

Prof. Dr. Jürgen Giesl  
René Thiemann

## Exercises *Functional Programming* – Sheet 10

Solutions will be collected until Thursday, Jan 9, 2003 in the exercise course.

### Exercise 1 ( 1.5 + 2.5 + 2.5 + 2.5 points)

Extend the set of transformation rules with additional rules which transform Haskell-expressions with

- where declarations
- conditional equations (usage of |)
- patterns using @
- patterns using  $x+n$ ,  $x$  is a variable,  $n$  is an integer greater than 0

to simple Haskell-expressions.

*Remark:* The pattern `var@pat` matches if `pat` matches and binds the whole value of `pat` to the variable `var`.

Example:

```
tails :: [a] -> [[a]]
tails [] = [[]]
tails xxs@(_: xs) = xxs: tails xs
```

`tails [1,2]` results in `[ [1,2], [2], [] ]`

### Exercise 2 (4 points)

The data structure `List` is defined by

```
data List a = Nil | Cons a (List a)
```

Transform the following Haskell-expression to a simple Haskell-expression using the transformation rules from the lecture.

```
let rev l = rev' l Nil
    rev' Nil ys      = ys
    rev' (Cons x xs) ys = rev' xs (Cons x ys)
in rev (Cons 1 (Cons 2 Nil))
```

You can simplify your results during the transformation like it was done in the example in the lecture, so

- $(\backslash \text{var} \rightarrow \text{exp}) \text{ exp}'$  can be replaced by  $\text{exp}$ , where all occurrences of  $\text{var}$  are replaced by  $\text{exp}'$
- $\text{sel}_{n,i} (x_1, \dots, x_n)$  can be replaced by  $x_i$
- $\text{if } (\text{isa}() \text{ exp}) \text{ then } \text{exp1} \text{ else } \text{exp2}$  can be replaced by  $\text{exp1}$
- ...

The last simplification rule given above is only valid for type-correct expressions!

**Merry Christmas and a Happy New Year!**