A Constraint-Solving Approach to Faust Program Type Checking

Imré Frotier de la Messelière[1], Pierre Jouvelot[1], Jean-Pierre Talpin[2]

[1]MINES ParisTech, PSL Research University
[2]INRIA, IRISA

July 4, 2014

Presentation Proposal for "Constraint Programming Meets Verification 2014 Workhsop"

In the Faust programming language, a domain-specific language dedicated to audio signal processing, signal processors take synchronous audio data streams as input and generate new transformed output signals, typically at an audio rate of 44.1 kHz. To handle such stringent processing requirements, Faust design adopts a two-level structure: (1) a lamba-calculus-like macro language expands high-level Faust constructs into a (2) low-level strongly-typed core. Even though the Faust compiler already sports a typechecker, we have specified and are currently implementing a new type inference algorithm for Faust, taking inspiration from the algorithm W of Hindley-Milner [3] and the algebraic reconstruction approach of Jouvelot and Gifford [1]. Our ultimate goal is to provide Faust with a formally proven static typing system, thus making it more reliable and efficient.

Our type inference algorithm is organized in two main parts: a classic type inference algorithm, coupled with the generation of typing constraints, and a solver to determine if the resulting constraints system is decidable and to provide a mapping yielding the type of Faust expressions. Solving will be handled by existing solvers, using SMT-LIB as a common representation framework for constraints. An implementation in OCaml is currently being developped. This first prototype will then be rewritten in C++ in order to be included within the current Faust compiler, so that it may be made available along with the official Faust release.

In the present form, this typing algorithm is handling the monorate version of Faust and will soon handle the multirate version as well, which we will present at the workshop. Multirate handling [2] is a requirement put forth in the next version of Faust, which will include vector capabilities: as vectors are built from an incoming signal, the output signal rate will be decimated accordingly, while serialization will operate in a dual manner. This extension requires an update of the typing specifications so that they may handle the presence of frequency information in Faust types. These new features have a significant impact on the constraints to be be handled, adding constraints on rates to other constraints on types and values, which we will discuss.

We believe our research project is interesting to the Constraint Programming community and may benefit from its inputs, since we intend to use constraints as the foundation of the whole typing process, contrarily to more standard approaches that adopt more ad-hoc techniques based on substitutions and principal types. This will entail the creation of large and multi-sorted constraint systems that will need to be processed efficiently to make the whole approach viable. We hope to be able to present at the workshop some ideas about the structure, size and specificities of the Faust-induced typing constraints.

Bibliography

[1] Jouvelot, P., and Gifford, D. K. Algebraic Reconstruction of Types and Effects. In *Proceedings of the 1991 ACM Conference on Principles of Programming Languages*. ACM, New-York, 1991.

[2] Jouvelot, P., and Orlarey, Y. Dependent vector types for data structuring in multirate Faust. In *Computer Languages, Systems & Structures Journal*. Elsevier, 2011.

[3] MILNER, R. A Theory for type polymorphism in programming. In *Journal of Computer and Systems Sciences*, Vol. 17, pages 348-375. 1978.