Mathematical Proofs and Software specifications in Natural Languages

Muhammad Humayoun

Laboratoire de Mathématiques (LAMA)
Université de Savoie
mhuma@univ-savoie.fr

March 29, 2008

Types 08, Turin, Italy

Co-Supervised by:

Christophe Raffalli
Laboratoire de Mathématiques (LAMA)
Université de Savoie
France

Aarne Ranta
Department of Computer Science
Chalmers University of Technology
Sweden

Introduction

What?

- Aim - to develop a controlled natural language:
  - For writing mathematical proofs and software specifications
  - Having large coverage
  - Grammatically correct
  - Interactively

Why?

- A normal practise: Writing mathematical proofs and software specifications in plain natural language
  - Undergraduate students
  - Domain experts
    (Software designers, programmers, engineers and mathematicians)
  - RFC's, patents ...

What is Controlled Language?

Controlled language

- A **subset of natural language**
- Grammar and dictionary have been restricted
- To reduce/eliminate ambiguity and complexity

- Connection/bridge between natural and formal languages

Why Mathematical Proofs and Specifications Together?

- Pose similar problems
- **A formal software specification** could be seen as a large statement = (smaller statements + logical connectives)

[1] *The arbiter does not send anything to ports that have not been assigned a mac except mac_grants and pings.*

only_talk_to_ports_macs t =
$(\forall$n p msg.)
  (t n = A_A2H(p,msg)) $\bigwedge$
  $\neg$ ($\exists$mac.msg = A2H_MAC_GRANT mac) $\bigwedge$
  $\neg$ ($\exists$pingid.msg = A2H_PING pingid) $\Longrightarrow$
  p $\in$ rng ((port_of_mac t n)))

- Therefore, any formal language capable of writing mathematical proofs is also capable of writing software specifications
- We just need a language of statement along with the language of Proofs

[1] A. Biltcliffe, M. Dales, S. Jansen, T. Ridge, P. Sewell 2006, An Optical Packet-Switch MAC in HOL.[1]

Why Mathematical Proofs and Specifications Together?

- Pose similar problems
- **A formal software specification** could be seen as a large statement = (smaller statements + logical connectives)

[1] *The arbiter does not send anything to ports that have not been assigned a mac except mac_grants and pings.*

only_talk_to_ports_macs t =
$(\forall$n p msg.)
  (t n = A_A2H(p,msg)) $\bigwedge$
  $\neg$ ($\exists$mac.msg = A2H_MAC_GRANT mac) $\bigwedge$
  $\neg$ ($\exists$pingid.msg = A2H_PING pingid) $\Longrightarrow$
  p $\in$ rng ((port_of_mac t n)))

- Therefore, any formal language capable of writing mathematical proofs is also capable of writing software specifications
- We just need a language of statement along with the language of Proofs

[1] A. Biltcliffe, M. Dales, S. Jansen, T. Ridge, P. Sewell 2006, An Optical Packet-Switch MAC in HOL.[1]

Our consideration

The case studies

1. Mathematical proofs found in university math books
2. Software specifications: we'll probably collaborate with Herman Geuvers (Formalizing Hybrid Systems in Coq)

Today's talk

- Parsing & translation of mathematical proofs written in English

Language of ordinary mathematics

- Mathematical register already a little bit controlled text
  - Text structure for definitions, theorems, proofs
  - Sentence structure
    - Formal expressions (formulas)
      - x + y = 4 ; A & B
    - Expressions in natural language using arithmetic, logic and predicates
      - it is not the case that x is true ; the sum of x, y and z
  - Diagrams

A distinction between:

- Mathematical proofs
- Other domain/theories

Domains/theories in our consideration:

