Interpretation and Compilation

TEST 2A

Luis Caires

Universidade Nova de Lisboa

2014

Test Statement

The goal of the exercise is to discuss how to extend your interpreter / compiler with an additional `for` construct (described in the last page).

PART I

This part of the test is about compiler design and implementation.

You will be required to add the `for` construct to your current interpreter code. For this you need to extend your LL(1) grammar and parser, define the additional AST node class(es), and implement the evaluation and typechecking methods

IValue eval(Environment e)

IType typecheck(TyEnvironment e)

Test Statement

The goal of the exercise is to discuss how to extend your interpreter / compiler with an additional `for` construct (described in the last page).

PART 2

This part of the test must be answered in a separate text file “answer.txt”, that you should add to the answer tar file.

Q1:

Write the typing rule for the `for` construct.

Q2:

Define a compilation scheme for the `for` construct, targeting the JVM.

$[\![\ \texttt{for id=}e_1\ \texttt{to}\ e_2\ \texttt{step}\ e_3\ \texttt{do}\ e_4\ \texttt{end}\ ]\!]_E = ?$

for construct

1 - Concrete syntax of the `for` construct

```
for id=e1 to e2 step e3 do e4 end
```

2 - Semantics of the `for` construct (informal).

The identifier `id` is locally declared with scope the loop body e_4 and represents an **imutable** integer value.

The expression e_1 is evaluated and its integer value bound to `id`.

Then, the body e_4 is executed for all integer values between the initial value of e_1 and the initial value of e_2, separated by the initial value of e_3 (e.g., e_1, e_2 and e_3 are evaluated only once, initially). This means that at the end of each execution of e_4 the step value e_3 is added to `id`.

The loop terminates as soon as `id` value falls out the interval $[e_1,e_2]$.

Example 1

```
for i = 0 to 20 step 1 do
  println i
end;;
```

Example 1

```
decl
   s = new 0
in
   for i = 0 to 20 step 2 do
      s := i + !s
   end;
   println !s
end;;
```

Example 2

```
decl
   s = new 0
in
   for i = !s to -10 step !s-1 do
      s := i + !s
   end;
   println !s
end;;
```