

Handout 1

Interpretation and Compilation

23-OCT-2018

due 2-NOV-2018

Luis Caires

Goal

Implement a complete interpreter for the basic imperative-functional language specified

Use the approach developed in the lectures

- LL(1) parser using JAVACC
- AST model
- Environment based evaluator
- Dynamic type checking

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

Abstract Syntax

EE \rightarrow EE ; EE | EE := EE
| num | id | bool | let (id = EE)+ in EE end
| fun id* \rightarrow EE end
| EE (EE*)
| new EE | <!> EE
| if EE then EE else EE end
| while EE do EE end
| EE binop EE
| unop EE

Concrete Syntax

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

Basic operations

Arithmetic operations (on integer values)

$E + E$, $E - E$, $E * E$, E / E , $-E$

Relational operations

$E == E$, $E > E$, $E < E$, $E <= E$, $E >= E$

Logical operations (on boolean values)

$E \&\& E$, $E || E$, $\sim E$

AST(schematic)

```
interface ASTNode {  
  IValue eval(Environment env) ...  
}
```

```
class AST??? implements ASTNode {  
  
}
```

IValues (schematic)

```
interface IValue {  
  void show();  
}
```

//Value constructors

VInt(n)

Closure(args,body,env)

VBool(t)

VCell(value)

IValues (schematic)

```
class VInt implements IValue {  
    int v;  
    VInt(int v0) { v = v0; }  
    int getval() { return v;}  
}
```


IValues (schematic)

```
class VCell implements IValue {  
    IValue v;  
    VCell(IValue v0) { v = v0; }  
    IValue get() { return v;}  
    void set(IValue v0) { v = v0;}  
}
```

IValues (schematic)

```
class ASTAdd implements ASTNode {
```

```
    IValue eval(Environmment env) {
```

```
        v1 = left.eval(env);
```

```
        if (v1 instanceof VInt) {
```

```
            v2 = right.eval(env)
```

```
            if (v2 instanceof VInt) {
```

```
                return new VInt((VInt)v1).getval()+((VInt)v2).getval())
```

```
            }
```

```
        throw TypeError("illegal arguments to + operator");
```

```
    }
```

Examples

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

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

```
let x = new 10
```

```
    s = new 0 in
```

```
while !x>0 do
```

```
    s := !s + !x ; x := !x - 1
```

```
end; !s
```

```
end;;
```

Examples

```
let f = fun n, b->
  let
    x = new n
    s = 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;;
```