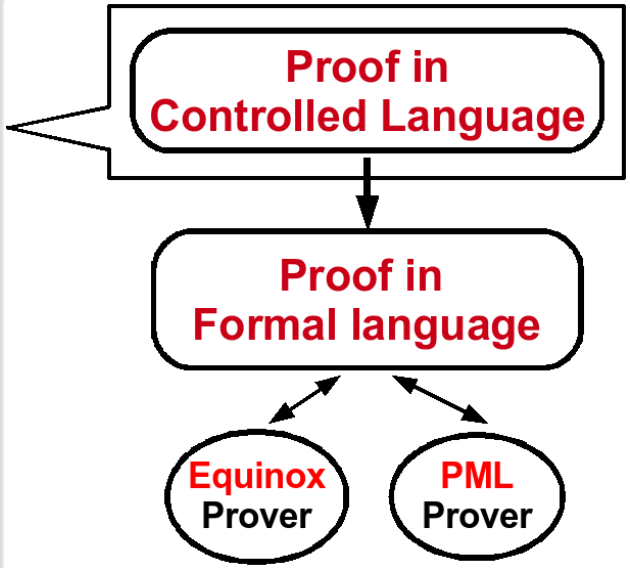
- Logic
- Arithmetic
- Specifications
- .....

The System (Example 1)

Controlled language

Theorem.
Let A and B be sets. If (A ∪ B) = (A ∩ B) then A ⊆ B .

Proof .
Assume A ∪ B = A ∩ B. Assume that x ∈ A. Since A ⊆ A ∪ B, then x ∈ A ∪ B. Similarly, it is clear that x ∈ A ∩ B, because A ∪ B = A ∩ B. Finally, as A ∩ B ⊆ B, we have x ∈ B. This concludes the proof.



Equinox: An automated theorem prover for pure first-order logic with equality[2]

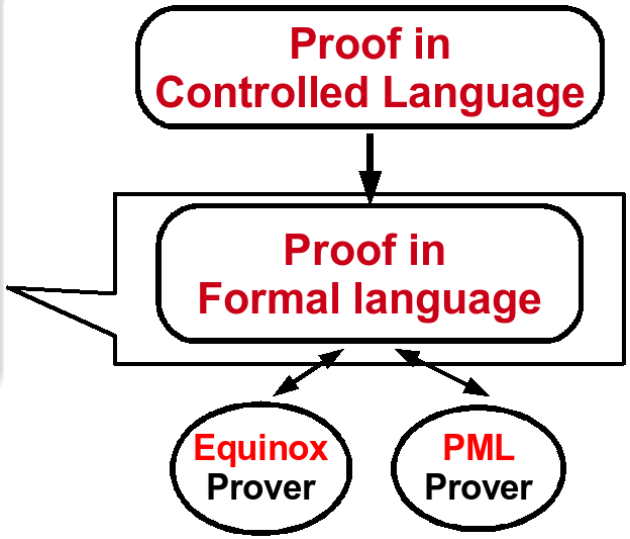
PML: A proof assistant for mathematics and Software Specifications (in development phase)[3]

The System (Example 1)

Formal language

Theorem
A, B : Set $((A \cup B) = (A \cap B)) \Rightarrow A \subseteq B$
Proof

- assume $(A \cup B) = (A \cap B)$
- assume $x \in A$
- assume $A \subseteq A \cup B$ by $x \in A \cup B$
- assume $x \in A \cap B$ by $(A \cup B) = (A \cap B)$
- assume $(A \cap B) \subseteq B$ by $x \in B$
- trivial ;



- Developed in DemoNat project by Patrick Thévenon
- A little bit of modification
- Semantics - Proof tree could be built by using any *Rule*

