8a. Overview and Goals

Martin Alfaro
PhD in Economics

In the upcoming chapters, we'll focus on two essential aspects for performance: type stability and reductions in memory allocation. These core principles represent the most basic procedures to achieve high performance, thus acting as the starting point for further optimizations.

This chapter in particular focuses on type stability, whose importance for Julia can't be overstated —**any attempt to generate fast code without ensuring type stability is destined to fail**.

At its core, type stability is rooted in how computers execute operations at a fundamental level. Specifically, regardless of the programming language used, the approach to computing operations differs depending on the inputs' types. This means, for instance, that the internal process for integer operations differs from computations based on floating-point numbers.

The consequence of this feature for performance is that speed demands the identification of concrete types for each variable. With this information available, the computation method can be specialized. Instead, if concrete types can't be identified, the code generated must accommodate multiple potential approaches, one for each possible combination of input types. This introduces additional runtime checks and type conversions, significantly degrading execution speed.

The discussion of type stability will be intertwined with functions, as *type stability requires wrapping code in function as a prerequisite*. The reason for this is that Julia only attempts to infer the types of variables within a function. Wrapping code in a function is only a necessary condition for type stability, and the chapter will provide additional conditions to guarantee the property.