Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 319 (2015) 403–421

www.elsevier.com/locate/entcs

Stateful Runners of Effectful Computations

Tarmo Uustalu[1]

Institute of Cybernetics, Tallinn University of Technology,
Akadeemia tee 21, 12618 Tallinn, Estonia

Abstract

What structure is required of a set so that computations in a given notion of computation can be run statefully with this set as the state set? For running nondeterministic computations statefully, a resolver structure is needed; for interactive I/O computations, a "responder-listener" structure is necessary; to be able to serve stateful computations, the set must carry the structure of a lens. We show that, in general, to be a stateful runner of computations for a monad corresponding to a Lawvere theory (defined as a set equipped with a monad morphism between the given monad and the state monad for this set) is the same as to be a comodel of the theory, i.e., a coalgebra of the corresponding comonad. We work out a number of instances of this observation and also compare runners to handlers.

Keywords: effects, monads, Lawvere theories, comodels, state monads, handlers

1 Introduction

This paper is about Moggi's monad-based and Plotkin and Power's Lawvere theories based approaches to effectful computation [8,10].

Given a monad (T, η, μ), a computation of a value in X is an element of $T\,X$. Computations are there to compute values, so we consider it natural to wish to extract these values, to *run* computations. Ideally, we might want to have at our disposal a polymorphic function $\theta : \forall X.\, T\,X \to X$ for extracting values from computations, but this is generally too much to ask (although it is possible, e.g., for writer monads).

However we can often produce a value, if we are allowed to rely on some input—think of it as an initial state—drawn from some set C with suitable structure. For example, if we have a finitely nondeterministic computation in the sense of a binary wellfounded leaf tree, a bitstream can be used to identify a leaf. As running should reasonably be compositional in the sense that running the sequence of two computations should be the same as composing two runs, a run should not only

[1] tarmo@cs.ioc.ee

http://dx.doi.org/10.1016/j.entcs.2015.12.024

depend on an initial state, but also return a final state (that can serve as the initial state for another run). In the case of nondeterminism and bitstreams, the final state could be the remainder of the bitstream provided as the initial state. So in general we might want to look for a polymorphic function $\theta : \forall X.\, T\, X \to T^C\, X$ where (T^C, η^C, μ^C) is the state monad for C as the state set. The compositionality we want amounts to θ being not just a natural transformation, but a monad morphism.

In this paper, we answer the question of when a set C can be used to run computations in a monad (T, η, μ) statefully, assuming that the monad corresponds to a Lawvere theory. The answer is: C has to carry a comodel of the Lawvere theory (i.e., a coalgebra of the corresponding comonad). We spell out a number of instances of this generality, for nondeterminism, interactive I/O and stateful computations. This is an easy exercise, but the results are quite instructive, we find. For some versions of nondeterminism, for instance, runners can only recover a part of the information in a given computation; other versions of nondeterminism admit only trivial runners that reveal nothing about the computation. So some variations of nondeterminism are inherently more operational than others.

Runners are somewhat similar to handlers, but one bigger difference is that runners are polymorphic in the value set. For example, handling allows one to extract a value from a nondeterministic computation (a binary wellfounded leaf tree) over a specific value set that carries a binary operation by folding this operation over the leaf labels. (If for us a nondeterministic computation is a nonempty list of values, this operation must be associative.) Running does not allow such things. In our view, the pragmatics of handlers and runners are different: handlers are a programming language construct, but runners are compilation schemes.

The paper is organized as follows. In Section 2, we review the few basic facts about Lawvere theories, models and comodels that we need. In Section 3, we show that stateful runners for a monad corresponding to a Lawvere theory are in a bijection with comodels of the theory (coalgebras of the corresponding comonad). We also compare this observation to a fact about monad morphisms to continuation monads—a different type of runners. In Section 4, we work out the instances for nondeterminism, interactive I/O and stateful computation. Just before concluding, in Section 5, we compare runners to handlers.

2 Lawvere theories, models, comodels

We begin by reviewing the most basic definitions and facts about finitary Lawvere theories and models (for a proper exposition, see, e.g., [6]) as well as Power's comodels [13,11]. Countable Lawvere theories and κ-ary Lawvere theories for a regular cardinal κ are defined analogously.

Lawvere theories

A *(finitary) Lawvere theory* is given by a small category $\mathbb{L}$ with finite products and a functor $L : \mathbb{F}^{\mathrm{op}} \to \mathbb{L}$ that is identity on objects and strictly preserves the finite products of $\mathbb{F}^{\mathrm{op}}$.