Functional Programming
WS 2007/08

Christian Sternagel[1] (VO + PS)
Friedrich Neurauter[2] (PS)
Harald Zankl[3] (PS)

Computational Logic
Institute of Computer Science
University of Innsbruck

18 January 2008

[1] christian.sternagel@uibk.ac.at
[2] friedrich.neurauter@uibk.ac.at
[3] harald.zankl@uibk.ac.at

Overview

Overview

Type Checking

- prove that some expression really has a given type w.r.t. an environment
- formally: $E \vdash e : \tau$
- use the inference rules of $\mathcal{C}$ to do so

Type Inference

- get the most general type for an expression w.r.t. an environment
- formally: $E \rhd e : \tau$
- task is split into two parts:
  1. transform given type inference problem into a unification problem
  2. solve the unification problem (result is substitution)

Overview

Lazyness in OCaml

Keyword **lazy**

used to transform arbitrary expression into lazy expression

Example

- **let** e0 = **lazy** (Format.printf "test\n");;
- **let** e1 = **lazy** (**let rec** f x = print_int x; f (x + 1) **in** f 0)

Function Lazy.force

used to evaluate lazy expressions

Example

- Lazy.force e0;;
- Lazy.force e1;;

Example - Lazy Lists

Live Demonstration