

Prof. Dr. Jürgen Giesl
René Thiemann

Exercises *Functional Programming* – Sheet 14

Solutions will be collected until Thursday, Feb 6, 2003 in the exercise course.

On the whole sheet we speak of well-typed terms iff a term has a *shallow* type schema. In Exercise 1 and 2 we use the following type assumption A :

$$\begin{aligned} A(1) &= Int \\ A(True) &= Bool \\ A(+) &= Int \rightarrow Int \rightarrow Int \\ A(Nil) &= \forall a. List\ a \\ A(Cons) &= \forall a. a \rightarrow List\ a \rightarrow List\ a \\ A(tuple_3) &= \forall a_1, a_2, a_3. a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow (a_1, a_2, a_3) \end{aligned}$$

Exercise 1 (4 points)

Give the most general type-schema for each well-typed term. For the remaining terms you should say why they can not be typed.

- $\lambda x\ y. Cons\ x\ (Cons\ x\ y)$
- $\lambda x. Cons\ x\ (Cons\ 1\ x)$
- $\lambda x\ y. (y\ x) + (y\ Nil) + (x\ y)$
- $\lambda x\ y. (y\ Nil) + (y\ (Cons\ x\ Nil)) + (x\ Nil)$

Exercise 2 (7.5 points)

For each of the following terms try to extend the type assumption A for the free variable x such that these terms are well-typed. Is this possible for each term? For all terms t where you found an extension determine the most general type schema for t using this extended type assumption.

- $Cons\ x\ x$
- $tuple_3\ (x\ 1)\ (x\ True)\ (x\ Nil)$
- $tuple_3\ (x\ 1)\ (x\ True)\ x$
- $tuple_3\ (x\ 1)\ (x\ True)$
- $x\ x$

Exercise 3 (3+3.5 points)

In the translations from Haskell to simple Haskell, we used twelve transformation rules.

- (a) We now replace the transformation rule (5) by the following one:

$$\frac{\text{match } \underline{\text{var}} \ \underline{\text{exp}} \ \underline{\text{exp}}_1 \ \underline{\text{exp}}_2}{\underline{\text{exp}}_1 \ [\underline{\text{var}}/\underline{\text{exp}}]}$$

Show that there exists a Haskell-expression such that this change leads to a well-typed lambda term, whereas the original transformation rules produce a non well-typed term.

- (b) In this part, we leave rule (5) unchanged but do drop rule (10) and modify rule (11) to the following rule:

$$\frac{\{\underline{\text{var}}_1 = \underline{\text{exp}}_1; \ \dots \ ; \ \underline{\text{var}}_n = \underline{\text{exp}}_n\}}{(\underline{\text{var}}_1, \dots, \underline{\text{var}}_n) = (\underline{\text{exp}}_1, \dots, \underline{\text{exp}}_n)}$$

where $n \geq 2$, $\underline{\text{var}}_i \neq \underline{\text{var}}_j$ for $i \neq j$.

Show that this transformation does not preserve type correctness. So, you have to find an expression such that the original transformation rules produces a well-typed term, but this modified transformation does not.

Hints:

- To get a counterexample you can extend one of the examples of Exercise 2.
- Find and use the difference between the term $(\lambda x.t) \ t'$ and the term $t[x/t']$.