Institution: University of Bristol		
Unit of Assessment: 11) Computer Science and Informatics		
Title of case study: Making real-time code navigation for tens of millions of Github users a reality		
Period when the underpinning research was undertaken: 2015 - 2019		
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:**
Nicolas Wu	Lecturer in Computer Science	01/2015 to 05/2019
Period when the claimed impact occurred: 2018 - 2020		
Is this case study continued from a case study submitted in 2014? No		

1. Summary of the impact

GitHub is a large source code hosting service, serving over 40 million users with more than 100 million repositories. GitHub provides powerful code navigation features, such as jump to definition and find all references, for these repositories through its website. In 2018, GitHub ran into a performance “brick wall” for the code navigation features, which was attributed to their Haskell implementation of algebraic effects. Their solution was to scrap this implementation and start afresh using the patterns and algorithms detailed in a series of papers written at the University of Bristol. They saw “immediate, dramatic speedups” and their open source implementation of the research not only enabled code navigation at GitHub, it has been downloaded over 8,000 times by the Haskell community between October 2018 and December 2020, and is used in a number of important open-source projects.

2. Underpinning research

In a pure functional programming language like Haskell, computational effects, such as outputting a string or generating a random number, must be carefully controlled to avoid violating the guarantee that a function will always return the same output for the same input.

There are two widely used approaches to implementing computational effects in Haskell. Traditionally, *monad transformers* have been used but, more recently, programmers are adopting a new approach called *algebraic effects.*

Algebraic effects give an extra layer of abstraction over the traditional monad transformers approach. The programmer writes a single piece of code that uses a computational effect, such as the output-string effect or the generate-random-number effect, but how that effect is *interpreted* (and thus, what happens) may differ depending on the context in which the code is run. For example, in one part of the program, the output-string effect may be interpreted as writing to the console, but in another it may be interpreted as writing to a log file. The programmer specifies which interpretation they want in any given context by writing *effect handlers*.

Algebraic effect handlers provide more flexibility than monad transformers, but with this greater flexibility comes a significant performance penalty. Overcoming this performance penalty was a widely studied open problem at the time of Nicholas Wu’s work in the University of Bristol, with state-of-the-art implementations at least 2-3x slower than monad transformer analogues.

The contributions of Wu and his co-authors are around making the new approach based on effect handlers *as performant* as the traditional approach based on monad transformers (which is considered optimal). This work was published in the papers *Fusion For Free: Efficient Algebraic Effect Handlers* in 2015 [1] and *Monad Transformers and Modular Algebraic Effects: What Binds them Together* in 2019 [2], written whilst Wu was a lecturer at the UoB.

There are three key insights in these works that have enabled the impact at GitHub:

- The first is the attribution of the performance penalty observed in algebraic effects' approach to the need for the effect handlers to traverse the description of the effectful computation multiple times, once for each handler (each kind of effect) [1]. This contrasted with previous work, which had assumed that the time to complete any given traversal was the most significant factor.
- The second was in observing that the traversals are a special kind of procedure – a fold. The theoretical properties of folds have been well studied and it is known that folds enjoy a property called *fusion*, which is the ability to combine several such procedures into one without significant loss of performance. Thus, Wu and his co-authors gave algorithms to automatically coalesce any number of effect handlers into one, which could therefore interpret the specification of the effectful computation in only a single traversal [1].
- Finally, the works describe a sophisticated implementation strategy for algebraic effects that enables the performance improvements to be seen in practice. This relies on two key insights. The first is regarding the behaviour of the Haskell compiler with respect to fusion opportunities: good speed-ups will only be seen if the effectful programs are polymorphic in the term monad and all the algebras are typeclass instances. The second is that, for a certain common class of algebraic effects, the whole algebraic effect apparatus may be automatically translated into an implementation based on monad transformers, which is much better understood and optimised by the compiler [2].

3. References to the research

[1] **Nicolas Wu** and Tom Schrijvers. 2015. Fusion for Free – Efficient Algebraic Effect Handlers. In *Mathematics of Program Construction (LNCS),* Ralf Hinze and Janis Voigtländer (Eds.), Vol. 9129. Springer, 302–322. DOI: 10.1007/978-3-319-19797-5

[2] Tom Schrijvers, Maciej Piróg, **Nicolas Wu** and Mauro Jaskelioff, 2019. Monad Transformers and Modular Algebraic Effects: What Binds Them Together. In *Proceedings of the 2019 ACM SIGPLAN Symposium on Haskell (Haskell 2014),* Richard A. Eisenberg (Ed.), ACM, 98—113. DOI: 10.1145/3331545.3342595
This work was conducted at Bristol and submitted to the 2019 ACM SIGPLAN Symposium on Haskell in May 2019, before the conference in August 2019

4. Details of the impact

GitHub is a large source code repository hosting service, serving over 43 million users and 1.5 million companies with more than 100 million repositories [A]. The web interface, github.com, is among the top 100 websites ranked by global Internet traffic and engagement [B]. In October 2018, Microsoft acquired the company for USD7.5 billion [C].

