Institution: University of Strathclyde		
Unit of Assessment: B11 Computer Science and Informatics		
Title of case study: Economic impacts and improvements to professional programming practice arising from the addition of dependent types to the Glasgow Haskell Compiler		
Period when the underpinning research was undertaken: 2008 – 2020		
Details of staff conducting the underpinning research from the submitting unit:		
Name(s):	**Role(s) (e.g. job title):**	**Period(s) employed by submitting HEI:**
Conor McBride	Reader	01/07/2008 – present
Sam Lindley	Research Fellow	01/07/2012 – 30/06/2013
Period when the claimed impact occurred: 1st August 2013 – December 2020		
Is this case study continued from a case study submitted in 2014? No		

1. Summary of the impact

As a result of original research by Dr Conor McBride at the University of Strathclyde, dependent types are now embedded in the Glasgow Haskell Compiler, the de facto standard compiler for Haskell, a widely deployed programming language used in industry. As of 2020, 17% of open source Haskell libraries use features directly underpinned by McBride's research. Dependent types in Haskell have been used to support millions of pounds worth of software development at Google, Habito, Galois, Digital Asset and Well-Typed, where it has been credited with improving programme robustness, speeding up development and deployment, and reducing costs.

2. Underpinning research

Context

Dependent types are a programming language feature that capture relative notions of data validity, enabling accurate and enforced description of properties of software, thereby improving software correctness and lowering development costs arising from errors. This is important, as relative notions of data validity are widely employed in programming: efficient searching relies on data structures which satisfy ordering and balancing invariants; matrix multiplication requires matrices of compatible dimensions, and database queries must fit with the structure of the database tables, to cite just three examples. Haskell is a functional programming language with a strong type system, allowing high assurance code to be compiled efficiently. It is the most widely used general purpose pure functional language in professional programming, as well as an active area of research in computer science. The de facto standard compiler for Haskell is the Glasgow Haskell Compiler (GHC).

The Strathclyde Haskell Enhancement

Building on his previous research into dependent types, McBride developed a pre-processor in 2009, the Strathclyde Haskell Enhancement (SHE), in order to demonstrate that dependently typed programming was possible and useful in the context of an industrially deployed programming language like Haskell [R1]. SHE extended the power of GHC with the notational support needed for simulating basic dependent types. These simulations showed the feasibility of adding dependent types to Haskell, and highlighted the changes required in GHC for it to be able to support true dependent types (as opposed to simulated dependent types). Written in collaboration with well-known Haskell programmers from Utrecht University and the University of Nottingham, McBride demonstrated implementation of dependent types in Haskell in R2, using lambda calculus as an example. This output presented the type rules for a dependently typed core calculus, highlighted the changes necessary to shift from a simply typed lambda calculus to the dependently typed lambda calculus, and described how to extend a core language with data types [R2].

Haskell Types with numeric constraints

Partly as a consequence of developing SHE, the US-based technology company Microsoft funded a PhD studentship at Strathclyde for McBride to supervise Adam Gundry on the project Haskell Types with Numeric Constraints (2009-2013). Directed by McBride, this project aimed to extend Haskell with a basic form of dependent types allowing numerical invariants such as matrix dimensions and buffer sizes. The original methodology was to incrementally extend Haskell's existing type inference mechanism with only numeric constraints. However, the project soon grew in scope to encompass more general type inference and unification algorithms for Haskell extended with type-level data and functions [R3]. This project laid the foundations for GHC's implementation of dependent types. A key finding leading to the impact described below is the realisation of the importance of heterogeneous equality, as invented by McBride, for stating equalities in the GHC core language: even when types look different, they can be provably compatible, so that it makes sense to equate a list of length 1+n with a list of length n+1.

Haskell Types with Added Value

The EPSRC-funded project, Haskell Types with Added Value (2012-2013), with Dr Sam Lindley as Research Fellow, aimed to use SHE, along with the emerging dependently typed features of GHC arising from the Haskell Types with Numeric Constraints project, to determine how much of dependently typed programming practice could be ported to an industrially relevant language like Haskell. Until this point, researchers in the Haskell community were still largely working with simulated dependent types, as used by McBride in SHE. However, Lindley and McBride questioned the potential to continue making progress towards dependent types in Haskell using this simulation approach, and argued that simulated dependent types should be replaced with true dependent types [R4]. In particular, this paper highlighted the importance of disentangling the run-time/compile-time and inferred/manifest distinctions implicit in Haskell.

