Logical Relations for Program Verification — Case for Support

Part 1A: Previous Research & Track Record - Dr Patricia Johann

See also `http://www.cis.strath.ac.uk/~patricia`

Dr Johann earned her PhD in Mathematics in 1991 from Wesleyan University, where she worked on first- and higher-order equational unification. She held tenure-track positions at several prestigious liberal arts colleges in the US, and research positions at the Universität des Saarlandes (supported by the DAAD and NSF) and the Oregon Graduate Institute. She joined Rutgers University as an Assistant Professor in the autumn of 2002, and was granted tenure and promoted to Associate Professor in April 2008. She joined the University of Strathclyde as a Senior Lecturer in July 2008 to co-found the Mathematically Structured Programming group. In May 2010 she was promoted to Reader. Dr Johann's research includes work in the following areas forming the pillars upon which this research proposal is built:

• **Core Foundations for Logical Relations:** For over a decade, Dr Johann has been extending the original treatment of logical relations to more realistic programming languages. For example, in a 2004 POPL paper she extended logical relations to a non-strict polymorphic λ-calculus with a selective strictness primitive (e.g., Haskell's *seq*), and has since done the same for other λ-calculi that approximate "real" functional languages. Her work on *seq* overturned decade-old conventional wisdom, and was cited in the "History of Haskell" retrospective by the language's designers. Dr Johann has also given a uniform treatment of logical relations for algebraic effects (as opposed to an *ad hoc* one handling one effect at a time). This resulted in a landmark paper at the premier conference LICS in 2010. All of these results have been formalised using operational semantics.

• **Applications of Logical Relations:** Since a major aim of this research proposal is to apply new foundations for logical relations to program verification, Dr Johann's experience in this area is highly relevant. She has used logical relations, again in an operational setting, to prove the correctness of program transformations, such as short cut fusion, that are used to optimise programs. In particular, she has used the logical relation for *seq* from the aforementioned POPL paper and related papers to give weakest preconditions for the correctness of short cut fusion and related optimising transformations in the presence of *seq* and other advanced language features.

• **Category Theory:** Dr Johann has considerable experience in the application of category theory to programming languages. For example, she used Kan extensions to develop an initial algebra semantics for nested types and GADTs, which led to the surprising observation that the standard `fold` combinators for these types are as expressive as their corresponding generalised `folds`. More recently, she has used the same techniques that underlie this proposal — namely, lifting functors on the base category of a fibration to functors on its total category — to give a clear and comprehensive axiomatic treatment of induction and coinduction. This categorical work was undertaken with Prof Ghani, and shows their capacity to collaborate successfully.

Publications: Dr Johann has written one book and 35 internationally refereed papers, including 18 journal papers. These can be found from the above url. Apart from the research mentioned above, Dr Johann has also worked on termination, strategy-based programming, and data type refinement. These topics are all concerned with programming languages research, and so are related to the proposed research.

Grants: The most significant of Dr Johann's grants are: i) Categorical Foundations of Indexed Programming, EPSRC 2010-2012, Principal Investigator, £281,879; ii) Applications of Fibrational Induction, SICSA 2012-2015, PhD Studentship, approx. £100,000; iii) Initial Algebra Packages for GADTs: Principled Tools for Structured Programming, NSF 2008-2010, Sole Investigator, $138,000; iv) Provable Safety for Performance-Improving Free Theorems-Based Program Transformations, NSF 2004-2008, Sole Investigator, $123,780 (the proposal for this grant earned a top rating from the assessment panel); v) Testing and Enhancing a Prototype Fusion Engine, NSF 1999-2002, Sole Investigator, $50,360.

Esteem Indicators: Dr Johann is an associate editor of *Higher-Order and Symbolic Computation*. She has also edited four special issues of journals. She was an invited speaker at MFPS'12, ARW'11, MSFP'09, and NWPT'05, and has served on the programme committees of LICS'13, FoSSaCS'13, CMCS'12, TFP'12, POPL'12 ERC, ICFP'10, RTA'10, FLoPS'10, PEPM'10, IFL'09, ML'09, POPL'09, IFL'09, PEPM'08, PADL'08, PPDP'07, HW'07, TFP'07, ICFP'06, and GPCE'05. She was Workshop Chair for ICFP'05 and Workshop co-Chair for ICFP'06. She has served on NSF panels charged with evaluating the quality of, and determining funding for, grant proposals. She has successfully supervised one PhD student and currently is supervising another. She also supervises an RA. She has been the external examiner for one PhD student.

