TID – Modern Theoretical Computer Science
Type Systems, Their Models and Usage in Decompilation

Peter Matula
xmatul01[at]stud.fit.vutbr.cz

October 12, 2013

Keywords

type system, lambda calculus, simply typed lambda calculus, type inference, decompilation, type recovery

Abstract

Data type information is one of the key characteristics that distinguish low-level machine code from high-level source code. Types are important for expression of the program in high-level terms – they partition the domain of program semantics and partition the data into distinct objects.

This presentation introduces the concept of the *type system*, which assigns type property to program constructs. It allows construction of type-checking algorithms implementing data *type inference* – conclusion about the types of the objects based on how they are used. Formal system of *lambda calculus* is used as a notation for stating the semantic properties of the programming languages. To incorporate type laws, extended system called *simply typed lambda calculus* is introduced and described in depth. Other possible extensions like System F introducing subtype polymorphism characteristic for object-oriented languages are mentioned.

Even though type inference is typical for functional programming languages, the same principles can be used for type recovery by the decompiler. Decompilation is a process of transforming a machine code into a higher-level programming language. It consists of series of analysis, one of which is the type analysis, that tries to associate each piece of data with a high-level type. Because there are no type information in input machine code, type inference from the context of objects usage similar to type-checking can be exploited.

References

[1] JongHyup Lee, Thanassis Avgerinos, and David Brumley. Tie: Principled reverse engineering of types in binary programs. In *NDSS*. The Internet Society, 2011.

[2] Benjamin C. Pierce. *Types and programming languages.* MIT Press, Cambridge, MA, USA, 2002.

[3] David A. Schmidt. *The structure of typed programming languages.* Foundations of computing series. MIT Press, 1994.