Applications of dependent types

McBride's research has continued to provide further examples of dependently typed programming. At the 2014 International Conference on Functional Programming, McBride argued that intrinsic dependent types, with ordering and balancing invariants internalised, guarantee the correctness of a balanced binary search tree implementation with little programmer effort [R5]. McBride used Agda (a dependent typed functional programming language), but the paper's technique was credited and emulated in a keynote in the same conference by Prof Stephanie Weirich as an advanced example of Dependent Haskell [S1b]. Similarly, in R6 McBride and co-authors showed that dependent types are effective for bug-free manipulation of programming language syntax in a manner now accessible to Haskell programmers.

3. References to the research (Strathclyde affiliated authors in **bold**; FWCI at 02/02/2021)

R1 **McBride, C.** (2009). The Strathclyde Haskell Enhancement. https://personal.cis.strath.ac.uk/conor.mcbride/pub/she/ (cited by peers since 2010)

R2 Löh, A., **McBride, C.**, & Swierstra, W. (2010). A tutorial implementation of a dependently typed lambda calculus. *Fundamenta Informaticae,* 102(2): 177-207. Available from: https://www.andres-loeh.de/LambdaPi/LambdaPi.pdf [FWCI: 3.4]

R3 **Gundry, A.**, **McBride, C.**, & McKinna, J. (2010). Type Inference in Context. In: *MSFP '10 Proceedings of the third ACM SIGPLAN workshop on Mathematically structured functional programming*. ACM, New York, NY, pp. 43-54. https://doi.org/10.1145/1863597.1863608

R4 **Lindley, S.**, & **McBride, C.** (2013). Hasochism: the pleasure and pain of dependently typed Haskell programming. In: *Proceedings of the 2013 ACM SIGPLAN symposium on Haskell.* ACM, New York, NY, pp. 81-92. https://doi.org/10.1145/2503778.2503786

R5 **McBride, C.** (2014). How to keep your neighbours in order. *ACM SIGPLAN Notices*, *49*(9), 297-309. https://doi.org/10.1145/2692915.2628163 [REF2]

R6 Benton, N., Hur C.-K., Kennedy, A., & **McBride C**. (2012). Strongly Typed Term Representations in Coq. *Journal of Automated Reasoning, 49*(2): 141-159 https://doi.org/10.1007/s10817-011-9219-0 [FWCI: 1.29; REF2 in 2014]

Notes on the quality of research: R2-R6 were peer-reviewed ahead of publication. The underpinning research was supported by GBP247,706 in peer-reviewed funding from the Engineering and Physical Sciences Research Council (EPSRC):

- Ghani (PI) & McBride (CI), *Reusability and Dependent Types,* EPSRC, 01/10/09-30/09/13, GBP151,124;
- McBride (PI), *Haskell Types with Added Value*, EPSRC, 01/07/12-30/06/13, GBP96,582).

4. Details of the impact

The above body of research has directly influenced the development of the Glasgow Haskell Compiler (GHC) by providing conceptual clarifications and technical advances showing that dependent types in Haskell were both possible and practical, contradicting the previous consensus which held that an easy-to-use dependently typed Haskell was very unlikely. In 2009, McBride gave a demonstration of SHE at the Haskell Implementer's Workshop, showing for the first time how simulated dependent types could in principle be added to Haskell [**S1**]. Following this, McBride was invited to work with the GHC design team, who were keen to incorporate the features of SHE into GHC. From 2009 to 2013 there then followed regular and direct collaboration between McBride and GHC's development team to work on GHC's type system features [**S1**].

As a result of this collaboration, **GHC extended standard Haskell by incorporating four programmer selectable dependent type extensions**. This in turn has **benefitted programming practice in Haskell by enabling greater functionality**, reaching the world-wide Haskell user base. It has also **led to positive economic impacts for numerous industrial users of GHC**, who have been able to make use of the dependent typing features to produce significantly more reliable software with fewer bugs; as bugs take time and money to fix, this reduces the overall cost of using Haskell.

Incorporating dependent types into the GHC

Collectively, the four extensions informed by McBride's research form the core of the dependently typed features of GHC, allowing Haskell to function like a native dependently typed language:

- DataKinds (allowing data to be used in types, released February 2012);
- PolyKinds (allowing types to abstract over the datatypes they use, released February 2012);
- ConstraintKinds (simplifying Haskell's operator overloading by integrating it better with the type system, released February 2012);
- TypeInType (allowing more type dependency, released May 2016).

