Logical Relations for Program Verification — Pathways to Impact

In an economy as relentlessly digital as the modern worldwide one, in which everything from toasters to interpersonal communications to global financial services are computerised, the need for formally verified software cannot be overestimated. Formal verification uses mathematical techniques to prove that programs actually perform the computations they are intended to perform and/or avoid performing unintended ones (e.g., leaking credit card details or launching nuclear weapons without authorisation). Since programmers make 15 to 50 errors per 1,000 lines of code [1], and since these come at enormous cost [2], the ever-increasing size and sophistication of programs makes formal verification increasingly critical to modern software development.

One major approach to software correctness is *language-based verification*, in which successful compilation of programs provides machine certification of their correctness. Language-based verification thus supports the development of software that is *correct by construction*, which is the logical conclusion of the persistent trend in software engineering toward ever earlier program verification. Logical relations arise naturally in language-based verification since they formalise the use of induction on the structure of types to prove properties of programs; examples of their use are given in the main body of the proposal. Although the proposed research — namely, providing a principled, comprehensive, and predictive foundation for logical relations — does not lend itself to immediate industrial uptake, we have devised the following five-point plan for maximising its impact, so as to get as close as possible to our goal of making logical relations fit-for-purpose for program verification.

• **Publication:** We will, of course, pursue the kinds of publication venues all good scientists pursue. Principally, we will produce high-quality papers and endeavour to publish them in the best journals. Publication of our papers in leading archival journals will confer validation by the community of the correctness and importance our our results, as well as allow them to serve as seminal references. However, the tradition in computer science is also to aim for early publication of important results to keep pace with the rapidly changing nature of the discipline. We will therefore also seek to publish our key results in top conferences. We expect each of our work packages to result in at least one publication at such a conference, and several to result in two.

• **Scientific Interaction:** To increase the impact of our research, we will continue to interact with our research community via conferences, research visits, and other meetings. We will also host specialist workshops such as the one on logical relations Dr Johann recently held at the University of Strathclyde. This workshop is exactly the kind of high-impact event — targeted at a small number of experts and in a very specific area — that truly aids and disseminates research. We have therefore requested funding for a follow-up workshop in the proposal budget. The cost is entirely minimal compared to the opportunities such focussed meetings offer.

We are especially involved with our research community in the UK. For example, Prof Ghani co-founded the Scottish Category Theory Seminar, and both Prof Ghani and Dr Johann are active members of the Scottish Programming Languages Seminar and SICSA, the Scottish computer science pooling body. More generally, Scotland is an excellent place to interact with other researchers on a regular informal basis: there are strong programming languages research groups at Edinburgh, Glasgow, Heriot Watt, and St Andrews, and there are also Scottish Theorem Proving meetings. Further afield, we will interact with researchers from across the UK. We have excellent contacts at all the major relevant research groups, and although distance will make interactions with them less frequent, that in itself will provide fresh perspectives as the grant progresses.

• **Collaboration:** To further maximise impact, we have invited a number of internationally leading project partners to work with us on specific work packages of the proposed research (see below). This is advantageous in several ways. First, working with these project partners will enhance the scientific quality of the proposed research. Secondly, working with these partners will mean that they are intimately involved with the proposed research; in effect, thorough dissemination of our ideas will be occurring with our partners even as the work is being done, and this will help get our ideas out into the broader research community. Finally, drawing on a variety of different perspectives will help ensure that the proposed research does not become overly insular, but instead is outward-looking and solves important problems in key application areas.

Our project partners are Prof Alex Simpson (U Edinburgh, WP1) and Drs Robert Atkey (Contemplate, WP4), Andrew Kennedy (Microsoft, WP5), Nick Benton (Microsoft, WP6), and Carsten Schürmann (ITU, WP7). We have thus secured the involvement of an industrial partner for each of the "application" work packages in Phase 2. In terms of maximising impact, the collaboration with Kennedy is the closest we come to actual industrial crossover. Since Kennedy's original units of measure system has been incorporated into Microsoft's *F#*, the generalisation we propose to develop may do as well. Although it is not the main focus of the proposed research, we will certainly investigate this possibility. Finally, we are keen to see the foundations we develop deployed in real language implementations, so are fortunate to have Dr Conor McBride, architect of Epigram, in our research group. Dr McBride is an expert in the design and implementation of dependently typed languages, and is a valuable source of information about what users expect from real systems.