Part 1B: Previous Research & Track Record - Prof Neil Ghani

See also `http://www.cis.strath.ac.uk/~ng`

Prof Ghani earned his PhD in Computer Science in 1995 from the University of Edinburgh, where he worked on categorical models of rewriting. He was subsequently awarded grants from the Royal Society of London and the EU-funded EUROFOCS program to conduct postdoctoral research at the École Normale Supérieure in Paris. In January 1997 he became a Research Fellow at the University of Birmingham, and in September 1998 he became a Lecturer at the University of Leicester. In September 2005 he moved to the University of Nottingham, where he was promoted to Reader in January 2007. In July 2008, he joined the University of Strathclyde as a Professor to co-found and lead the Mathematically Structured Programming (MSP) research group. While Dr Johann's track record focusses directly on logical relations, Prof Ghani's track record focusses directly on the mathematical foundations of the project. These are:

- **Category Theory:** Category theory has been a key component of Prof Ghani's work throughout his research career, and he has significant expertise in the specific categorical foundations that underlie the proposed research. Indeed, he has applied various categorical structures such as initial algebras, monads, comonads, final coalgebras, enriched categories, Kan extensions, and fibrations to problems in computation. Of closest relevance to this proposal is his work (jointly with Dr Johann) on fibrational models of induction and coinduction, the categorical semantics of effects, and the use of slice categories to model dependent types.
- **λ-Calculi:** In addition to his categorical research, Prof Ghani has also studied various λ-calculi — including $\lambda 2, \lambda\omega$, and the Calculus of Constructions — from a purely syntactic perspective. For each of these calculi, Prof Ghani developed the subject of η-expansions and showed it to be better behaved than the more traditional theory of η-contractions. He also solved the long-standing open problem of the decidability of $\beta\eta$-equality for sum types, which had attracted the attention of research groups across the world. Prof Ghani's familiarity with the key systems of the λ-cube will feed directly into the development of logical relations for it.

Publications: Prof Ghani has written 58 papers internationally refereed papers, including 17 journal papers. These can be found from the above url. Apart from the research mentioned above, Prof Ghani has worked on containers and induction recursion, and on data structures for representing continuous functions.

Grants: Prof Ghani's most significant grants are: i) Theory and Applications of Induction Recursion, EPSRC 2009-2012, Principal Investigator, £310,904; ii) Categorical Foundations of Indexed Programming, EPSRC 2010-2012, Co-Investigator, £281,879; iii) Reusability and Dependent Types, EPSRC 2009-2013, Principal Investigator, £148,280; iv) Theory and Application of Containers, EPSRC 2005-2008, Principal Investigator, £228,561; v) Kan – A Categorical Approach to Computer Algebra, EPSRC 2001-2004, Sole Investigator, £126,322; vi) Categorical Rewriting: Monads and Modularity, EPSRC 2000-2002, Sole Investigator, £52,560.

Esteem Indicators: Prof Ghani has served as a member of the EPSRC College charged with assessing the quality of grant applications. He was the director of the Midlands Graduate School in the Foundations of Computer Science, and was on the Steering Committee for the British Colloquium on Theoretical Computer Science. He has served on the programme committees of MFPS'11, CMCS'10, RTA'09, CMCS'08, RTA'08, CMCS'06, and RTA'02, is the leader of the MSP group, and is also Deputy Head of Department. He has successfully supervised five PhD students and two RAs, and is currently supervising two more PhD students. He has been the external examiner for two PhD students.

The Host Institution

The MSP group at the University of Strathclyde is an ideal venue for conducting this research. Dr Johann and Prof Ghani have a well-established, productive collaboration centred on using mathematical structures to guide programming languages research, and are internationally known for their work in λ-calculi, category theory, functional programming, and logical relations. Moreover, Dr Conor McBride's membership in the MSP group ensures local access to a programming perspective that complements their more foundational one, and Epigram, a state-of-the-art dependently-typed programming system of the sort intended to incorporate the results of the proposed research. The MSP group also has eight PhD students and two RAs. The strength of the group has been recognised by the University of Strathclyde, which is just now hiring a new Lecturer for it. Finally, central Scotland is home to vibrant theoretical computer science and functional programming research communities, and the University of Strathclyde is an active participant in the Scottish Informatics and Computer Science Alliance (SICSA). The MSP group also is active in other Scottish meetings, such as ScotCats and SPLS.

