Prompt, Lazy Assertions in Haskell
A Monadic Approach

Olaf Chitil
University of Kent, UK
oc@kent.ac.uk

Frank Huch
University of Kiel, Germany
fhu@informatik.uni-kiel.de

Abstract

Assertions check expected properties of run-time values without disrupting the normal computation of a program. Unfortunately, in lazy languages like Haskell checking assertions may be problematic since the check will evaluate data structures although they are not demanded by the computation of the program. We obtain a strict program. This will especially be problematic if the programmer makes use of laziness, e.g. uses infinite data structures.

As a solution, the evaluation of an assertion has to respect how far corresponding data structures are evaluated. We developed a pattern logic for enriching Haskell programs with assertions. In this pattern logic expected properties combine pattern matching with logical operations and predicates. The assertions are both lazy, that is, they do not force evaluation but only examine what is evaluated by other parts of the program, and prompt, that is, assertion failure is reported as early as possible, before a faulty value is used by the main computation.

However, in comparison to Haskell this logic is very restricted and some properties are very difficult to explain. In this talk, we present a monadic extension of this pattern logic which allows the definition of assertions similar to arbitrary Haskell functions. The evaluation of these assertions is also lazy and prompt. Its implementation is based on lazy observations and computing/suspending assertion checks in continuation-based coroutines.