• **The Stream Model:** One prevalent model of research is the *stream model.* Under this model, to solve a problem one looks both "upstream" for fundamental theoretical ideas to feed into the research, and "downstream" for validation of the research by those who stand to benefit as end-users. This model maximises impact by ensuring that research is picked up not only by those working on the same problem, but also by those upstream, who will be interested in how their own work is applied, and by those downstream, who will look to apply it to their work. To help ensure that the proposed research has a high impact we will follow this stream model.

Upstream of us there are theoretical computer scientists, including type theorists and category theorists, whose results we will apply to develop our new foundations for logical relations. Logical relations themselves are, as already indicated, of great interest to programming languages researchers. Downstream, those in the programming languages and program verification communities will be interested in applying our results.

• **The Work Packages:** Our main consideration in developing our work packages is the quality of the scientific ideas underlying them, since high-impact research obviously requires high-quality ideas. For the specific research proposed here it also requires i) not just a much better understanding of, and facility with, known results about logical relations, but also a predictive framework for them that unifies those results and extends them to new computational settings; ii) not just evidence of the overall power of our new framework, but also a collection of examples outlining a methodology for deploying it to solve key computational problems; and iii) not just theoretical ideas presented in a way that theoreticians can understand, but also concrete program verification ideas that programmers can understand and use on their own terms.

These three requirements are addressed in the three phases of the proposed research. Phase 1 aims to develop a new foundational theory for logical relations, but goes beyond what has been done before by examining the ideas of morphisms between, and constructions on, them. Phase 2 showcases the flexibility of our fibrational framework for applications in terms of the choice of base category, total category, and functor to be lifted from the former to the latter. Phase 3 seeks to move away from the fibrational foundations of our framework by implementing our results in a tool that makes them directly usable by non-specialists. We are particularly excited about Phase 3. It stretches our current abilities, but we see it as a critical component of the proposed research. And while it is definitely the riskiest part of the proposed research, we will minimise that risk by consulting experts in the area, looking to hire such an expert as an RA, and keeping our goals for this phase modest. The latter is reasonable, since achieving even our modest goals will significantly increase the impact of the research proposal on the wider community.

• **Hidden Foundations:** Although the tool produced in WP7 will be based on the our new fibrational framework for logical relations, what users of our tool will see is a sound and effective proof environment whose use requires no knowledge of the underlying category theory whatsoever. We have already applied this "hidden foundations" approach to categorical descriptions of programming language artefacts in, e.g., [3, 4]. There, category theory is used to organise formalisations of programming languages, but the resulting definitions and lemmas can be effectively used with absolutely no category-theoretic knowledge at all.

In WP7 we will develop a logical system that practitioners can use (either by hand or via its implementation) without needing a complete understanding of why the derivations of the system are correct. By removing the categorical over head completely in WP7 we will considerably increase the scope of impact of the proposed research. However, even in the other work packages, we will still use the hidden foundations approach, e.g., by comprehensively treating special fibrations of interest like the families fibration and the subobject fibration. We will also concentrate on showcasing our results via examples as much as possible, since this will allow those with modest or no categorical background to use and profit from our research.

References

[1] Cost of Independent Software Verification & Validation. 2011. At `http://www.galorath.com/wp/cost-of-independent-software-verification-validation-ivv.php`

[2] Total economic cost of insecure software: $180 billion a year in the U.S. 2008. At `http://news.`

cnet.com/8301-13846_3-9978812-62.html

[3] N. Ghani, P. Johann, C. Fumex. Generic Fibrational Induction. *Logical Methods in Computer Science* 8(2), 2012.

[4] P. Johann, N. Ghani. Foundations for Structured Programming with GADTs. POPL'08, pp. 297-308.