ALGORITHMIC COMPLETENESS OF IMPERATIVE LANGUAGES

Yoann Marquer

University Paris-Est Créteil

A function is effectively calculable if there exists an effective method for calculating its output for a given set of inputs. This effective method is called an algorithm. According to the Church-Turing thesis the effectively calculable functions on the integers are the functions computable by a Turing machine. A computation model is Turing complete if it can compute all these functions. For example the Turing machines, the lambda-calculus or the recursive functions are Turing complete. More generally, a model is functionally complete for a given class of functions if it can compute all the functions of the class. That means that it is possible to define in the computation model at least one algorithm for calculating the function.

Loïc Colson proved that the optimal algorithm for the `min` function cannot be used in the computation model of the primitive recursive functions, and Moschovakis did the same with the `gcd` function. But these functions are primitive recursive. That means that the computation model can compute one of their algorithms, not all their algorithms. It is functionally complete but not algorithmically complete.

Yuri Gurevich axiomatized the algorithms with sequential time, abstract data structure and bounded exploration, and he characterized these sequential algorithms with the Abstract State Machines (ASMs). So, a computation model is algorithmically complete if it can simulate the abstract state machines. The simulation is said to be faithful if it allows only a constant temporal dilatation and a bounded number of new temporary variables.

I formalized imperative languages with a state transition system allowing updates, and commands for the control flow. The transition system verifies the sequential time, the states are the same as the ASMs, and the exploration is bounded.

I proved that with the `update`, `if` and `while` commands every ASM can be faithfully simulated, so this computation model is algorithmically complete. That means that every sequential algorithm can be written in an imperative language with these commands.

But some interesting subclass of algorithms can be obtained in the same way with a less powerful language. For example, every ASM with a primitive recursive complexity and data structure can be faithfully simulated with the, `if`, `loop`, and `exit` commands. So, for the studied subclasses, it is possible to determine the commands needed to write the optimal algorithm of the desired function.