GitHub's semantic code team are responsible for the code navigation features, including *jump-to-definition* and *find all references* that allow users of the GitHub website to browse and understand repositories effectively.

The software that implements these features is called *Semantic* and is written in Haskell. The underlying methodology implemented in the software is a programme analysis technique, proposed in an ICFP paper of 2017 called *Abstracting Definitional Interpreters (Functional Pearl);* written by Darais, Labich, Nguyen and Van Horn at the University of Maryland. According to this approach, abstract interpreters are constructed from a common pattern by varying the kinds of computational effects that each of the interpreters can perform. The abstract interpreter walks the syntax trees of the code, but – unlike a normal interpreter – rather than generate and execute the code, it instead records and analyses information derived from it. This process of recording and analysing can be seen as performing certain computational effects at different points during the syntax tree traversal. Different computational effects correspond to different facets of the program analysis.

Consequently, having a flexible and extensible computational effects system is essential for allowing a natural implementation of the ideas of that paper. At GitHub, in 2016, an initial implementation was based on the algebraic effects system described in the work *Extensible Effects: An Alternative to Monad Transformers* by Kiselyov, Sabry and Swords. However, by 2018 it became apparent that this approach to implementing algebraic effects led to a performance "brick wall" [D]. GitHub contains an enormous variety of code repositories, some of which are very large indeed with *Semantic's* abstract interpreters required to visit and perform computational effects at millions upon millions of syntax tree nodes.

Increased speed of task execution

The Semantic Code team recognised a solution in the work of Wu and Schrijvers [1, 2]. The team directly implemented the definitions and algorithms of these papers in an algebraic effects library for Haskell called *fused-effects,* and this wholesale replaced their previous implementation. A senior engineer on the Github Semantic code team, reports that *"after switching from our effect library without fusion to one that displayed this property, I observed a factor of 250 speedups, and needless to say that's pretty extraordinary, maybe the most extraordinary speedup I have seen in my career"* [D].

The reach of this work is enormous. Every time a developer, anywhere in the world, makes a change to their code (written in a supported language) and commits that change to the master branch on GitHub, *fused-effects* is executing to analyse and index the code that was changed. Taking just one of those 100 million repositories as an example: in September 2020 alone there were 922 commits to the master branch of the Google Kubernetes repository, with 63 different contributors around the globe making changes to over 2,000 files [E]. GitHub write "*Semantic would not have been able to support this kind of throughput, and could not have supported as wide a range of programming languages as quickly, had we not discovered and adopted the research of Dr Wu*." [G]

This analysis powers code navigation features on the website, like the ability to click on an identifier in the source code and jump to its definition, and the ability to immediately look up all places in the code where a particular definition is used. One of the many enthusiastic new users of this feature speculated on twitter: the amount of developer time it saves *"must be in the hundreds of hours across the planet per day!"* [F] GitHub's implementation, which originally supported analysing code written in Go, Python and Ruby has been such a “hit with developers” [G] that the company have quickly moved to add support for Javascript, TypeScript, PHP, Java, JSON, JSX and CodeQL too, with more languages currently in the pipeline. It is now “*an integral part of the GitHub offering for these languages*” [G].

Open source access

GitHub’s *fused-effects* implementation of Wu’s work at University of Bristol has been open sourced. It is also available from the Haskell package repository *Hackage,* where it has had over 8,000 downloads since late 2018 and over 600 during the month that this case was being written (as of May 2020) [H]. Notable projects where the library has been used beyond Github are the Aura package manager for Arch Linux and the Now-Haskell client library for AWS Lambda [H].

5. Sources to corroborate the impact

[A] GitHub (2020). i) About: GitHub is how people build software and ii) Users [Accessed 1 June 2020].

[B] Alexa (2020). github.com - Competitive Analysis, Marketing Mix and Traffic. *The site github.com has an “Alexa Rank” of 90,* [Accessed 22 Sept 2020].

[C] Microsoft News Center (2018). Microsoft to acquire GitHub for $7.5 billion [Accessed 22 September 2020].

[D] Strange Loop (2019). Talk: Building Haskell programs with fused effects (Senior Engineer, Semantic Code team, GitHub) [Accessed 14 May 2020].
Work of Wu and Shrijvers mentioned at 26’45”, and quote from Senior Engineer at 34’55”

[E] Github Kubernetes, https://github.com/kubernetes/kubernetes/compare/master@%7B06-01-20%7D...master@%7B06-30-20%7D, [Accessed 22 September 2020].

[F] Twitter (2019). Quote for Github’s features: https://twitter.com/w3af/status/1139246954947518464, [Accessed 22 September 2020].

[G] GitHub (2020). Corroborating statement – Senior Engineer Manager, Semantic Code

[H] Hackage: The Haskell Package Repository (2020). fused-effects: A fast, flexible, fused effect system [Accessed 15 May 2020].