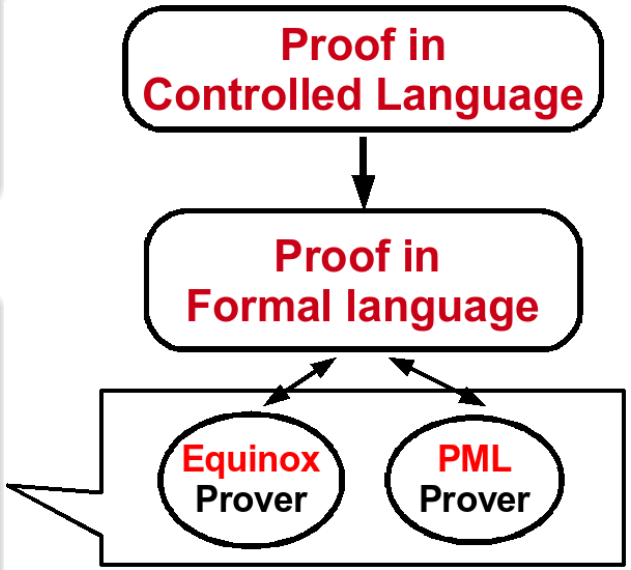
The System (Example 1)

TPTP syntax for Equinox

- fof(conj1,conjecture,((cA ∪ cB)= (cA ∩ cB)) ⇒ (cx ∈ cA ⇒ (cA ⊆ cA ∪ cB ⇒ (cx ∈ cA ∩ cB ⇒ ((cA ∩ cB)⊆ cB ⇒ (((cA ∪ cB)= (cA ∩ cB)) ⇒ cA ⊆ cB))))))).
- fof(conj2,conjecture,(((cA ∪ cB)= (cA ∩ cB)) ⇒ (cx ∈ cA ⇒ (cx ∈ cA ∪ cB ⇒ (cA ⊆ cA ∪ cB ⇒ (((cA ∪ cB)= (cA ∩ cB)) ⇒ (cx ∈ cA ∩ cB ⇒ (cx ⊆ cB ⇒ ((cA ∩ cB)⊆ cB ⇒ (((cA ∪ cB)= (cA ∩ cB)) ⇒ cA ⊆ cB)))))))))⇒ (((cA ∪ cB)= (cA ∩ cB)) ⇒ cA ⊆ cB)).

Theorem A
Proof assume B assume C assume D assume E trivial

- fof(conj1,conjecture,
  (cB ⇒ (cC ⇒ (cD ⇒ (cE ⇒ cA))))).
- fof(conj2,conjecture,
  ((cB ⇒ (cC ⇒ (cD ⇒ (cE ⇒ cA)))) ⇒ cA)).



Translation to PML

Not yet...

Formal language

⟨*PDocument*⟩ ::= ⟨*ListPTheorem*⟩
⟨*PTheorem*⟩ ::= `Theorem` ⟨*Formula*⟩ `;` `Proof` ⟨*Proof*⟩ `;`
⟨*Proof*⟩ ::= ⟨*PRule*⟩
⟨*PRule*⟩ ::= `trivial`
| `show` ⟨*Formula*⟩ ⟨*Proof*⟩
| `unfinished`
| ⟨*Assignment*⟩ ⟨*PRule*⟩
| `Split` ⟨*ListPRule*⟩
⟨*Assignment*⟩ ::= `let` ⟨*ListIdent*⟩ `:` ⟨*Ident*⟩
| `let` ⟨*ListIdent*⟩ `:` ⟨*Formula*⟩
| `let` ⟨*ListIdent*⟩
| `assume` ⟨*Formula*⟩
| `assume` ⟨*Formula*⟩ `by` ⟨*Hint*⟩

Example 2:

Controlled language

Theorem.
Prove that $(A \rightarrow B \rightarrow C) \leftrightarrow ((A \,\&\, B) \rightarrow C)$ holds.
Proof.
Suppose A, B and C are propositions. We have two cases to prove.

- Case 1: We prove that $(A \rightarrow B \rightarrow C) \rightarrow ((A \,\&\, B) \rightarrow C)$ holds. Assume $(A \rightarrow B \rightarrow C)$ – (i) and $A \,\&\, B$ – (ii). We show C. We have three cases. **First**, we show A. It is trivial. **Second**, we show B. It is trivial. **Third**, we assume A and B. It is trivial.
- Case 2: We prove that $((A \,\&\, B) \rightarrow C) \rightarrow (A \rightarrow B \rightarrow C)$ holds. We assume $((A \,\&\, B) \rightarrow C)$, A and B. We show C . It is trivial. This was the last case .

