Praktische Totale Funktionale Programmierung im Church-Stil

Baltasar Trancón y Widemann

Universität Bayreuth

Zusammenfassung. Ein Ausspruch David Turners [1] besagt, dass konventionelle Turing-vollständige funktionale Programmiersprachen den mathematischen Funktionsbegriff weniger gut abbilden, als es zunächst den Anschein hat. Der wesentliche Unterschied ist offensichtlich die Partialität und Nichtterminationauf der einen, und die Totalität und Zeitlosigkeit auf der anderen Seite. Diese Unterscheidung macht sich besonders im Bereich formaler und modellbasierter Methoden der Softwaretechnik bemerkbar, wo ein und derselbe Text als mathematisches Modell und als ausführbarer Prototyp gelesen werden kann.

Reduziert man andererseits die Semantik auf einen puren Kalkül totaler Funktionen, z.B. Coquands *Calculus of Constructions*, muss man zunächst auf viele bewährte Features der funktionalen Programmierung verzichten: rekursive Definitionsgleichungen von Datentypen und Funktionen, Typinferenz, Typklassen. Die experimentelle Programmiersprache Tofu ist bemüht, diesen ergonomischen Verlust so weit wie möglich zu kompensieren, ohne bei der semantischen Fundierung Kompromisse zu machen. Es werden ausgewählte Features und Design-Prinzipien von Tofu als Alternativen zu den erwähnten Konstrukten vorgestellt und kritisch verglichen.

Literatur

1. Turner, D. (2004) *Total Functional Programming.* Journal of Universal Computer Science **10**(7). DOI 10.3217/jucs-010-07-0751.