Melchior: Web Programming in Haskell

Kieran Gorman, Timothy Jones, Lindsay Groves

Victoria University of Wellington, New Zealand
{Kieran.Gorman,tim,lindsay}@ecs.vuw.ac.nz

Developing JavaScript applications requires manipulating the state of the underlying DOM using events which are applied with no static guarantee of consistency or correctness. Using the browser's event loop means that actions are performed sequentially with no direct communication between them, creating applications that are the result of many combined side-effects. This makes program correctness difficult to reason about, with large applications being formed from many individual pieces, each of which may invalidate an assumed invariant of another [1].

Reactive programming provides an abstraction of event driven programming that can remove direct use of an event loop. The Haskell programming language's type system provides powerful static constraints on types and effects within programs. We have built the *Melchior* framework to explore functional reactive programming in Haskell for web browsers. This consists of a JavaScript runtime that exploits the underlying event engine to provide reactive values, as well as a Haskell API that allows access and interaction with a web page. The runtime provides composable asynchronous actions, while the Haskell layer provides type safety to their composition.

While reactivity has been explored in JavaScript with libraries such as Flapjax [4] and Arrowlets [3], no solution currently exists that also contributes the static guarantees offered by Haskell. Additionally, the use of Haskell means that common code can be shared between the client and server, allowing for these guarantees to be persisted from user interaction through to server logic and database storage. While there a number of custom languages to allow for functional reactive programming on the web, there is no stable implementation of a runtime that maintains the lazy and pure semantics of the language.

The use of Haskell also allows well typed access to the DOM, which makes aspects of operation and manipulation of elements explicit. For instance, the 'value' field only exists for input (and related) elements. Modelling this field access as a function of type $InputElement \rightarrow String$ ensures that one cannot erroneously attempt to access the value of an element which cannot have one.

This work creates a modern FRP domain specific language in an existing language decoupled from the document object model. This makes Melchior distinct from both projects such as Flapjax [4], and Elm, which is a bespoke, Haskell inspired language for FRP that couples the entire presentation layer [2]. It is also unlike modern model-view-controller JavaScript frameworks like Ember.js and Angular.js as it does not require any custom HTML elements or attributes.

References

[1] Cooper, E., Lindley, S., Wadler, P., and Yallop, J. Links: Web programming without tiers. In *Formal Methods for Components and Objects*, F. Boer, M. Bonsangue, S. Graf, and W.-P. Roever, Eds., vol. 4709 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 266–296.

[2] Czaplicki, E., and Chong, S. Asynchronous functional reactive programming for GUIs. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation* (New York, NY, USA, 2013), PLDI '13, ACM, pp. 411–422.

[3] Khoo, Y. P., Hicks, M., Foster, J. S., and Sazawal, V. Directing javascript with arrows. In *Proceedings of the 5th Symposium on Dynamic Languages* (New York, NY, USA, 2009), DLS '09, ACM, pp. 49–58.

[4] Meyerovich, L. A., Guha, A., Baskin, J., Cooper, G. H., Greenberg, M., Bromfield, A., and Krishnamurthi, S. Flapjax: A programming language for Ajax applications. *SIGPLAN Not. 44*, 10 (Oct. 2009), 1–20.