“Putback” is the Essence of Bidirectional Programming

Sebastian Fischer

(joint work with Zhenjiang Hu and Hugo Pacheco)

Bidirectional transformations, programs with a forward and a backward transformation that maintain consistency between input and output, are routinely written in ways that do not let programmers specify their behavior completely. Several bidirectional programming languages exist to aid programmers in writing bidirectional transformations with increased maintainability but decreased expressiveness.

Such languages allow programmers to write bidirectional transformations as one program for both directions, which is easier to maintain than separate programs for each direction. However, the maintainability provided by existing bidirectional languages comes at the cost of expressiveness because the ambiguity of synchronization is solved by default strategies which are hidden from programmers. The programmers' inability to influence synchronization strategies has led to the proposal of a vast number of approaches that consider tailor-made synchronization strategies for particular applications.

We argue that such ambiguity is essential for bidirectional transformation and advocate that the synchronization strategy should not be hidden from programmers but considered from the start. We propose a novel approach to specifying so called well-behaved bidirectional programs by their backward transformations, capable of expressing all aspects of a bidirectional transformation completely, while retaining maintainability.

Soundness of our approach results from a systematic analysis of the laws describing well-behaved bidirectional transformations based on existing mathematical concepts. We show that well-behaved bidirectional transformations are uniquely determined by their backward transformations and corresponding forward transformations can be obtained for free.