Monadic Rig-Module Linearity

Kim-Ee Yeoh
Dcolour.com Research

Probabilistic functional programming [RP02, EK06] use monads to encode the linearity properties of a vector space. Analysis of the unit and multiplication operations of the monad reveals that not all of the properties of a vector space are utilized. It suffices to work over a less restrictive mathematical structure, in particular a module over a rig, viz. a riNg without Negatives, also known as a semiring. Rigs are arguably more useful than rings for modelling computation structures, as demonstrated by the surveys [L07, P98].

We show how to implement the linearity properties of rig-modules using a monad in Haskell. Our work differs from [P07] in that we employ type improvement constraints [KLS04] in order to treat rigs and rig-modules uniformly. Multilinear maps and basis-oriented tensor spaces are elegantly expressed. The Haskell code stays short and concise. As an example of rig-module applications, we solve the all-pairs shortest paths problem in graph theory via the standard min-plus rig initially and subsequently via a customized rig that encodes the actual one or more shortest paths.

References:

[RP02] N. Ramsey and A. Pfeffer. Stochastic Lambda Calculus and Monads of Probability Distributions. In *29th ACM Symposium on Principles of Programming Languages*, pages 154–165. ACM Press, 2002.

[EK06] M. Erwig and S. Kollmansberger. Probabilistic Functional Programming in Haskell. *Journal of Functional Programming*, 16(1):21-34, 2006.

[L07] G. L. Litvinov. Maslov dequantization, idempotent and tropical mathematics: A brief introduction. *Journal of Mathematical Sciences*, 140(3):426-444, 2007.

[P98] J.-E. Pin. Tropical Semirings. In *Idempotency*, pages 50-69. J. Gunawardena (éd.), Cambridge University Press, 1998.

[KLS04] O. Kiselyov, R. Lammel, Ralf, and K. Schupke. Strongly Typed Heterogeneous Collections. In *Haskell '04: Proceedings of the ACM SIGPLAN workshop on Haskell*, pages 96-107. ACM Press, 2004.

[P07] D. Piponi. How to write tolerably efficient optimzation code without really trying, 2007. `http://sigfpe.blogspot.com/2007/06/how-to-write-tolerably-efficient.html`