universität innsbruck

Advanced Functional Programming | WS 2025/2026 | LVA 703139

Exercise Sheet 2, 10 points | Deadline: Tuesday, October 21, 2025, 4pm

- Solve the tasks in file `Exercise02.hs` and upload only this file in OLAT.
- Mark the solved exercises in OLAT.
- Your modified `Exercise02.hs` file must compile with `ghci` without error messages.

Task 1 *Type Inference* **5 p.**

Consider the following definition of a `fold` function on lists:

```
fold [] e f = e
fold (x : xs) e f = f x (fold xs e f)
```

1. Construct constraints to determine the most generic type of `fold`, similarly to slide 12 of week 2.
2. Encode the constraints in Haskell, and use the provided implementation of the unification algorithm to obtain the most generic type of `fold`. Compare the computed type to the type-inference algorithm of ghc. The latter can be invoked as follows:

   ```
   cabal repl
   ghci> :m Exercise02
   ghci> :t fold
   ```

Task 2 *Type Inference with Let and λ* **5 p.**

1. Extend the type-inference algorithm so that it can handle λ-abstractions of the form `\ x -> e`, where `x` is a variable and `e` some expression. What will be the constraints for type-inference of `function`?

   ```
   function = \ x -> x x
   ```

2. Compare the difference of type-inference of Haskell when treating λ and `let`. To this end, invoke ghc on the following two functions. Try to explain the observed difference.

   ```
   polymorphicLet :: (Bool, String)
   polymorphicLet =
     let f = id
     in (f True, f "hello")

   polymorphicLambda :: (Bool, String)
   polymorphicLambda =
     (\ f -> (f True, f "hello")) id
   ```