

CS320 Programming Languages

Homework #9: COREL

Due: 29 May 2019

The goal of Homework #9 is to implement the interpreter of **COREL**, which is the **CORE** Language with types. Implement its type checker and the interpreter:

```
<COREL> ::= <num>
| true
| false
| {+ <COREL> <COREL>}
| {- <COREL> <COREL>}
| {= <COREL> <COREL>}
| {with {<id> : <Type> <COREL>} <COREL>}
| <id>
| {fun {<id> : <Type>} <COREL>}
| {<COREL> <COREL>}
| {if <COREL> <COREL> <COREL>}
| {recfun {<id> : <Type> <id> : <Type>} <COREL>}
| {withtype {<id> {<id> <Type>}+} <COREL>}
| {cases <id> <COREL> {<id> {<id>} <COREL>}+}

<Type> ::= num
| bool
| {<Type> -> <Type>}
| <id>
```

It provides number literals with three binary operations; addition, subtraction, and equality. The equality operation is only for numbers thus it should throw an error when either value of left or right hand side is not number. The **with** expressions, identifiers, **fun** expressions, and function applications have semantics that you learned in the lecture:

```
test(run("42"), "42")
test(run("true"), "true")
test(run("{+ 1 2}"), "3")
test(run("{- 2 1}"), "1")
test(run("{= 1 0}"), "false")
testExc(run("{= true 0}"), "no type")
test(run("{with {x : num 1} x}"), "1")
test(run("{fun {x : num} {+ x 1} 2}"), "3")
```

The value of the first expression of **if-then-else** expression should be evaluated into the boolean value and if it is **true**, evaluate the second expression, otherwise, evaluate the third one. The **recfun** form denotes recursive function:

```
test(run("""
  {recfun {f: {num -> num} x: num}
    {if {= x 0} 0 {+ {f {- x 1}} x}}
  10}"""), "55")
testExc(run("{if 1 2 3}"), "no type")
```

In this homework, the **withtype** and **cases** are more general version than in the lecture. It accepts one or more constructors in **withtype** thus **cases** also have one or more cases. With such generality, you should care about the mismatch between **withtype** and **cases**. If **cases** does not exactly cover the all the cases of the corresponding **withtype**, **typeCheck** should throw an error with a message containing "not all cases":

```
test(run("""
  {withtype
    {fruit {apple num}
      {banana num}}
    {cases fruit {apple 1}
      {apple {x} x}
      {banana {y} y}}}"""), "1")
testExc(run("""
  {withtype
    {fruit {apple num}
      {banana num}}
    {cases fruit {apple 1}
      {apple {x} x}}}"""), "not all cases")
```