Part 2: The Proposed Research and Its Context

1 Introduction

Formal reasoning is critical for building truly secure and reliable software, and one of the key techniques for establishing properties of software systems is that of *logical relations* [18]. Logical relations can be used to prove i) properties of programs, e.g., that they perform no illegal operations, satisfy certain security criteria, or are observationally equivalent to their compiled forms; ii) whole-language properties, e.g., that languages satisfy parametricity, enforce information flow policies, or guarantee privacy; and iii) properties of implementations, e.g., compiler correctness. Logical relations have been well-studied for the polymorphic λ-calculus $\lambda 2$, which forms the core of many modern programming languages and verification systems. But as languages, type systems, and properties have become increasingly sophisticated and expressive, logical relations have struggled to keep pace, and this has led to a fragmented understanding of how to develop and use them. Indeed, although there is a large body of work on logical relations — comprising a core theory for $\lambda 2$ and case-by-case extensions for an enormous array of language features — and despite good efforts in this direction (e.g., [4, 6, 7]), there is currently no comprehensive theory that both accounts for existing logical relations and also has sufficient predictive power to guide their extension to features — e.g., security types, dependent types, quantum types — in development for programming languages of the future.

We believe that there are fundamental limitations to our current understanding of logical relations, and that these have led to a severe disconnect between the need for a foundational theory of logical relations on the one hand, and the *ad hoc* approach to them actually taken in practice on the other. This disconnect not only impedes the development of logical relations for today's languages, but threatens to do the same for tomorrow's as well.

2 Overview of the Proposed Research

The goals of the proposed research are thus: i) to provide a proper foundation for logical relations in the form of a new axiomatic framework that is conceptually simple, broadly applicable, uniform (rather than* ad hoc*), predictive, and implementable; ii) to bridge the disconnect between the need for a foundational theory of logical relations and the current* ad hoc *approach to their construction and effective application by showing that our framework can be instantiated to solve state-of-the-art problems in programming languages and program verification; and iii) to ensure uptake of our framework by providing a logic and tool support for it. The proposed research will be undertaken in the following distinct-but-interrelated phases.

• **The Core Phase:** Our new framework for logical relations will be based on the category-theoretic concept of *comprehension for a fibration* [11]. Fibrations are a fundamental tool for describing the relationship between a programming language and a logic for reasoning about it. In fibrations that support it, comprehension allows explicit representation of logical properties of a language within the language itself. In [8, 9], we used comprehension to give a uniform and axiomatic treatment of induction and coinduction rules for proving logical properties of data types that both accounts for known induction and coinduction rules, and develops previously unknown ones. Since properties in certain fibrations can be thought of as relations on types, and since comprehension can remove conditions that prevent current fibrational approaches to logical relations from covering important models, we expect comprehension to support a similarly uniform and axiomatic treatment of logical relations. To our knowledge, comprehension has not previously been identified as a key ingredient in the construction of logical relations, so our use of it distinguishes our proposed framework from all other treatments of logical relations in the literature. Our framework will be capable of both *de*scribing the wide array of logical relations already used in existing applications and *pre*scribing new logical relations for future ones.

• **The Exploitation Phase:** We will apply our uniform and axiomatic framework for logical relations to cutting-edge problems that are the focus of active research, and for which there is presently no consensus on the way forward. Successful application of our framework will show that it can solve problems that are the focus of considerable research effort (e.g., parametricity for dependently typed languages and languages with effects), and also open up unanticipated new research directions (e.g., generalised units of measure). Conversely, the practical applications we pursue will raise challenges that prompt us to further refine our framework.

• **The Impact Phase:** Developing a new and fundamentally better understanding of logical relations is not just an end in itself, but is also a key prerequisite for effectively applying them. To help ensure uptake of our research by the wider community, we will produce a logic for deriving consequences of logical relations, and a prototype implementation of it as tactics for the interactive prover Coq. This will allow users to experiment with our framework; at the same time, their practical experiences with it will feed back into its foundations.

3 Methodology and Research Programme

A fundamental theory of logical relations should not deal with specific programming languages, but rather abstract the key features of languages to concepts that can later be instantiated for a wide range of applications. Our proposed framework abstracts the notion of a model for a language (set-theoretic, domain-theoretic, realisability, etc.) by giving each type (term) a *basic interpretation* as a functor on a category $\mathscr{B}$ (resp., a natural transformation between such functors). To study logical properties of languages, specific notions of property (e.g., boolean-valued, proof-relevant, admissible) are abstracted via a category $\mathscr{E}$. A functor $U : \mathscr{E} \to \mathscr{B}$ mapping each property to the type whose programs it is a property of is also given. Crucially, every substitution on types must lift to a substitution on properties; this is captured categorically by requiring U to be a *fibration*.

