Embedded resource tool in Haskell

Attila Góbi, Tamás Kozsik, Barnabás Králik

Dept. of Programming Languages and Compilers, Eötvös Loránd University

{gobi,kto,kralikba}@elte.hu

In our previous work [1], we have created a way to check size annotations of higher-order polymorphic functional programs supporting nested lists. By extending the lambda-calculus, these annotations are able to express the relations between sizes of arguments and those of the cooresponding results of functions. These relations are exact, and can be non-linear and non-monothonic polynomials.

We provided a way for verification condition generation as well. This paper focuses on how it is possible to implement the proposed verification condition generation, and how close we can get to a size-checking functional programming language.

Our approach is based on creating a deep embedding of the lambda calculus into Haskell, and creating two different interpreters of the embedded language. The first one executes the code, while the second one is able to infer verification conditions from the code. VCs are then checked by the Z3 solver [2].

References

[1] Góbi, A., Shkaravska, O., van Eekelen, M.: *Higher-order size checking without subtyping* 7829 Trends in Functional Programming Springer Berlin Heidelberg Lecture Notes in Computer Science 53–68 (2013)

[2] De Moura, L., Bjørner, N.: *Z3: An efficient smt solver* Tools and Algorithms for the Construction and Analysis of Systems 337–340 (2008)