Software Specifications and Mathematical Proofs in Natural Languages

Muhammad Humayoun

mhuma@univ-savoie.fr

Laboratoire de Mathématiques (LAMA), Université de Savoie, France

Department of Computer Science, Chalmers University of Technology, Sweden

Supervisors: Christophe Raffalli (LAMA) and Aarne Ranta (Chalmers)

Région RhôneAlpes

Cluster de recherche ISLE Rhône-Alpes

Abstract: This project is an attempt to make a connection between formal and natural languages. We are developing a controlled natural language having large coverage, which will be good enough for writing Software Specifications and Mathematical Proofs interactively.

Introduction

- **A normal practice:** Writing Software specifications & Mathematical proofs in plain natural language
- **Problem:** rich, complex, and ambiguous
- **Solution:** replacing it with a rich formalism (formal language)
- **Formalisms:** understood by model checkers or theorem provers
- Precise, accurate and clear
- Not easily understood by domain experts (Software designers, programmers, engineers and Mathematicians)

Why combine Proofs and Specifications

- Formal languages to write Mathematical Proofs are often used to write logical statements
- **Software Specifications** written under some formal language could be seen as **very large statement which is formed by small statements** with the help of logical connectives
- Hence, any formal language capable of writing mathematical proofs is also capable of writing software specifications
- A **controlled natural language** on top of formalism: the best of two worlds i.e. natural languages and formal languages.

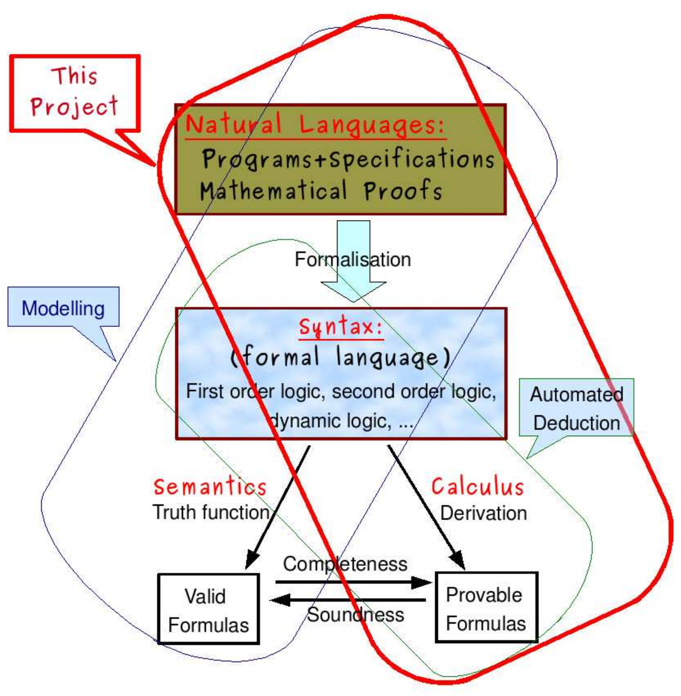

Figure 1: *An overview of the field*

The case studies

1. Software specifications for a network protocol, the Media access protocol (MAC) for a high-capacity optically-switched network: the SWIFT network[1]. The protocol design is expressed in higher-order logic by using HOL which is a general-purpose, automated proof assistant. Further the conformance test is performed between the specification and two implementations of the protocol
   - An NS-2 simulation model
   - The VHDL code of the network hardware
2. Mathematical proofs found in university math books

With these two case studies, we are trying to develope a controlled language large and expressive enough to write Specifications and Proofs.

Issues in Natural Languages

- Anaphoric resolution, Disambiguation, Indexicality, Complexity, . . .

Solution

- Introducing a Controlled language
- Applying modularity in the text
- An Intelligent text editor enforcing a user to write clear and plain language