Example 2:

Formalism

Theorem $((A \rightarrow (B \rightarrow C)) \leftrightarrow (A \,\&\, B \rightarrow C))$;
Proof

- let A, B, C : Prop
- {
  - show $((A \rightarrow B) \rightarrow C) \rightarrow ((A \,\&\, B) \rightarrow C)$
  - assume $(A \rightarrow (B \rightarrow C)$ i
  - assume $(A \,\&\, B)$ ii
  - show C
  - {
    - show A trivial;
    - show B trivial ;
    - assume A assume B trivial
  - }
  - show $((A \,\&\, B) \rightarrow C) \rightarrow ((A \rightarrow B) \rightarrow C)$
  - assume $(A \,\&\, B \rightarrow C)$
  - assume A assume B show C trivial
- }

Example 3

Controlled language (Proof by contradiction)

Theorem.
Prove that $2^{1/2}$ is irrational.
Proof.

- We assume that $2^{1/2}$ is rational and obtain a contradiction.
- Let a and b are non zero integers with no common factor.
- Let ($2^{1/2}$) = (a / b) by the definition of rational number.
- Thus, (b*$2^{1/2}$) = a.
- We get $2*b^2 = a^2$ – (i) by squaring both sides.
- a^2 is even which implies that a is even by the fact that b^2 and a^2 are non zero integers and a is a multiple of 2.
- So we can write a = 2*c, where c is an integer.
- We get $2*b^2 = (2 * c)^2 = 4*c^2$ by substitution into the equation (i).

Example 3

- Dividing both sides by 2, yields $b^2 = 2^*c^2$.
- b^2 is even which implies that b is even by the fact that b is a non zero integer and b is a multiple of 2.
- If a and b are even, then a and b have a common factor.
- It is a contradiction. QED

Implementation details

Controlled langauge

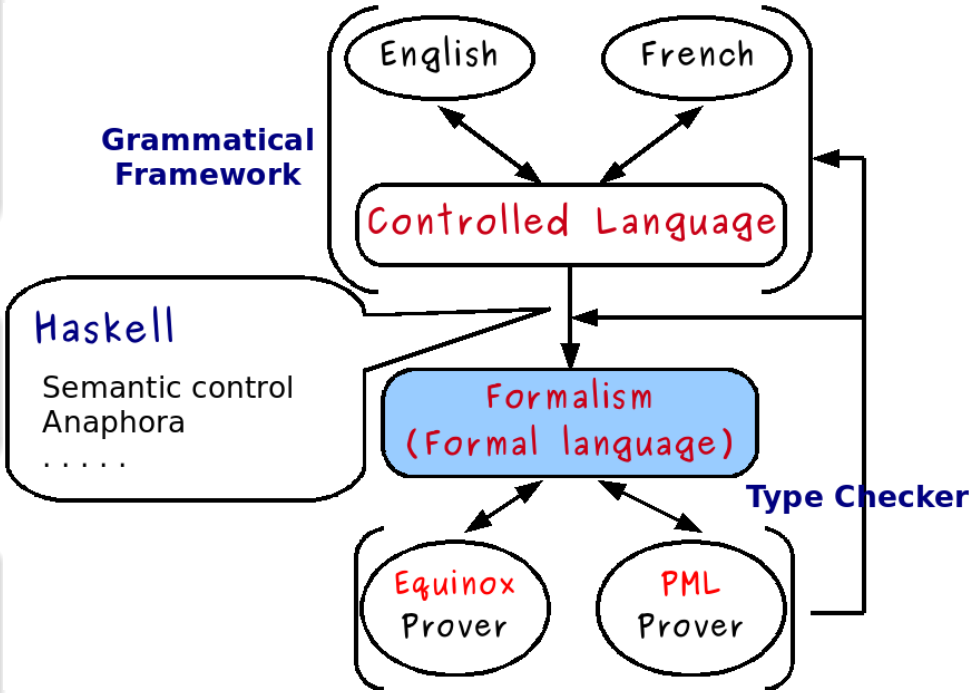
- Grammatical Framework **(GF)** [4]
  - for the development of grammar and lexicon

