

Functional Programming

WS 2007/08

Christian Sternagel¹ (VO + PS)
Friedrich Neurauter² (PS)
Harald Zankl³ (PS)

Computational Logic
Institute of Computer Science
University of Innsbruck

18 January 2008

¹christian.sternagel@uibk.ac.at

²friedrich.neurauter@uibk.ac.at

³harald.zankl@uibk.ac.at

Overview

Week 11 - Laziness

Summary of Week 10

Lazyness

Overview

Week 11 - Laziness

Summary of Week 10

Lazyness

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: $E \triangleright e : \tau$
- ▶ 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

Week 11 - Laziness

Summary of Week 10

Lazyness

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