To study logical relations, a new fibration $Rel(U) : Rel(\mathscr{E}) \to \mathscr{B} \times \mathscr{B}$ is first obtained by pulling U back along the product functor on $\mathscr{B}$. Since $Rel(\mathscr{E})$ abstracts the specific notion of relation (e.g., arbitrary, $\top\top$-closed, proof-relevant), we call $Rel(U)$ the *relations fibration* on $\mathscr{B}$. A fundamental feature of logical relations is that every type T has not only a basic interpretation $[\![T]\!]$ with respect to $\mathscr{B}$, but also a *relational interpretation* $[\![T]\!]^{\mathscr{R}}$ with respect to $Rel(\mathscr{E})$, and similarly for terms. Our important innovation is to use comprehension to define relational interpretations simultaneously with basic ones, and then to view these interpretations of types (terms) as *fibred functors* with respect to $Rel(U)$ (resp., *fibred natural transformations* between such functors). This extends the usual categorical interpretations of types (terms) as functors (resp., natural transformations) to fibrations, and allows all of fibred category theory to be applied to logical relations. For example, interpreting a type T as a fibred functor entails that $[\![T]\!]^{\mathscr{R}}$ is a *lifting* of $[\![T]\!]$ — i.e., if $R : A \leftrightarrow B$ then $[\![T]\!]^{\mathscr{R}} R : [\![T]\!]A \leftrightarrow [\![T]\!]B$ — and that the relational actions of type constructors in standard approaches are liftings in the fibrational one.

The essence of a logical relations argument is i) a proof of the *Identity Extension Lemma (IEL)*, i.e., a proof that the relational interpretation of each type maps equality relations to equality relations; ii) a proof of the *Fundamental Theorem of Logical Relations (FTLR)*, i.e., a proof that each term is related to itself by the relational interpretation of its type; and iii) an instantiation of the FTLR that derives the properties of interest. Since our framework abstracts equality to a functor $Eq : \mathscr{B} \to Rel(\mathscr{E})$ mapping each object of $\mathscr{B}$ to the equality relation on it, the IEL is simply the requirement that the lifting $[\![-]\!]^{\mathscr{R}}$ of $[\![-]\!]$ preserves Eq, i.e., that $[\![T]\!]^{\mathscr{R}} \circ Eq = Eq \circ [\![T]\!]$. Perhaps surprisingly, the FTLR has an even simpler fibrational formulation as the requirement that every term is interpreted as a fibred natural transformation. Fibrational treatments of logical relations tend to be very involved, and so are often usable only by those with specialist categorical knowledge. But the simplicity of the above formulations will ensure that our framework is maximally accessible and minimise the categorical overhead required to use it. This will enable us to give a sound and widely applicable framework that supports effective reasoning with logical relations, and whose use requires no knowledge whatsoever of the underlying category theory. We have already successfully applied this "hidden foundations" approach extensively in our previous work, where category theory is used to organise formalisations of programming languages.

3.1 A Core Fibrational Theory of Logical Relations

• **WP1: A Basic Fibrational Theory of Logical Relations** Any credible theory of logical relations must specialise to the standard one for $\lambda 2$, and so must define relational interpretations of $\to$-types and $\forall$-types.

$\to$-types: Rather than defining relational interpretations just for $\to$-types, we will instead solve the more general problem of defining uniform liftings for arbitrary mixed variant functors. The lifting in [8, 9] for covariant functors is defined using comprehension and reindexing; comprehension's crucial role is to turn properties into types so that the relational interpretation of a type can be defined using its standard interpretation. In preliminary work, we have extended this lifting to mixed variant functors by using comprehension and opreindexing to treat their contravariant actions. We expect the resulting lifting to be equality-preserving for mixed variant functors because it is equality-preserving for covariant ones, because equality is a colimit, and because opreindexing is a left adjoint and thus preserves colimits. We will validate this lifting by i) showing that it agrees with the lifting obtained by assuming that the base and total categories of the underlying Lawvere fibration are cartesian closed and that the fibration preserves this structure; and ii) using it to extend the induction rules from [8, 9] from covariant to mixed variant data types. We will further show that our extended rules not only specialise to Pitts' for locally continuous functors over $CPO_{\perp}$ [15] and Hermida and Jacobs' for FPC [10], but also generalise Pitts' to arbitrary predicates and fibrations, and Hermida and Jacobs' to arbitrary mixed variant functors.