Citing **R1**, the paper introducing the implementation of the DataKinds extension notes that: *'Our design is inspired by Conor McBride's Strathclyde Haskell Enhancement (SHE) preprocessor'* [**S2a**]. The coordination page for the four dependent type extensions to GHC listed above cites Lindley and McBride's Hasochism paper [**R4**] as source material providing conceptual clarification on how Haskell's existing feature set can be integrated with dependent types [**S3**]. Dependent types are also being incorporated into the Core language of GHC as an ongoing project, using the findings from the 'Haskell Types with Added Value' project as a *'road map'*, as stated by a member of the GHC development team in a 2014 keynote conference speech [**S2b**]. This keynote speech also referenced **R5** as a source of examples for applications of dependent types, noting that many of the examples were transferrable to GHC; the limitations of dependent types in GHC in comparison to Agda, as demonstrated in **R5**, were discussed as a key motivator for the integration of dependent types in GHC Core language [**S2b**]. A new design for GHC with full-spectrum dependent types was published in 2017 and the specification referenced Gundry's thesis stemming from the project with McBride in multiple places, including the statement that *'Our design of DC is strongly based on two recent dissertations that combine type equality coercions and irrelevant quantification in dependently-typed core calculi'* [**S2c**].

Confirming the role of McBride's research in these developments, a Senior Principal Researcher at Microsoft Research, who was involved in the 'Haskell Types with Numeric Constraints' project and is also a member of the GHC development team, stated:

> *'The effort to incorporate dependently typed features into GHC was originally inspired by discussions between myself and McBride, as well as his work on the Strathclyde Haskell Enhancement (SHE)… the project of making Haskell support richer dependent types continues to this day. These additions enable types to more precisely capture program invariants, allowing programmers to build programs that are correct-by-construction, dramatically reducing errors at compile time and hence run time'* [**S1**].

Enabling greater functionality in professional programming practice

With dependent types incorporated into the design of GHC Core language, every Haskell programme compiled today using GHC uses McBride's research. Due to the ubiquity of GHC, it is impossible to state how many programmes have relied on this research implicitly. Nonetheless, explicit use of the GHC extensions directly underpinned by McBride's research can be identified using Hackage, which, as the main repository of open source Haskell libraries contributed by developers worldwide, provides a strong indication of how Haskell is being used [**S4**]. Of the 15,433 packages on Hackage, as of December 2020:

- 998 (6.5%) use the PolyKinds extension.
- 2321 (15.0%) use the DataKinds extension.
- 1568 (10.2%) use the ConstraintKinds extension.
- 227 (1.5%) use the TypeInType extension.
- 3017 (19.5%) use at least one of them [**S4**].

That 19.5% of a large repository of open source libraries use relatively new extensions to an established language like Haskell demonstrates the notability of McBride's contribution to programming practice. Moreover, a number of these users are sizeable companies, including:

GitHub

GitHub is a Microsoft-owned code hosting platform with over one hundred million code repositories. Semantic, a GitHub tool for code analysis launched in June 2019, is written in Haskell, and its codebase makes substantial use of the DataKinds and ConstraintKinds extensions [**S5**]. Semantic is used to enable interactive code navigation via the GitHub web interface [**S5**]. This is a deployment of significant scale, reaching millions of users worldwide.

Digital Asset

Digital Asset, a US-based financial technology company, developed a smart contract programming language DAML in 2018 and uses the ConstraintKinds extension to simplify the code in their build systems [**S6**]. A Senior Product Architect at Digital Asset has stated that *'ConstraintKinds has been a massive simplification'* having made *'key signatures 3x simpler, and thus more understandable'* [**S6**].

Well-Typed

Well-Typed LLP is a UK-based Haskell consultancy, founded in 2008, that is actively using and supporting the use of dependently typed Haskell in client projects [**S7**]. Examples since August 2013 include the implementation of a major cryptocurrency, in which dependent types are used as part of a spectrum of verification techniques for higher assurance development, which would not be possible without dependently typed features [**S7**].

Facilitating economic benefits for industrial users of Haskell

McBride's work, via GHC, has led to significant economic impacts at a number of major technology companies thanks to improved software correctness and lower development costs.

