Mechanisms for side-effect free I/O

Ulrich Neumerkel

Technische Universität Wien, Austria
ulrich@complang.tuwien.ac.at

Constraints, coroutining, and other advanced language features entered mainstream Prolog through extensible unification via attributed variables. Still missing are I/O streams, originally developed in concurrent logic programming. They give us first and foremostly side effect free input from files. Compared to functional languages, side-effect free I/O hasn't received the same attention in Prolog. Prolog programmers still have to rely on state-ridden language constructs to perform I/O. Providing side-effect free I/O for Prolog, requires to resolve several problems. In particular the presence of backtracking which we do not want to sacrifice imposes significant restrictions. Our approach relies essentially on four different elements. The ISO I/O model, DCGs, coroutining via freeze/2 and a recent cleanup mechanism. While all of these elements are considered to be well understood, it is their interaction that complicates the situation. For example, the interaction of couroutining and DCGs forces us to reconsider commonly held beliefs about the steadfastness of DCGs. The implicit freeing of resources facilitated by the cooperation of DCGs and cleanup mechanisms requires several adjustments to both. Further fine print is required to address asynchronous events like interrupts, timeouts and resource errors.

We conjecture that a system needs all of the discussed mechanisms to provide reliable and efficient side effect free I/O. This might be the reason why side effect free I/O has not yet found broader acceptance. We will discuss current implementations and focus on the challenges a system implementer has to face.