$\forall$-types: The treatment of $\forall$-types is well-known to require delicacy. For example, although interpreting $\forall$-types as ends in the base category of a Lawvere fibration appears promising at first glance, this idea is flawed because

it entails interpreting terms as dinatural transformations (which, unlike terms, do not compose), and also allows "non-parametric" functions in the model [4]. To address this, we will instead take ends in the total category of relations constructed from the Lawvere fibration, rather than in its base, and then use comprehension to map these ends back down to the base. This will give an axiomatic definition of relational interpretations for $\forall$-types in any Lawvere fibration. We will validate our definition by instantiating it to obtain the usual interpretation of $\forall$-types, as found, e.g., in [18]. Since calculi that, like $\lambda 2$, support binding of type variables are fibred over type contexts, the main technical challenge will be to adapt the above ideas to relations fibrations that are themselves fibred over fibrations of types over type contexts. Fibrations between fibrations have already been used in the context of logical relations, but in fact we will use a new and more refined idea, namely that of Lawvere fibrations between $\lambda 2$-fibrations (which are currently the canonical fibrational models of $\lambda 2$).

We will verify that the relational interpretations we define can be used to derive standard results, such as the existence of initial and final coalgebras of functors. The result will be a new axiomatic theory of logical relations for $\lambda 2$ that encompasses and unifies the diversity of currently existing ones.

• **WP2: Relating and Deriving Logical Relations** WP1 focuses on one logical relation at a time. But we may want to relate reasoning with respect to one logical relation with reasoning with respect to another; e.g., one logical relation may interpret types as sets while the other may use domains, or one may be proof irrelevant (i.e., use sets), while the other may be proof relevant (i.e., use setoids). Or we may want to compile one language into another and know what properties of the source code hold for the compiled code, or to derive results about call-by-name and call-by-value variants of a language from one another. These examples show that transporting proofs with respect to one logical relation to proofs with respect to another is a common and important problem.

The categorical emphasis on *both* objects *and* relationships between them suggests it will be feasible to extend our uniform and axiomatic framework to relating logical relations. The key technical challenge is to give an appropriate definition of a morphism between Lawvere fibrations. This requires determining which structural components of Lawvere fibrations morphisms should preserve, and whether these should be preserved up to natural isomorphism, natural transformation, or even the interpretation of observational equivalence induced by the fibrational rendering of the FTLR. The ability to specify preservation of structural components of Lawvere fibrations up to observational equivalence, and thus to incorporate the experience of practitioners directly into our uniform framework, is a testament to the robustness and flexibility of our categorical methodology.

• **WP3: Canonical Constructions on Logical Relations** One of the greatest strengths of category theory is that it supports canonical constructions of new categories from old ones; examples are slice categories, functor categories, and Kleisli categories. It is thus natural to ask whether our axiomatic framework similarly supports canonical constructions of new logical relations from old ones. Although such constructions would be of tremendous value, as far as we know this question has not even been asked. We believe this is because current approaches are too cumbersome to make investigations into canonical constructions of logical relations feasible.

We aim to show that our axiomatic framework is stable under three categorical constructions: i) slices, which are the canonical construction for handling indexing; ii) functor categories, which are the canonical construction for modeling Kripke logical relations; and iii) Kleisli categories, which are the canonical construction for incorporating effects. The morphisms we define in WP2 will be crucial for ensuring that our constructions are correct, i.e., satisfy the required universal properties. We will also need to show that our constructions preserve the fibrational structure needed to define logical relations. We have experience here: we have shown in [8], for example, that the pullback of a Lawvere fibration along a fibration is again a Lawvere fibration.

3.2 Exploiting the Core Fibrational Theory

Once we have developed our new fibrational framework we will show how it can be used to derive logical relations for calculi with state-of-the-art language features. These cutting-edge applications have been chosen specifically to showcase the flexibility inherent in the fibrational approach. This flexibility derives from the support fibrations provide for choosing different functors, base categories, and total categories.

