Functoriality of monotonicity witnesses in the system of positive (interleaved) inductive types

Ralph Matthes*
Lehr- und Forschungseinheit für theoretische Informatik
Institut für Informatik der Universität München, Germany

As part of the definition of system $\mathsf{M2LJ}$ in [1] (p. 311), for every type ρ and every type variable α which occurs only (not necessarily strictly) positively in ρ, a closed term of type $(\alpha \to \alpha') \to \rho \to \rho[\alpha := \alpha']$ is defined. This term $\mathsf{map}_{\lambda\alpha\rho}$ witnesses monotonicity of ρ in the argument α and is used to define β-reduction for the type $\mu\alpha\rho$, which intuitively represents the least pre-fixed-point of $\sigma \mapsto \rho[\alpha := \sigma]$. For the work to be reported it is essential that already ρ may have been built up by the help of some $\mu\alpha'\rho'$. This may happen by nesting as e.g. the countably-branching trees are defined by the help of the inductive type of the naturals. The more difficult case is given by interleaving μ-types as e.g. $\mu\alpha.1 + (\mathsf{tree}(\alpha) \to \alpha)$ with $\mathsf{tree}(\rho) := \mu\alpha.1 + (\rho \to \alpha)$ (This example is due to Ulrich Berger.) In [2] for the non-interleaved μ-types (more precisely: only for positive type schemes) it is shown that the terms $\mathsf{map}_{\lambda\alpha\rho}$ are functorial, i.e. they preserve identity and composition with respect to β- and η-equality. We also find in [2] the notion of initial $\lambda\alpha\rho$-algebra as opposed to weakly initial algebra which is modelled by β-equality.
We now extend functoriality to any $\mathsf{map}_{\lambda\alpha\rho}$ in the system $\mathsf{M2LJ}$. For this to be true we have to add the canonical η-rule for $\mu\alpha\rho$ but also the conditional equality expressing (full) initiality of $\mu\alpha\rho$, i.e. every term of type $\mu\alpha\rho \to \sigma$ which behaves on constructor terms of type $\mu\alpha\rho$ as some μ-elimination is equal to the function defined by this elimination. This makes the resulting equality a priori undecidable because it gives a formal induction principle. The proof of functoriality obviously also needs a similar statement for the terms $\mathsf{comap}_{\lambda\alpha\rho}$ with α only negatively in ρ which had to be defined simultaneously with the terms $\mathsf{map}_{\lambda\alpha\rho}$. Moreover, we simultaneously have to prove a permutation rule for map and comap (for different indices) which in some sense expresses naturality. The essential case is the interleaved μ. We need all the induction hypotheses due to a lack of permutative conversions for $\mu\alpha\rho$. The idea is to introduce permutative conversions following the well-known example of permutative conversions for sum types which are confluent and strongly normalizing together with β- and η-reduction.
I define equalities in order to permute μ-eliminations with any elimination which follows—including another μ-elimination. Unlike the initiality rule these equalities are not conditional. They do not apply to any μ-elimination followed by another elimination but to sufficiently many so as to ensure functoriality: Functoriality may be proved from these equalities (plus β- and η-equality) without the rule of (full) initiality. On the other hand, they are provably correct in the former theory, and the proof uses initiality, functoriality and the permutation rule which had to be proved simultaneously with functoriality.
I believe that the proposed equations may be turned into rewrite rules which preserve strong normalization and confluence. The resulting decidable equality theory will be a great help in the study of the relation between iteration and (full) primitive recursion on positive inductive types and positive fixed-point types.

References

[1] Daniel Leivant. Contracting proofs to programs. In P. Odifreddi, editor. *Logic in Computer Science*, Academic Press, pp 279–327.

[2] Herman Geuvers. Inductive and coinductive types with iteration and recursion. In Nordström et al, editors. *Preliminary Proceedings Workshop on Types for Proofs and Programs, Båstad, June 1992*, pp 193–217 (`ftp://ftp.cs.chalmers.se/pub/cs-reports/baastad.92/proc.dvi.Z`).

*I am thankful for support by the Volkswagenstiftung.