From Object-Oriented Code with Assertions to Behavioural Types

Cláudio Vasconcelos António Ravara
NOVA LINCS, DI-FCT, Universidade NOVA de Lisboa, Portugal

12 September 2016

The aim of our work is to take a step in the direction of infering behavioural types for object-oriented languages, bridging the world of programming with assertions with the world of behavioural typed programming [1], in particular in Java. While the former is becoming increasingly popular and well supported (being part of the Java language for more than a decade now[1]), the latter has a growing impact and will probably soon be incorporated in mainstream languages. We developed an algorithm that converts Java with assertions in a form of behavioural types (henceforth called *usage*, a textual representation of a finite automata). Usages represent all the safe sequences of method calls and are (enhanced forms of) class types, checkable at compile-time.

We present a behavioural type inference approach that, from a program written in Mool [2, 3] with assertions, generates the usage types necessary for the program to have its behaviour statically checked by a type system. We developed and implemented algorithms to automatically perform the transformation [6].[2] The tool starts by generating a permissive and nondeterministic state machine representing a typestate, based on the assertions on the code. The tool then translates the generated typestates into usages. In the end, it defines the usage state that each object of the class starts with by using the assertions and the usages obtained in the previous stage. This tool is composed by three algorithms, with the first two adapted from algorithms presented in other works [4, 5], and the third one being original.

In a nutshell, our approach is the following: given a program written in a subset of Java, fully annotated with assertions, if such annotations are correct our algorithm returns its usage. The goal is to provide developers with abstractions extracted from the code to represent its behaviour. Furthermore, these abstractions can be attached to the code and then statically verified.

Although behavioural types are intuitive, informative, and easy to verify, the assurances given by type systems using them and by proof systems based on assertions are complementary. Many type systems do not guarantee, for instance, NullPointerExceptions. Furthermore, assertions are part of Java and are used by many developers and present in many code supporting running applications. Our algorithm allow thus for analysing statically this legacy code, checking even automatically its correctness, and gives additional support to developers not familiarised with behavioural types.

References

[1] Davide Ancona, Viviana Bono, Mario Bravetti, Joana Campos, Pierre-Malo Deniélou, Nils Gesbert, Elena Giachino, Raymond Hu, Einar Broch Johnsen, Francisco Martins, Fabrizio Montesi, Rumyana Neykova, Vasco T. Vasconcelos, and Nobuko Yoshida. Behavioral types in programming languages. *Foundations and Trends in Programming Languages*, 2016.

[2] Joana Campos and Vasco Thudichum Vasconcelos. Channels as objects in concurrent object-oriented programming. In *Proceedings of the Third Workshop on Programming Language*

[1] `http://docs.oracle.com/javase/7/docs/technotes/guides/language/assert.html`
[2] Available at `http://usinfer.sourceforge.net/`

Approaches to Concurrency and communication-cEntric Software, volume 69 of *EPTCS*, pages 12–28, 2011.

[3] Joana Correia Campos. Linear and shared objects in concurrent programming. Master's thesis, University of Lisbon, 2010.

[4] Guido De Caso, Victor Braberman, Diego Garbervetsky, and Sebastian Uchitel. Enabledness-based program abstractions for behavior validation. *ACM Transactions on Software Engineering and Methodology*, 22(3):1–46, 2013.

[5] Peter Collingbourne and Paul H. J. Kelly. Inference of session types from control flow. *Electronic Notes in Theorectical Computer Science*, 238(6):15–40, 2010.

[6] Cláudio Vasconcelos and António Ravara. From Object-Oriented Code with Assertions to Behavioural Types. In *Proceedings of the Portuguese Simposium in Informatics*. INFORUM, 2016. URL: http://usinfer.sourceforge.net/articles/inforum2016.pdf.