• **WP4: Varying Functors: Logical Relations for the λ-Cube** The λ-cube [2] gives a uniform presentation of several calculi — including $\lambda 2$, the dependently typed λ-calculus (λP), and the calculus of constructions — that lie at the core of modern programming languages. By varying the class of functors, our framework can derive logical relations for these calculi in a uniform and axiomatic way; our methodology is thus comprehensive since it is relevant to all languages based on calculi of the cube. Logical relations for the calculi of the cube are studied in, e.g., [3], but a definitive treatment remains elusive. Our framework will either confirm the definitiveness of

existing logical relations, or the logical relations it yields will become candidates for the definitive ones.

We will first derive logical relations for the dependently typed λ-calculus λP. For this, we will instantiate our axiomatic framework to functors that are Lawvere fibrations between locally cartesian closed categories. Since such categories are the canonical models for λP, this is the analogue for dependent types of deriving logical relations for $\lambda 2$ from Lawvere fibrations between $\lambda 2$-fibrations, as in WP1. We expect to similarly derive logical relations for the other calculi of the cube using Lawvere fibrations between the fibrations that model them, resulting in a *fibred λ-cube*. It time permits, we will extend this methodology to pure type systems [2, 11].

• **WP5: Varying the Base Category: Logical Relations for Algebraically Indexed Types** Andrew Kennedy of Microsoft Research recently incorporated dimensional analysis into type-checking, thus making it possible to detect dimensional errors — such as the units mismatch that caused the Mars orbiter's famous $125 million "metric mishap"— at compile time. This feature, dubbed *units of measure*, has been integrated into Microsoft's *F#*. Its technical justification involves constructing a suitable logical relation for the unit-indexed language.

Kennedy's key observation is that units of measure form a free abelian group (i.e., can be multiplied and divided to get, e.g., square metres or metres per second). But in fact the algebraic structure on units can be generalised to arbitrary equational theories; [1] develops such *generalised units of measure*. Significantly, each of [1] and [13] define exactly one logical relation for the unit-indexed language it considers; this hardwires in the notion of "relatedness" of programs, and so is too restrictive. To remedy this, we propose to develop an axiomatic approach to logical relations for units of measure. We will first use our axiomatic framework from WP1 to construct logical relations for Kennedy's original language; preliminary investigations show that, in addition to the usual structure required of fibrations, its base category will need an abelian group object. We will then use our framework to construct logical relations for the language of [1]. This will require that the base category has a representing object for the particular algebraic structure given by the equational theory. The flexibility of fibrations is crucial because different equational theories on units require different base categories.

• **WP6: Varying the Total Category: Logical Relations for Effects** The approach to effects pioneered by Moggi based on monads, has been a remarkable success because of its simplicity and uniformity, but it also has a serious drawback: the effect-triggering operations are absent! As a result, Plotkin and Power [17] have identified a class of effects, called *algebraic effects*, that are generated by algebraic operations and have monadic structure as a derived artifact. Algebraic effects are widely used and have good logical properties; for example, unlike monads, they compose nicely. Although logical relations for languages with algebraic effects have been studied, e.g., in [14], there is currently no comprehensive treatment of them that abstracts specific models and relations. We propose to develop a uniform and axiomatic treatment of logical relations for algebraic effects. The flexibility of fibrations will be key because different algebraic effects require different kinds of relations, and hence different total categories [5]. Since more than one effect is often present in a computational setting, we also propose to use our axiomatic framework to solve the problem, left open in [12], of constructing logical relations for languages supporting combinations of algebraic effects. Given the inherently algebraic nature of the canonical operations for combining algebraic effects (e.g., sum and tensor), fibrations provide a natural framework for studying this problem. We thus propose to lift effect-combining operations to logical relations by defining corresponding operations on fibrations. This will extend the constructions from WP3. If time permits, we will use our axiomatic framework to construct logical relations for non-algebraic effects as well.

3.3 Generating Impact from the Core Fibrational Theory

• **WP7: A Logic for Logical Relations** The ultimate goal of the proposed research is to support program verification via logical relations. Our final work package comes full circle to our original motivation: we will use our axiomatic framework to provide a tool to aid the construction of logical relations that make this possible.

We will do this in two stages. In the first stage, we will codify our axiomatic framework in a logical system of inference rules for reasoning with logical relations. The construction of logical systems to guide formal reasoning is widely accepted in computer science and has already been applied to logical relations (see, e.g., [16]). Our logical system will be based on the basic structure of a Lawvere fibration and its refinements from the previous six work packages. In the second stage, we will implement this logical system in Coq. By contrast with the previous work packages, we have deliberately set cautious goals for this one. Implementers have strongly advised us not to underestimate the engineering difficulties of constructing a comprehensive tool, which could easily constitute a whole project on its own. So we will provide a collection of prototype Coq tactics that establishes proof-of-concept, and thus paves the way for integrating the results of the proposed

research into mainstream theorem provers.

