Characterizing and representing *neededness* in functional logic languages

Thomas Peikenkamp

October 8, 1992

In functional languages arguments needed for the evaluation are detected by a strictness analysis. Results of this analysis are represented by annotating source programs with evaluation transformers. In functional logic languages the characterisation of needed arguments using only strictness information leads to poor results. This is mainly due to the fact that a failure occuring in the implementation may cause suspended evaluations *not* to be evaluated. On the other side a failure is handled semantically as bottom, which, as usually, indicates that there is not sufficient information available. Unlike functional languages, however, the lack of information cannot be overcome by a deeper evaluation of the arguments, but instead by taking alternative variable bindings into account. Since different variable bindings allow for different degrees of evaluation, it is necessary to provide the "logical analogon" of the evaluator with certain type checks for the variables. Although we show some of these logica l evaluators, the concrete choice of a set of evaluators (e.g. for certain recursive types) is not the topic of this discussion.