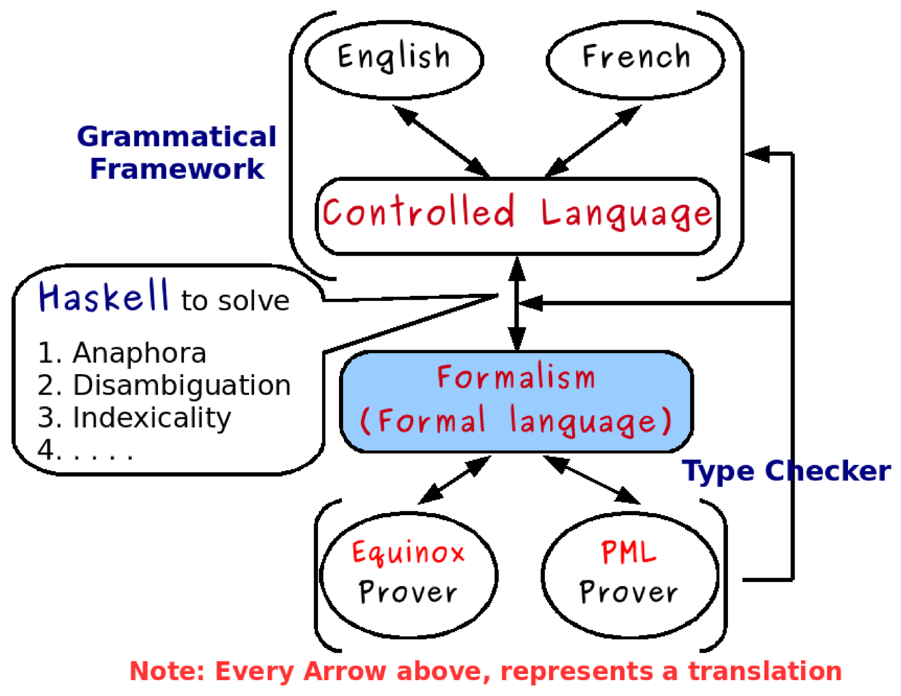

Figure 2: *Overall picture of the project*

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

PML: An under development proof assistant for mathematics and Software Specifications[3]

Controlled language

- A **subset of natural language** whose grammar and dictionary have been restricted in order to reduce or eliminate both ambiguity and complexity.
- We use grammar and lexicon which is type-theoretic; containing constants, types, & definitions

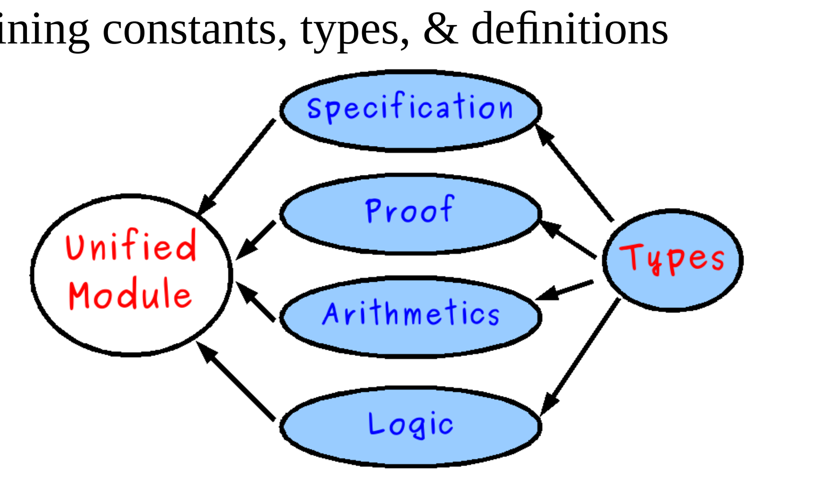

Figure 3: *Controlled Language Grammar Architecture*

What Framework?

1. Grammatical Framework (GF)[4] to a large extent

- Grammar formalism based on type theory
- A framework to defining interlingual translations (Grammar = **Abstract** syntax + **Concrete** syntax)
  - **Abstract syntax** = semantic conditions (correct syntactic structures / trees)
  - **Concrete syntax** = abstract syntax into strings along-with the grammatical features (and back, by reversibility)

2. Functional Programming language Haskell

Software Specifications - Example

A Document contains a list of Spec Modules

Example 1:

Controlled language:

The arbiter can send a message on different ports with the same ping only if the interval between two consecutive pings is greater or equal to MIN_PING_REUSE_TIME.

HOL:

pingids_not_reused_too_soon t =
($\forall$n p p' pingid n'.
  (t n = A_A2H(p,A2H_PING(pingid))) $\wedge$
  (t n' = A_A2H(p',A2H_PING(pingid))) $\wedge$
  (n' > n) $\Longrightarrow$
  a_time t n' – a_time t n $\geq$ MIN_PING_REUSE_TIME)

Example 2:

Controlled language:

The arbiter does not send anything to ports that have not been assinged a mac except mac_grants and pings. **HOL:**

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

Mathematical Proofs - Example

A Document contains a list of Theorems

Controlled language:

Theorem.
Prove that ((A $\rightarrow$ (B $\rightarrow$ C)) $\leftrightarrow$ (A & B $\rightarrow$ C)) holds.

Proof. Suppose A, B and C are propositions. First we prove left to right implication.
We assume (A $\rightarrow$ (B $\rightarrow$ C)) —(1) and (A & B) —(2).
From equation (1), we can deduce C because (2) implies both A and B.
For converse, we assume (A & B $\rightarrow$ C), A and B. By last three assumptions, it is clear that C holds.

Formal language:

Theorem ((A $\rightarrow$ (B $\rightarrow$ C)) $\leftrightarrow$ (A & B $\rightarrow$ C))
Proof
{
assume (A $\rightarrow$ (B $\rightarrow$ C) (1) assume (A & B) (2) show C
  {
  show A trivial by (2),
  show B trivial by (2),
  assume A assume B trivial by (1)
  } ,
assume (A & B $\rightarrow$ C) (3) assume A (4) assume B (5) show C trivial by (3) (4) (5)
}

Related work

K. Johannisson's PhD Thesis: Formal and Informal Software Specifications, June 2005 Chalmers Sweden.

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.