit dynamic function introduction

Suppose that, for each vertex v of K, there is a vertex $g(v)$ of L such that $f(st_K(v)) \subset st_L(g(v))$. Then g is a simplicial map $V(K) \to V(L)$, and $|g| \simeq f$.

M. Lackenby: Topology and groups (2008)

- Solution: **Typed Higher-Order Dynamic Predicate Logic** (**THODPL**)

THODPL

- There can be a complex term after a quantifier:
  1. $\forall x\ \exists f(x)\ R(x, f(x))$

THODPL

- There can be a complex term after a quantifier:
  1. $\forall x \, \exists f(x) \, R(x, f(x))$
  2. $\forall x \, \exists y \, R(x, y)$
  3. $\exists f \, \forall x \, R(x, f(x))$
- 1 has the same truth conditions as 2.

THODPL

- There can be a complex term after a quantifier:
  1. $\forall x \; \exists f(x) \; R(x, f(x))$
  2. $\forall x \; \exists y \; R(x, y)$
  3. $\exists f \; \forall x \; R(x, f(x))$
- 1 has the same truth conditions as 2.
- But unlike 2, 1 dynamically introduces the function symbol f, and hence turns out to be equivalent to 3.

THODPL in proof checking

- Quantification over a complex term is checked in the same way as quantification over a variable:

For each vertex v of K, there is a vertex $g(v)$ of L such that $f(st_K(v)) \subset st_L(g(v))$. Then g is a simplicial map $V(K) \to V(L)$.

$\Gamma, \text{vertex}(v, K) \vdash^{?} \exists w\, (\text{vertex}(w, K) \wedge f(st_K(v)) \subset st_L(w))$

THODPL in proof checking

- Quantification over a complex term is checked in the same way as quantification over a variable:

For each vertex v of K, there is a vertex $g(v)$ of L such that $f(st_K(v)) \subset st_L(g(v))$. Then g is a simplicial map $V(K) \to V(L)$.

$$\Gamma, \text{vertex}(v, K) \vdash^? \exists w\, (\text{vertex}(w, K) \wedge f(st_K(v)) \subset st_L(w))$$

- However, it dynamically introduces a new function symbol.
- The premise corresponding to this quantification gets skolemized with this new function symbol:

$$\Gamma, \forall v\, (\text{vertex}(v, K) \to (\text{vertex}(g(v), K) \wedge f(st_K(v)) \subset st_L(g(v)))) \\ \vdash^? \text{simplicial\_map}(g, V(K), V(L))$$

THODPL in proof checking (2)

- The theorem prover does not need to prove the existence of a function, but its existence may nevertheless be assumed as a premise.

THODPL in proof checking (2)

- The theorem prover does not need to prove the existence of a function, but its existence may nevertheless be assumed as a premise.
- Similarly, $\forall x\ \exists f(x)\ R(x, f(x))$ is proof-checked in the same way as $\forall x\ \exists y\ R(x, y)$, but as a premise it has the force of $\exists f\ \forall x\ R(x, f(x))$.

Outline

Undefined terms

- Mathematical texts can involve **potentially undefined terms** like $\frac{1}{x}$.

Undefined terms

- Mathematical texts can involve **potentially undefined terms** like $\frac{1}{x}$.
- Such terms arise by applying **partial functions** to ungrounded terms.

Undefined terms

- Mathematical texts can involve **potentially undefined terms** like $\frac{1}{x}$.
- Such terms arise by applying **partial functions** to ungrounded terms.
- First-order logic has no means for handling partial functions and potentially undefined terms.

Undefined terms

- Mathematical texts can involve **potentially undefined terms** like $\frac{1}{x}$.
- Such terms arise by applying **partial functions** to ungrounded terms.
- First-order logic has no means for handling partial functions and potentially undefined terms.
- We make use of **presupposition theory** from formal linguistics for solving this problem.

Presuppositions

- A **presupposition** of some utterance is an implicit assumption that is taken for granted when making the utterance and needed for its interpretation.

Presuppositions

- A **presupposition** of some utterance is an implicit assumption that is taken for granted when making the utterance and needed for its interpretation.
- Presuppositions are triggered by certain lexical items called **presupposition triggers**, e.g. "the", "to know", "to stop", "still".

Presuppositions

- A **presupposition** of some utterance is an implicit assumption that is taken for granted when making the utterance and needed for its interpretation.
- Presuppositions are triggered by certain lexical items called **presupposition triggers**, e.g. "the", "to know", "to stop", "still".

Example

He stopped beating his wife.

Presupposition in mathematical texts

- Most presupposistion triggers are rare or absent in mathematical texts, e.g. "to know", "to stop" and "still".

Presupposition in mathematical texts

- Most presupposistion triggers are rare or absent in mathematical texts, e.g. "to know", "to stop" and "still".
- Definite descriptions do appear, e.g. "the smallest natural number n such that $n^2 - 1$ is prime".

Presupposition in mathematical texts

- Most presupposistion triggers are rare or absent in mathematical texts, e.g. “to know”, “to stop” and “still”.
- Definite descriptions do appear, e.g. “the smallest natural number n such that $n^2 - 1$ is prime”.
- A special mathematical presupposition trigger: Expressions denoting partial functions, e.g. “/” and “$\sqrt{\ }$”

Proof checking algorithm with presuppositions

Presuppositions also have to be checked in No-Check Mode.

Proof checking algorithm with presuppositions

Presuppositions also have to be checked in No-Check Mode.

Example 1

Assume that B contains $\sqrt{y}$.

Proof checking algorithm with presuppositions

Presuppositions also have to be checked in No-Check Mode.

Example 1

Assume that B contains $\sqrt{y}$.

$\Gamma \vdash^{?} y \geq 0$

Proof checking algorithm with presuppositions

Presuppositions also have to be checked in No-Check Mode.

Example 1

Assume that B contains $\sqrt{y}$.

$\Gamma \vdash^{?} y \geq 0$

Example 2

B does not contain $\sqrt{y}$.

Proof checking algorithm with presuppositions

Presuppositions also have to be checked in No-Check Mode.

Example 1

Assume that B contains $\sqrt{y}$.

$\Gamma \vdash^? y \geq 0$

Example 2

B does not contain $\sqrt{y}$.

$\Gamma \vdash^? y \geq 0$
$\Gamma, y \geq 0 \vdash^? \neg\sqrt{y} \in B$

Outline

Conclusion

- We have developed a controlled natural language for mathematical texts.

Conclusion

- We have developed a controlled natural language for mathematical texts.
- The Naproche system can check the correctness of texts written in this language.

Conclusion

- We have developed a controlled natural language for mathematical texts.
- The Naproche system can check the correctness of texts written in this language.
- Interesting theoretical work linking mathematical logic and formal linguistics