4 Quality, Management, and Planning

Calibre, Ambition, and Adventure: The fundamental importance of logical relations, their diverse and ubiquitous application, and the quality of our ideas for constructing them in an axiomatic, uniform, and widely applicable way all attest to the calibre of the proposed research. Our ambition is demonstrated by our desire to transform the subject of logical relations from one developed in an *ad hoc* way to one that has a principled, comprehensive, and predictive foundation. The proposed research certainly is not incremental! The adventurous nature of the proposed research is demonstrated by its scope, which ranges from fundamental research (WP1-WP3) to applications (WP4-WP6) to tool development (WP7). The proposed research is also...
...timely: This is an extremely timely moment to embark on the proposed research. Not only do we have our own results to draw on, but there have also been significant recent advances in directly related areas, such as step-indexed logical relations and logical relations for pure type systems. Moreover, programming languages have advanced to the stage where, for the first time, the results of fundamental research on logical relations are starting to be realised in code, and so can be used to structure both proofs and programs.
...novel: Perhaps the most novel aspect of the proposed research, especially when compared against all other research on logical relations, is our idea of defining logical relations via uniform liftings of functors rather than by induction on type structure. In addition, our use of comprehension to accomplish this is a distinctive feature of our methodology, Finally, the infrastructure our framework provides for considering morphisms between, and standard categorical constructions on, logical relations is a notable benefit of our categorical perspective.

National Importance: The software market is estimated at $500 billion per year, and this figure is likely to grow significantly in real terms as software becomes ever more ubiquitous. It is thus essential to the UK's national interest to have a strong presence in this market. One crucial aspect of software is that it is correct, i.e., does what's intended and does not go wrong. Even failures of everyday devices like iPods and mobile phones are inconvenient, but software leaking voting records, compromising the global financial sector, or launching nuclear weapons without authorisation can lead to unprecedented and clearly unacceptable global uncertainties.

While testing of programs has dominated the last 50 years of software development, the next 50 years are likely to see an increasing demand for provably correct software. This is partly because testing is by its very nature only a partial guarantee, and partly because programming language technology is finally advancing to the stage where it is feasible to formally verify critical programs. Both programming languages and program verification are identified in EPSRC's portfolio as areas of vital importance for cybersecurity, and EPSRC thus intends to grow them. This proposal uses ideas from programming languages to enhance program verification, so lies squarely in their intersection. The UK is a world leader in these areas, but continued investment is required to maintain that status in a rapidly changing world and a rapidly evolving field.

Within programming languages and program verification, we are aiming high. The current *ad hoc* approaches to logical relations are all based on Reynolds' seminal idea from 30 years ago, and we are hopeful that our new foundation will acquire the same status and have similar longevity. The proposed research has the potential to become a cornerstone for reasoning about programs, and to be cited by both theoreticians and practitioners well into the future. If successful, the proposed research can be expected to have great impact on programming languages and program verification over the next 10 to 50 years, and perhaps even beyond.

Feasibility: We are *uniquely well-positioned* to conduct the proposed research. Dr Johann is an expert on logical relations-based reasoning (operational, denotational, categorical), and has applied this to languages with effects and novel languages supporting algebraically indexed types; Prof Ghani is an expert in the categorical semantics of programming languages; and they have already successfully used the core idea of this project — namely, using comprehension to lift functors from the base category to the total category of a fibration — when giving their uniform and axiomatic foundation for induction and coinduction for arbitrary data types.

Success criteria: The success of the core phase of the proposed research will be demonstrated by developing a framework for logical relations for $\lambda 2$ that is uniform over functors, interpretations of types, and interpretations of relations (WP1), as well as new techniques for reasoning about logical relations in terms of one another (WP2) and for combining them (WP3). The success of the exploitation phase will be demonstrated by generalising the results of WP1 to all of the languages of the λ-cube (WP4), by developing a theory of equational parametricity (WP5), and by marrying our fibrational models and effects (WP6). Finally, the success of the impact phase will be demonstrated by developing a proof-of-concept tool showing that results from WP1

through WP6 can be used to extend existing systems to more sophisticated properties and programs (WP7).

