Grading criteria for languages and compilers course

Dr. Giuseppe Maggiore

April 28, 2015

1 Introduction

In this document the

2 Grading criteria

...

2.1 General grading criteria

Purity, correctness, compilation, runtime errors.

Idiomatic style such as seen in the lectures.

2.2 Assignment 3

Assignment 3: Complete the definition of the typeCheck function so that it checks the type relations of terms.

Note: for bonus points you can make this function correctly handle higher-order-functions and generic types.

Sample solution ...

```
let typeCheck (p:Term) : Option<Type> =
  let rec typeCheck (p:Term) (env:Map<string,Type>) : Option<Type> =
    match p with
    | Var s -> env |> Map.tryFind s
    | Term.Int _ -> Some Type.Int
    | Term.Bool _ -> Some Type.Bool
    | Application(f,a) ->
      match typeCheck f env,typeCheck a env with
      | Some(Fun(fI,fO)),Some(a) when a = fI -> // TODO: generic instantiation should happen here
        Some fO
      | _ -> failwithf "Invalid function invocation %A" p
    | FunctionDef(arg,body) ->
      match typeCheck body (env |> Map.add arg.Name arg.Type) with
      | Some bT -> Some(Type.Fun(arg.Type, bT))
      | _ -> failwithf "Invalid function definition %A" p
    | Plus(a,b) ->
      match typeCheck a env,typeCheck b env with
      | Some(Int),Some(Int) -> Some Int
      | _ -> failwithf "Invalid sum %A" p
    | GreaterThan(a,b) ->
      match typeCheck a env,typeCheck b env with
      | Some(Int),Some(Int) -> Some Bool
      | _ -> failwithf "Invalid sum %A" p
  typeCheck p Map.empty
```

Specific criteria Answer model for this assignment.

This assignment requires building a function that exhibits the following characteristics in order to pass the assignment: - the function is recursive - the function is referentially transparent - the function should handle types of functions such as int -¿ int - the function should handle primitive types such as

int or bool if the language supports them - the function should represent information about variable bindings in the form of some mapping container - the input of this container are variable identifiers (int's or strings) - the output of this container are values or terms - in the lambda calculus values are terms (thus there is no separate definition) - (advanced) the function should handle generic type variables and generic instantiation of terms - with the same environment as that for variables - (advanced) Option¡'a¿ bindings are handled with a monad

The assignment is passed if at least five of the above-criteria are handled.