ADT4HPC: Algebraic Data Types for High Performance Computing

Thaïs Baudon*[1], Gabriel Radanne[2], and Laure Gonnord[3]

[1]LIP – École normale supérieure - Lyon (ENS Lyon), CNRS – France
[2]LIP – École normale supérieure - Lyon (ENS Lyon), CNRS, INRIA – France
[3]Grenoble-INP / LCIS – LCIS, Université Grenoble Alpes, Grenoble-INP – France

Résumé

Initially present only in functional languages such as OCaml and Haskell, Algebraic Data Types have now become pervasive in mainstream languages, providing nice data abstractions and an elegant way to express functions though pattern-matching. Numerous approaches have been designed to compile rich pattern matching to cleverly designed, efficient decision trees. However, these approaches are specific to a choice of internal memory representation which must accommodate garbage-collection and polymorphism.

ADTs now appear in languages more liberal in their memory representation such as Rust. Notably, Rust is now introducing more and more optimizations of the memory layout of Algebraic Data Types. As memory representation and compilation are interdependent, it raises the question of pattern matching compilation in the presence of non-regular, potentially customized, memory layouts.

In this article, we present Knit&Frog, a framework to compile pattern-matching for monomorphic ADTs, parametrized by an arbitrary memory representation. We propose a novel way to describe choices of memory representation along with a validity condition under which we prove the correctness of our compilation scheme. The approach is implemented in a prototype tool ribbit.

*Intervenant