Management and Planning: Work on WP1 precedes work on the other work packages because it develops the fundamental techniques to be extended, applied, and implemented. Work on WP2 precedes work on WP3 since morphisms between Lawvere fibrations are needed to establish universal properties of constructions on logical relations. WP4, WP5, and WP6 are independent, but WP1, WP2, and WP3 all feed into them. WP7 will be integrated with the others to the greatest extent possible by starting work on it as soon as results from WP1 are available. Risk *between* work packages is minimised since WP4, WP5, WP6, and WP7 can influence one another, but lack of progress on one will not inhibit progress on others. WP1, WP2, and WP3 are relatively risk free: they build on our previous work, and we already have ideas how to proceed and fallback positions should our first approaches fail. Risk *within* work packages is minimised by applying the right expertise to each. WP1, WP2, and WP3 will be led by Dr Johann and Prof Ghani; Prof Ghani will lead WP4; and Dr Johann will lead WP5 and WP6. The RA will focus on WP7 but may contribute to other work packages as well if they are able.

The RA and Their Training: We will seek an RA with expertise in programming languages, program verification, λ-calculi, and fibrational models of programming languages, or some combination of these, as well as significant experience with a formal proof system such as Coq or Agda. This is feasible: we know of several highly-qualified researchers — e.g., Fredrik Forsberg, Clément Fumex, Barbara Petit, and Noam Zeilberger — due to finish PhDs or postdoctoral positions within the next year, and we will advertise to recruit the best RA possible. MSP group meetings provide opportunities to report on research progress and generate new ideas, and will help integrate the RA into the project. Our reading group for discussing research papers related to the project will also help train the RA. The RA will have the opportunity to write papers and grant proposals, lead research, and help mentor PhD students. At the end of the project the RA will possess highly-desirable knowledge and skills, and be well-positioned to lead future research and/or development efforts. This is important since there is more work to be done in the research and development of logical relations than the active workforce can handle. Overall, the proposed project will have a high impact in terms of training.

Collaboration: In carrying out the proposed research we will collaborate with internationally leading researchers so as to maximise its potential for impact. See our *Pathways to Impact* statement for details.

References

[1] R. Atkey, P. Johann, A. Kennedy. Abstraction and Invariance for Algebraically-Indexed Types. Submitted, 2012.

[2] H. P. Barendregt. Introduction to Generalized Type Systems. *Journal of Functional Programming* 1(2), 1991.

[3] J.-P. Bernardy. A Theory of Parametric Polymorphism and an Application. PhD thesis, Chalmers University of Technology, 2011.

[4] E. S. Bainbridge, P. J. Freyd, A. Scedrov, P. J. Scott. Functorial Polymorphism. *Theoretical Computer Science* 70(1), pp. 35-64, 1990.

[5] N. Benton, A. Kennedy, M. Hofmann, L. Beringer. Reading, Writing and Relations. APLAS'96, pp. 114-130.

[6] L. Birkedal, R. Møgelberg. Categorical Models for Abadi and Plotkin's Logic for Parametricity. *Mathematical Structures in Computer Science* 15(4), pp. 709-772, 2005.

[7] B. Dunphy, U. S. Reddy. Parametric Limits. LICS'04, pp. 242-251.

[8] C. Fumex, N. Ghani, P. Johann. Indexed Induction and Coinduction, Fibrationally. CALCO'11, pp. 176-191.

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

[10] C. Hermida, B. Jacobs. Structural Induction and Coinduction in a Fibrational Setting. *Information and Computation* 145 (2), pp. 107-152, 1998.

[11] B. Jacobs. *Categorical Logic and Type Theory*. Elsevier, 1999.

[12] P. Johann, A. Simpson, J. Voigtländer. A Generic Operational Metatheory for Algebraic Effects. LICS'10, pp. 209-218.

[13] A. Kennedy. Relational Parametricity and Units of Measure. POPL'97, pp. 442-455.

[14] R. Møgelberg, A. Simpson. Relational Parametricity for Comptuational Effects. LICS'07, pp. 436-355.

[15] A. Pitts. Relational Properties of Domains. *Information and Computation* 127, pp. 66-90, 1996.

[16] G. Plotkin, M. Abadi. A Logic for Parametric Polymorphism. TLCA'93, pp. 361-375.

[17] G. D. Plotkin, J. Power. Notions of Computation Determine Monads. FOSSACS'02, pp. 342-356.

[18] J. C. Reynolds. Types, Abstraction and Parametric Polymorphism. *Information Processing* 83(1) (1983), pp. 513-523.