When Desugaring Makes Your Code Sour
Reducing The Number Of False Positives and Negatives With External Tool

Artúr Poór[a]

[a]Eötvös Loránd University
poor_a@inf.elte.hu

Abstract

Programming language designers tend to continuously seek ways to make their languages convenient to use. Such a way is to introduce syntactic sugars, which are alternative but more terse notations for existing language constructs, and defines formal rewriting rules for those syntactic sugars which desugar code by reducing into equivalent code snippets of the base language. In case of Scala, this may come at the cost of inaccurate compiler checks, particularly check for defined but unused variables. When used together, the combination of independent rewrite rules may introduce both false positive and negative compiler warnings. Conversely, codes produced by other rewriting rules would cause so many false positive warnings that the compiler omit checks making those language constructs prone to programmer errors.

In this paper we introduce an external static code analyser for Scala programs that produces more accurate analysis by performing it on the original code with syntacic sugars.

Keywords: static analysis, syntactic sugars, software correctness, compiler warnings, Scala

MSC: 68

References

[1] Odersky, M., Spoon, L., Venners, B. Programming in Scala, *Artima Press*, 2016, ISBN: 0981531687

[2] Nielson, F., Nielson, H., Hankin, C. Principles of Program Analysis, *Springer-Verlag*, 2004, ISBN: 3540654100