From Natural Language Proofs to Correct Programs

Olga Petrovska (843356@swansea.ac.uk)
Swansea University

HOW NOT TO BUILD "A SHAKY PIER"?

Usually software development involves multiple rounds of testing and bug fixing. This process, however, is prone to human error. How can one be sure that the software is correct and that it will not fail?

The aim of my research is finding out an efficient way of extracting programs from natural language proofs. This is an alternative approach to software development, where correctness properties of software specifications are formally proved and software is extracted from this proof.

RESEARCH AREAS

Natural Language Processing

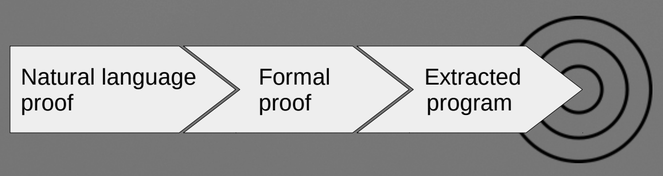
Mathematicians and logicians tend to use more natural style when writing proofs. Existing proof assistants are more restrictive in terms of language they use, thus being not very appealing for inexperienced users. This research looks into natural language processing in order to offer a simple and more friendly way of writing easily readable proofs, which can then be processed by a proof assistant.

Formal Systems

The *λ-calculus* is like the smallest universal programming language of the world. In its most simple form *λ-calculus* has three main sorts of terms, namely variables (e.g., *x*), functions (*λx.t*) and function application *(t t)*. We use a modified version of *Church's simple theory of types*, which is a formulation of higher-order logic based on *simply typed lambda calculus*, a typed version of *λ-calculus*. This formal system has additional terms (constants, pairs and projections) and is used for *realizability interpretation.* The latter gives a theoretical basis for program extraction.

Program Extraction

This research is looking at program extraction as a way of generating executable programs from intuitionistic proofs.



SCRIPT SYSTEM

Script is a Haskell-based proof search tool that is currently being developed as a prototype system to later be used for program extraction. It supports Natural Deduction Rules in first order logic.

An example of a proof in *Script* system:

```
Theorem th1
Ex x A(f(x)) -> All x (A(x) -> B(g(x))) -> Ex x B(x)
Proof
Assume 1 Ex x A(f(x)).
Assume 2 All x (A(x) -> B(g(x))).
Let x be arbitrary.
Assume 3 A(f(x)).
Hence 4 B(g(f(x))) by 2 and 3.
Hence 5 Ex x B(x) by 4.
Hence 6 All x (A(x) -> B(g(x))) -> Ex x B(x) by 5.
Hence 7 Ex x A(f(x)) -> All x (A(x) -> B(g(x))) ->
Ex x B(x) by 6.
Qed
```

USEFUL VOCABULARY

λ-calculus: a formal system developed by Alonso Church in 1930s to formalise the idea of effective computability.

Formal system: is a well-defined system based on a model of mathematics.

Realizability: a collection of methods for studying constructive proofs and extracting information from these proofs.

More information available here

Swansea University
Prifysgol Abertawe