National Technical University of Athens
School of Electrical & Computer Engineering
Department of Computer Science
https://courses.softlab.ntua.gr/pl2/

Programming Languages II

Unless otherwise stated, exercises should be submitted in electronic form, via the collaborative learning environment `moodle.softlab.ntua.gr`. Deadlines will be strict. You are allowed at most one late exercise.

Exercise 5 Type inference

Deadline: 2/2/2020

Implement type inference for the λ-calculus with simple types. Submit your solution to the automatic grading system `grader.softlab.ntua.gr`, in any programming language that you want (among those supported by the `grader` — if you want a different one, talk to the instructor). As a piece of advice, we recommend that you pick a functional language: Haskell, Standard ML (MLton) or OCaml.

Input and output. Your program will read from the standard input (stdin) and print the results to the standard output (stdout).

The first line of the input will contain a natural number N. Each of the following N lines will contain one λ-term. The precise form in which the λ-terms are given will be explained in the task description that you will find at the `grader`.

Your program must output N lines, each of which will contain the type that was inferred for the corresponding λ-term or the message "type error", in case the type inference was not succesful. The precise form in which the types must be printed will be explained in the task description that you will find at the `grader`.

Brief description of the solution. The following grammar defines the syntax of λ-terms (M, N) and the syntax of types (σ, τ).

$$M, N ::= x \mid (\lambda x.M) \mid (M\ N)$$
$$\sigma, \tau ::= \alpha \mid (\sigma \to \tau)$$

To construct the set of constraints that results from the semantic analysis of a λ-term, you may use the following rules. They are the same rules that were written on the blackboard, at the lecture of 11/12/2019, and they are equivalent to the typing rules of Curry-style simply-typed λ-calculus, given on slide 41 of the same lecture.

The typing relation $\Gamma \vdash e : \tau \mid C$ denotes that in context Γ, expression e has type τ, under the assumption that the constraints found in set C are satisfied.

$$\frac{(x : \tau) \in \Gamma}{\Gamma \vdash x : \tau \mid \varnothing} \qquad \frac{\alpha \text{ is a fresh type variable} \quad \Gamma, x : \alpha \vdash e : \tau \mid C}{\Gamma \vdash \lambda x.\, e : \alpha \to \tau \mid C}$$

$$\frac{\Gamma \vdash e_1 : \sigma \mid C_1 \quad \Gamma \vdash e_2 : \tau \mid C_2 \quad \alpha \text{ is a fresh type variable}}{\Gamma \vdash e_1\, e_2 : \alpha \mid C_1 \cup C_2 \cup \{\sigma = \tau \to \alpha\}}$$

Your function that will implement the above rules will take as input Γ and e and produce as output τ and C. Algorithm W can be used to compute the most general unifier for C. It is given on slide 42.