Erratum sheet for 'Combining Inheritance and Parametric Polymorphism in a Functional Database Language'

Simon Courtenage and Alexandra Poulovassilis

August 7, 1995

In Section 4.2, page 41, there is an error in the typing of a *letrec* clause. That section should instead read as follows:

Typing expressions of the form *letrec* $(x_1, \ldots, x_n) = (\mathrm{E}_1, \ldots, \mathrm{E}_n)$ *in* E is complicated by the need to unify the types of recursive expressions. We first infer types for $\mathrm{E}_1, \ldots, \mathrm{E}_n$. A new coercion set C_{n+1} is then created, containing subtype relations between the type σ_i of each E_i and the type τ_i of its associated recursive variable x_i, as well as between σ_i and a fresh type variable γ_i. The subtype relation $\sigma_i < \tau_i$ reflects the fact that the body of a recursive definition may be substituted for x_i during the calculation of the fix-point, while the relation $\sigma_i < \gamma_i$ reflects the relationship between the recursive definition and its final fix-point. The types of free variables (including the x_i) are unified by unification over the type environments. As usual, unification is followed by matching, to give a substitution S that respects all the coercion sets. We are now in a position to type E with respect to the updated coercion set (including C_{n+1}) and type environment (in which the type of each recursive definition is the type of the fix-point γ_i under the substitution S), obtaining the typing $(\mathrm{C}', \mathrm{A}', \tau)$ below. Finally, as for "let" expressions, we need to unify and match again for free variables occurring in the whole "letrec" expression:

$\mathcal{M}$ (C,A,*letrec* $(x_1, \ldots, x_n) = (\mathrm{E}_1, \ldots, \mathrm{E}_n)$ *in* E)
= let $(\mathrm{C}_i, \mathrm{A}_i, \sigma_i) = \mathcal{M}(\mathrm{C}, \mathrm{A}, \mathrm{E}_i)$ for all $1 \leq i \leq n$
$\tau_i = \tau_i'$, if $x :: \tau_i' \in \mathrm{A}_i$
$= \beta_i$, otherwise for all $1 \leq i \leq n$
$\mathrm{C}_{n+1} = \{\sigma_1 < \tau_1, \ldots, \sigma_n < \tau_n, \sigma_1 < \gamma_1, \ldots, \sigma_n < \gamma_n\}$
$R = \mathrm{UNIFY}(\{\mathrm{A}_1 = \ldots = \mathrm{A}_n)$
$S = \mathrm{MATCH}(R\mathrm{C}_1 \cup \ldots \cup R\mathrm{C}_n \cup R\mathrm{C}_{n+1}) \circ R$
$(\mathrm{C}', \mathrm{A}', \tau) = \mathcal{M}(S \cdot (\mathrm{C} \cup \mathrm{C}_1 \cup \ldots \cup \mathrm{C}_n \cup \mathrm{C}_{n+1}),$
$S\mathrm{A} + \{x_1 :: \forall \bar{\alpha}_1 . S\gamma_1, \ldots, x_n :: \forall \bar{\alpha}_n . S\gamma_n\}, \mathrm{E})$
$T = \mathrm{UNIFY}(\{S\mathrm{A}_1 \cup \ldots \cup S\mathrm{A}_n = \mathrm{A}'\}) \circ S$
$U = \mathrm{MATCH}(T \cdot (\mathrm{C}_1 \cup \ldots \cup \mathrm{C}_n \cup \mathrm{C}_{n+1}) \cup T\mathrm{C}') \circ T$
in
$(U \cdot (\mathrm{C}_1 \cup \ldots \cup \mathrm{C}_n \cup \mathrm{C}_{n+1} \cup \mathrm{C}'), U\mathrm{A}_1 \cup \ldots \cup U\mathrm{A}_n \cup U\mathrm{A}', U\tau)$
where $\bar{\alpha}_i$ is a list of the free variables in $S\gamma_i$ which do not appear in
$S\mathrm{A}_1 \cup \ldots \cup S\mathrm{A}_n$ and $\beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_n$ are fresh type variables