SORBONNE UNIVERSITÉ

TD : Compilateur ml2java — semaine 3

22 février 2018

Objectif(s)

✯ Manipulation d'un traducteur de code ML vers Java.

1 ML2Java

Exercice 1 – Structure du runtime

1. Déterminer la hiérarchie de classes correspondant à la représentation des données choisie.
2. Construire directement en Java les données suivantes :
   — `(37.2, true)`
   — `''un''::''jour''::[]`
   — `''un''::2::[]`
   — `ref 3`
   — `()`

Exercice 2 – Ajout des tableaux

1. Ajouter un type tableau à cette bibliothèque pour implanter des tableaux à la ML avec les constructeurs et accesseurs habituels.

Exercice 3 – Code généré

1. Chaque fonction ML sera traduite par la classe `MLFun`.
   — En reprenant le code engendré pour la fonction `fib`, simplifiez le à la main (voir Annexe 3).
   — Soit la fonction `map` suivante :

```
let rec map = function f -> function l ->
    if l =[] then []
    else (f (hd l))::: (map f (tl l));;
```

   Écrivez à la main le code produit par le compilateur suivant les schémas de compilation indiqués dans le cours.
2. Que construit l'appel de `map fib` ?
3. Que construit l'appel `map fib [1;2;3]` ?

2 Annexe : code du traducteur

Listing 2 – runtime.java

```
// compile with: javac runtime.java ftest.java

import java.util.ArrayList;

abstract class MLvalue extends Object {

  abstract void  print();
}

class MLunit extends MLvalue
{
  private int val;

  MLunit(){val=0;}

  public void print(){System.out.print("()");}
  public int MLaccess(){return val;}
}


class MLbool extends MLvalue
{
  private boolean val;

  MLbool(boolean a){val=a;}

  public void print(){if (val) System.out.print("true");
                      else System.out.print("false");}
  public boolean MLaccess(){return val;}
}

class MLint extends MLvalue
{
   private int val;
   MLint(int a){val=a;}


   public void print(){System.out.print(val);}
   public int MLaccess(){return val;}
}

class MLdouble extends MLvalue
{
   private double val;
   MLdouble(double a){val=a;}

   public void print(){System.out.print(val);}

   public double MLaccess(){return val;}
}

class MLstring extends MLvalue
{
  private String val;
  MLstring(String a){val=a;}
```

```
  public void print(){System.out.print("\""+val+"\"");}
  public String MLaccess(){return val;}
}

class MLpair extends MLvalue
{
  private MLvalue MLfst;
  private MLvalue MLsnd;

  MLpair(MLvalue a, MLvalue b){MLfst=a; MLsnd=b;}

  public MLvalue MLaccess1(){return MLfst;}
  public MLvalue MLaccess2(){return MLsnd;}
  public void print(){System.out.print("(");
                      MLfst.print();
                      System.out.print(",");
                      MLsnd.print();
                      System.out.print(")");}

}

class MLlist extends MLvalue
{
  private MLvalue MLcar;
  private MLlist  MLcdr;

  MLlist(MLvalue a, MLvalue b){MLcar=a; MLcdr=(MLlist)b;}
  MLlist(MLvalue a, MLlist b){MLcar=a; MLcdr=b;}

  public MLvalue MLaccess1(){return MLcar;}
  public MLlist  MLaccess2(){return MLcdr;}
  public void print(){if (MLcar == null) {System.out.print("[]");}
                      else {MLcar.print();
                            System.out.print("::");
                            MLcdr.print();}}

}

abstract class MLfun extends MLvalue
{
  public int MLcounter;
  protected MLvalue[] MLenv;

  MLfun(){MLcounter=0;}
  MLfun(int n){MLcounter=0;MLenv = new MLvalue[n];}

  public  void  MLaddenv(MLvalue []O_env,MLvalue a)
   { for (int i=0; i< MLcounter; i++) {MLenv[i]=O_env[i];}
     MLenv[MLcounter]=a;MLcounter++;}

 abstract public MLvalue invoke(MLvalue x);

  public void print(){
     System.out.print("<fun>␣[");
     for (int i=0; i< MLcounter; i++)
           MLenv[i].print();
     System.out.print("]");
  }

}
```

```
class MLprimitive extends MLfun {

  String name="";

  MLprimitive(String n){name=n;}

  public MLvalue invoke(MLvalue l) {
    if (name.equals("hd")) return MLruntime.MLhd_real((MLlist)l);
    else if (name.equals("tl")) return MLruntime.MLtl_real((MLlist)l);
    else if (name.equals("fst")) return MLruntime.MLfst_real((MLpair)l);
    else if (name.equals("snd")) return MLruntime.MLsnd_real((MLpair)l);
    else {System.err.println("Unknown primitive "+name); return l;}
  }
}


class MLruntime {

// booleens
  public static MLbool MLtrue = new MLbool(true);
  public static MLbool MLfalse = new MLbool(false);
// unit
  public static MLunit MLlrp = new MLunit();
// nil
  public static MLlist MLnil = new MLlist(null,null);

// arithmetique sur les entiers
  public static MLint MLaddint(MLint x, MLint y) {
       return new MLint(x.MLaccess()+y.MLaccess());
  }
  public static MLint MLsubint(MLint x, MLint y) {
       return new MLint(x.MLaccess()-y.MLaccess());
  }
  public static MLint MLmulint(MLint x, MLint y) {
       return new MLint(x.MLaccess()*y.MLaccess());
  }
  public static MLint MLdivint(MLint x, MLint y) {
       return new MLint(x.MLaccess()/y.MLaccess());
  }
// fonction equal
  public static MLbool MLequal(MLvalue x, MLvalue y) {
       return new MLbool((x == y) || (x.equals(y)));
  }
// inegalites sur les entiers
  public static MLbool MLltint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()<y.MLaccess());
  }

  public static MLbool MLleint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()<=y.MLaccess());
  }

  public static MLbool MLgtint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()>y.MLaccess());
  }

  public static MLbool MLgeint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()>=y.MLaccess());
  }
```

```
// paire
  public static MLpair MLpair(MLvalue x, MLvalue y) {
        return new MLpair(x,y);
  }

// liste
  public static MLlist MLlist(MLvalue x, MLvalue y) {
        return new MLlist(x,y);
  }

// string
  public static MLvalue MLconcat(MLstring x, MLstring y) {
        return new MLstring(x.MLaccess()+y.MLaccess());
  }

// acces aux champs des paires
  public static MLvalue MLfst = new MLprimitive("fst");
  public static MLvalue MLfst_real(MLpair p) {
      return p.MLaccess1();
  }

  public static MLvalue MLsnd = new MLprimitive("snd");
  public static MLvalue MLsnd_real(MLpair p) {
      return p.MLaccess2();
  }

// acces aux champs des listes
  public static MLvalue MLhd = new MLprimitive("hd");
  public static MLvalue MLhd_real(MLlist l) {
      return l.MLaccess1();
  }

  public static MLvalue MLtl = new MLprimitive("tl");
  public static MLvalue MLtl_real(MLlist l) {
      return l.MLaccess2();
  }

// la fonction d'affichage
  public static MLvalue MLprint(MLvalue x) {
     x.print();
     System.out.println();
     return MLlrp;}

}
```