Semantic control

- From host system - Haskell

Translation to a specific prover

- From formal language to the language of Prover - Haskell

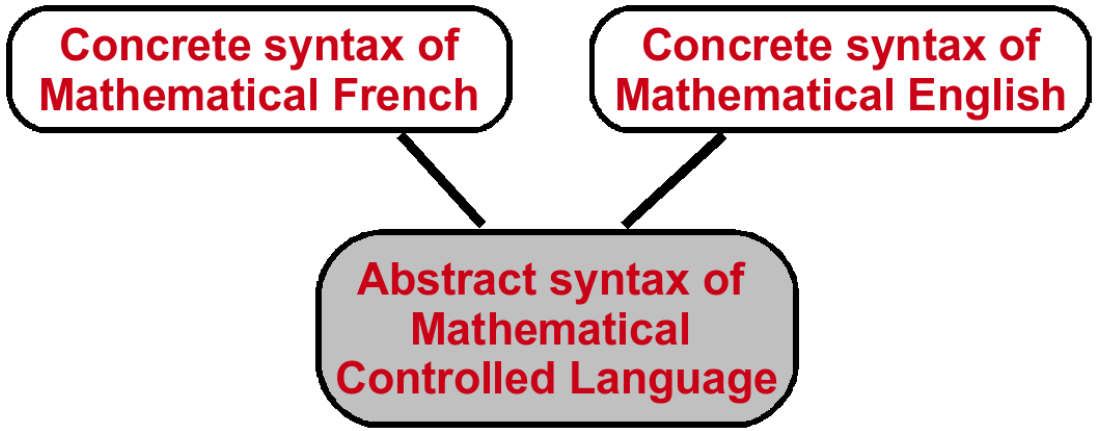


Overall picture

Controlled language (1)

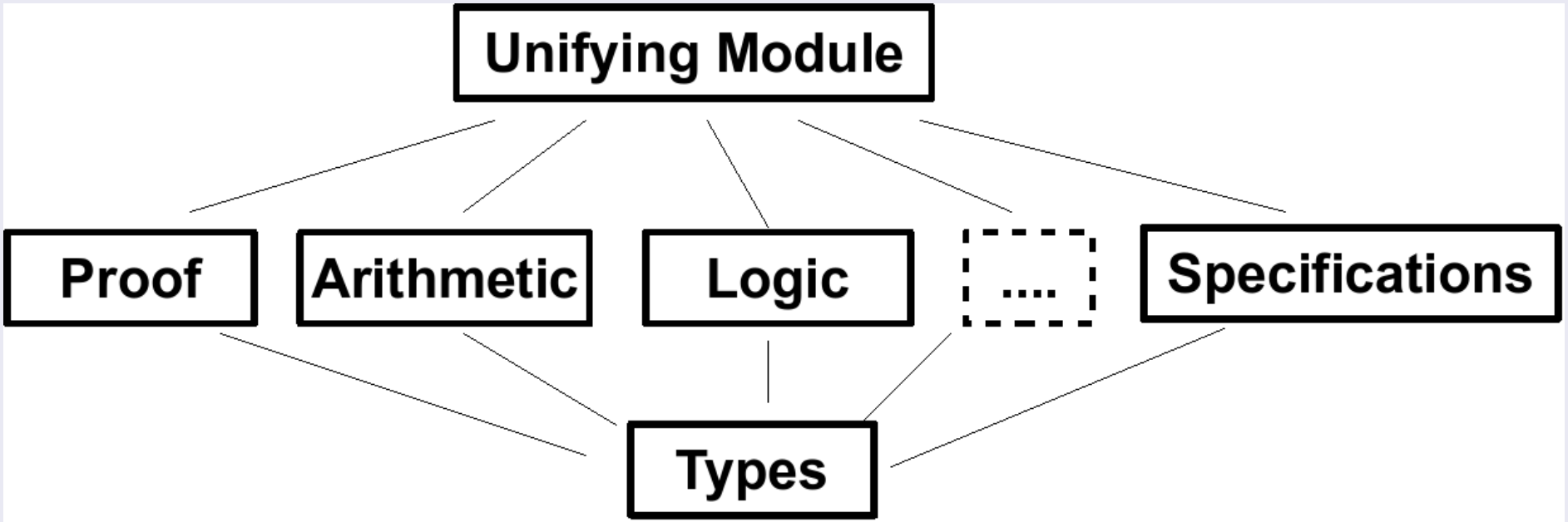
Grammatical Framework (GF)

