Typed Language

Interpretation and Compilation
15-NOV-2018

Luis Caires

Concrete Syntax (Typed Language)

Ty -> **int**	ASTIntType()
| **bool**	ASTIntType()
| **ref** Ty	ASTIntType(Ty)
| (Ty,...,Ty)Ty	ASTFunType(List<Ty>,Ty)

Concrete Syntax (Typed Language)

```
EM -> E(<;>EM)*                            ASTSeq(E1,E2)
E -> EA(< == > EA)?                        ASTEq(EA,EA)
EA -> T(<+>EA)*                            ASTAdd(E1,E2)
T -> F ( (<*>T)*                           ASTMul(F,T)
          | (<(>AL<)>)*                    ASTApply(F,AL)
          | <:=> E)                        ASTAssign(F,E)
AL -> (EM(<,>EM)*)?
PL -> (id:Type(<,>id:Type)*)?
F -> num | id | bool | let (id : Type = EM)+ in EM end
   | fun PL -> EM end | <(> EM <)>
   | new F | <!> F
   | if EM then EM else EM end                ASTIf(EM,EM,EM)
   | while EM do EM end                       ASTWhile(EM,EM)
```

Goal

Implement a complete type checker for the basic imperative-functional language specified

Use the approach developed in the lectures

- extend parser to support type declarations
- AST model for types
- Environment based typechecker
- Integrate with your interpreter, before running the program, typecheck it!

Fully understanding the handout statement is part of the handout as well. Contact me if you need help.

Examples

```
(new 3) := 6;;

let a : int = new 5 in a := !a + 1; !a end;;

let x : int = new 10
     s : int = new 0  in
while !x>0 do
     s := !s + !x ; x := !x – 1
end; !s
end;;
```

Examples

```
let  f :( int,int)int = fun n:int, b:int->
        let
         x : int = new n
         s : int = new b
        in
          while !x>0 do
            s := !s + !x ; x := !x – 1
          end;
          !s
        end
      end
in f(10,0)+f(100,20)
end;;
```