3 Annexe : Génération d'une fonction (Fibonacci)

Listing 3 – fib.ml

```
let rec fib = function x -> if x < 3 then 1 else (fib(x-1))+(fib(x-2));;
```

Listing 4 – fibtrad.java

```
/**
 *  fib.java engendre par ml2java
 */


/**
 *  de'claration de la fonction fib___1
```

```
 *    vue comme la classe : MLfun_fib___1
 */
class MLfun_fib___1 extends MLfun {

  private static int MAX = 1;

  MLfun_fib___1() {super();}

  MLfun_fib___1(int n) {super(n);}

  public MLvalue invoke(MLvalue MLparam){
    if (MLcounter == (MAX-1)) {
      return invoke_real(MLparam);
    }
    else {
      MLfun_fib___1 l = new MLfun_fib___1(MLcounter+1);l.MLaddenv(MLenv,MLparam);
      return l;
    }
  }


  MLvalue invoke_real(MLvalue x___2) {

    {
      MLvalue T___3;
      {
        MLvalue T___4;
        MLvalue T___5;
        T___4=x___2;
        T___5=new MLint(3);
        T___3=MLruntime.MLltint( (MLint )T___4,(MLint )T___5);
      }
      if (((MLbool)T___3).MLaccess())
        {
          MLvalue T___6;
          T___6=new MLint(1);
          return T___6;
        }
      else
        {
          MLvalue T___7;
          {
            MLvalue T___8;
            MLvalue T___13;
            {
              MLvalue T___9;
              MLvalue T___10;
              T___9=fib.fib___1;
              {
                MLvalue T___11;
                MLvalue T___12;
                T___11=x___2;
                T___12=new MLint(1);
                T___10=MLruntime.MLsubint( (MLint )T___11,(MLint )T___12);
              }
              T___8=((MLfun)T___9).invoke(T___10);
            }
            {
              MLvalue T___14;
              MLvalue T___15;
```

```
              T___14=fib.fib___1;
              {
                MLvalue T___16;
                MLvalue T___17;
                T___16=x___2;
                T___17=new MLint(2);
                T___15=MLruntime.MLsubint( (MLint )T___16,(MLint )T___17);
              }
              T___13=((MLfun)T___14).invoke(T___15);
            }
            T___7=MLruntime.MLaddint( (MLint )T___8,(MLint )T___13);
          }
          return T___7;
        }
    }
  }

}
// fin de la classe MLfun_fib___1
/**
 *
 */
class fib {

  static MLvalue fib___1= new MLfun_fib___1(1);
  static MLvalue value___18;

public static void main(String []args) {

{
  MLvalue T___19;
  MLvalue T___20;
  T___19=fib.fib___1;
  T___20=new MLint(3);
  value___18=((MLfun)T___19).invoke(T___20);
}
{
  MLvalue bidon___21;
  bidon___21=MLruntime.MLlrp;
  bidon___21=MLruntime.MLprint( (MLvalue )value___18);
}
}}

// fin du fichier fib.java
```