- Mostly developed at Chalmers University, Sweden by Aarne Ranta
- Grammar formalism based on type theory
- Special purpose programming language for defining grammars
- Grammar = The Abstract syntax + Concrete syntax
- A good framework to defining interlinguas & translations



Controlled language (2)

Grammar Engineering



Some example sentences

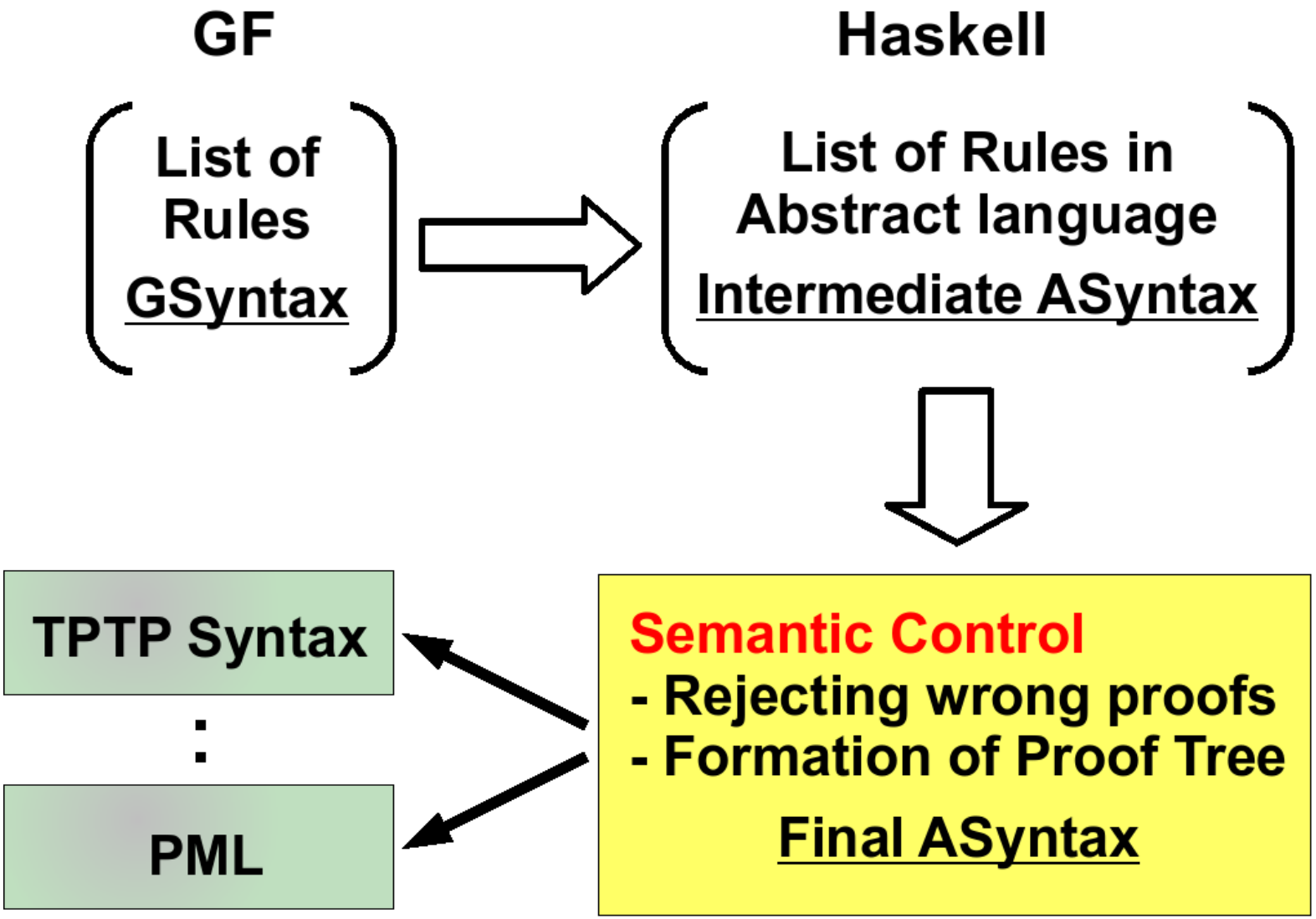
Arithmetic

- A intersects B. A belongs to B. A is a subset of B.
- x and y have common factor. 1/zero/x is even/odd/rational/integer.
- x/zero, y and z are even integers.
- x is greater or equal to y. x is at minimum y.
- x is less or equal to y. x is at least y.
- The multiplication of 1/zero/A, B and C. We multiply A, B and C.

Arithmetic+Logic

- It is not the case that x is an integer.
- If a and b are even , then a and b have a common factor.
- b^2 is even which implies that b is even by the fact that b is a non zero integer and b is a multiple of 2.

Semantic Control (1)



Semantic Control (2)

$$\text{GSyntax} \Longrightarrow \text{ASyntax}$$
$$\text{many} \Longrightarrow \text{one}$$

GF

- we deduce A by A & B
- we deduce $x \in A \cup B$ by the fact that $A \subseteq A \cup B$
- Since $A \subseteq A \cup B$, then $x \in A \cup B$
- it is clear that $x \in A \cup B$, as $A \subseteq A \cup B$
