Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Prof. Dr. W.-P. de Roever
Harald Fecher, Martin Steffen

Fool

Sommersemester 2005

Serie 5

May 27 2005

Thema: Subtyping and minimal typing

Ausgabetermin: May 27 2005

Abgabe:

Aufgabe 1 (Typing/subtyping) We had a long discussion to make sense out of the subtyping for object updates, especially the (sub-)typing for the calculus of Section 7 and 8 of [1]. Think of the *typing* rule for update, i.e., the rule from Section 7.2 (p. 81), before we introduced subtyping:

$$\frac{\begin{array}{c} A = \{\ldots l_j : B_j \ldots\} \qquad j \in 1, \ldots, n \\ E \vdash a : A \qquad E, x : A \vdash b : B_j \end{array}}{E \vdash a.l_j{:=}\varsigma(x : A)b : A}\ \textsc{ValUpdate}$$

Would it be ok if we replaced this this by the following rule (which is no longer a pure typing rule):

$$\frac{\begin{array}{c} A = \{\ldots l_j : B_j \ldots\} \qquad j \in 1, \ldots, n \\ E \vdash a : A' \qquad A' \leq A \qquad E, x : A' \vdash b : B_j \end{array}}{E \vdash a.l_j{:=}\varsigma(x : A)b : A'}\ \textsc{ValUpdate}'$$

Before answering that: Think about what criterion you use for "ok-ness".

Aufgabe 2 (Minimal typing) This is a similar exercise. This time consider the rule for update on page 96:

$$\frac{\begin{array}{c} A = \{\ldots l_j : B_j \ldots\} \qquad j \in 1, \ldots, n \\ E \vdash a : A' \qquad \vdash A' \leq A \qquad E, x : A \vdash_{min} b : B'_j \qquad \vdash B'_j \leq B_j \end{array}}{E \vdash_{min} a.l{:=}\varsigma(x : A) : A}\ \textsc{ValMinUpdate}$$

Would it be ok to use the following rule instead? Before answering that: Think again about what criterion you use for "ok-ness". Consider also a possible connection with Exercise 1.

$$\frac{E \vdash a : A' \qquad A = \{\ldots l_j : B_j \ldots\} \qquad j \in 1, \ldots, n \qquad \vdash A' \leq A \qquad E, x : A' \vdash_{min} b : B_j \qquad \vdash B'_j \leq B_j}{E \vdash_{min} a.l{:=}\varsigma(x : A') : A'} \textsc{ValMinUpdate}'$$

Aufgabe 3 (Minimal typing) Let us relax the type system a little. From a practical point of view, static, strong typing is a desirable property to have; sometimes, however, it is considered burdensome to require from the user to explicitly annotate all variables with their types, when introduced. Some users prefer to *leave out* the type annotation (or at least some of the annotation), relying on the type analysis to figure out which type(s) are meant. This is known as *type inference* or *type reconstruction.* Being more relaxed in this respect may also render the system more flexible in the sense that more programs are accepted as well-typed.

So let's do this, i.e., we allow to leave out the explicit mention of the type of a ς-binder, i.e., methods can be written as follows:

$$\{\ldots l = \varsigma(x)b \ldots\}$$

What happens with minimal typing? Is this a good way of making the language more flexible/user-friendly?

Literatur

[1] Martín Abadi and Luca Cardelli. *A Theory of Objects.* Monographs in Computer Science. Springer, 1996.