Habito

Habito is a UK-based online mortgage broker. Their online platform has brokered over GBP5,000,000,000 worth of mortgage applications since its founding in 2015, helping over 350,000 people with their home-financing needs [**S8**]. Habito is almost exclusively written using GHC Haskell with dependently typed extensions, including all four extensions based on McBride's research [**S8**]. As Habito's Chief Technologist states: *'Heavy use of language extensions, particularly those which extend Haskell's type system, is made throughout the Habito codebase… Over 90% of Habito's codebase is deployed to production in customer- and institution-facing products many times a week, where it serves many thousands of visitors and customers each day. The majority of engineers interact with this codebase on a daily basis'* [**S8**].

By using dependent types in their Haskell code, Habito has been able to save time and lower the risk of introducing logical errors into their programming by reducing the need to validate the same piece of data twice when checking business constraints in their refined library. Dependent types are also used by Habito to generate code that would otherwise have to be written by hand, and to automatically generate audit logs, both of which have saved considerable time for and increased performance for the business. Automatic code generation in particular has been credited as having *'enabled changes which have increased performance by orders of magnitude, or added completely new back-ends, in relatively short timespans (weeks instead of months)'* [S8].

Galois

Galois is a US-based research and development company. Galois has used dependently typed Haskell to develop its Crucible software, a retargetable software simulator for analysing software written in multiple programming languages, primarily for security and correctness. Crucible makes essential use of dependently typed Haskell to enforce crucial invariants about the systems it is analysing, preventing bugs in the analysis [S9]. Galois has calculated that Crucible has played an important role in contracts worth over USD22,600,000 (10-2019), demonstrating its significance to the company [S9]. These include use by Amazon to help ensure security of their 's2n' cryptography library and in several projects funded by the US government on Fully Homomorphic Encryption and Software Brittleness.

Google

As of 2019, US-based technology company Google uses dependently typed Haskell to develop a project to produce a hardware artefact and its supporting software infrastructure that will, in the words of two Senior Software Engineers at Google, be *'deployed at a huge scale within Google and serve high volume traffic'* [S10]. This project is being developed with a team of 20 engineers, *'a large team of engineers (even by Google standards)'* [S10]. Language extensions that support dependently typed programming in Haskell are used extensively in this project: by January 2019, the codebase contained 15 modules that use the TypeInType extension, 378 modules that use DataKinds, 51 that use PolyKinds and 69 modules that use ConstraintKinds. Highlighting the benefits of these extensions for the project, the Engineers stated:

> *'Since we are very much a product group required to develop and deploy sophisticated software in a tight timescale we rely heavily on Haskell's expressive power as well as the robustness of language extensions for features like dependently typed programming, many of which have been developed at your department [Computer & Information Science, University of Strathclyde]…These language extensions have significantly increased our productivity and helped to find errors early at compile time…Given the tight deadlines we operate to, the extra productivity afforded by the dependently typed programming features developed by Dr McBride et al. has been of great operational importance to us'* [S10].

5. Sources to corroborate the impact

S1 Corroborating statement from Senior Principal Researcher, Microsoft Research, Cambridge, UK, dated 20 January 2021.

S2 Papers by GHC development team:

a. Yorgey et al. (2012). Giving Haskell a promotion. In *TLDI '12: Proceedings of the 8th ACM SIGPLAN workshop on Types in language design and implementation*. https://doi.org/10.1145/2103786.2103795

b. Weirich, S. (2014). Depending on types. In International Conference on Functional Programming 2014. https://www.youtube.com/watch?v=rhWMhTjQzsU

c. Weirich, S., Voizard, A., de Amorim, P. H., & Eisenberg, R. A. (2017). A specification for dependent types in Haskell. *Proceedings of the ACM on Programming Languages*, 1(ICFP). https://doi.org/10.1145/3110275

S3 Web content from Haskell. Dependent Haskell https://gitlab.haskell.org/ghc/ghc/-/wikis/dependent-haskell [accessed 14 January 2020].

S4 Summary of Hackage report. Raw data available from HEI on request.

S5 Collated web content from GitHub. Semantic code main page (https://bit.ly/3kfUuYQ) and appended examples DataKind and ConstraintsKind use in sematic.

S6 Corroborating statement from former Senior Product Architect, Digital Asset, dated 27 January 2019.

S7 Corroborating statement from Haskell Constultant, Well-Typed, dated 28 January 2019.

S8 Corroborating statement from Chief Technologist, Habito, dated 20 January 2021.

S9 Corroborating statement from Software Engineer and Researcher, Galois Inc., dated 31 October 2019, with experience report by Galois on dependently typed Haskell in industry.

S10 Corroborating statement from Senior Staff Software Engineers, Google, dated 24 January 2019.