- it is clear that $x \in A \cup B$, because $A \subseteq A \cup B$
- as A, so B
- .....

Formal language

- assume Formula by Formula - *(assume A by A & B)*

Current and Future work

Current:

- + Fairly complete Proof grammar suitable for Propositional Logic
- - Looks artificial but readable
- Essential for the other domains (Arithmetic, Set, Specification)
- Developing the grammar and lexicon for Arithmetic

Future:

- The use of GF resource grammar
  - x is even. x and y are even.
  - Even x = ss (x.s ++ be.s!x.n ++"even");
  - Even x = mkS (mkCl x (mkA "even")) ;
- Anaphoric resolution
- Specification

Related Work

- Y. Coscoy. A natural language explanation of formal proofs
  - Generation
- K. Johannisson. OCL (Formal and Informal Software Specifications)
  - Syntax editing and generation

References

1. A. Biltcliffe, M. Dales, S. Jansen, T. Ridge, P. Sewell 2006. Rigorous Protocol Design in Practice: An Optical Packet-Switch MAC in HOL. *The 14th IEEE International Conference on Network Protocols*
2. K. Claessen 2005. Equinox http://www.cs.chalmers.se/∼koen/folkung
3. C. Raffalli 2007. *PML: a new proof assistant* conférence au workshop Types, Italy. http://www.lama.univ-savoie.fr/∼raffalli/pml
4. A. Ranta 2004. Grammatical Framework: A Type-Theoretical Grammar Formalism. *Journal of Functional Programming*, 14(2), pp. 145-189.