Reversible Programming Languages

Robert Glück

University of Copenhagen

The principles of reversible programming languages are explicated and illustrated with reference to the design of a high-level imperative language, Janus. The fundamental properties for such languages include backward as well as forward determinism and reversible updates of data. The unique design features of the language include explicit postcondition assertions, direct access to an inverse semantics and the possibility of clean (i.e., garbage-free) computation of injective functions. We suggest the clean simulation of reversible Turing machines as a criterion for computing strength of reversible languages, and demonstrate this for Janus. We show the practicality of the language by implementation of a reversible fast Fourier transform. Our results indicate that the reversible programming paradigm has fundamental properties that are relevant to many different areas of computer science.

Joint work with Tetsuo Yokoyama and Holger Bock Axelsen.