Challenge Novembre – 2015

Monkey Business

A solution with OPL CPLEX by Alex Fleischer
afleischer@fr.ibm.com

OPL (Optimization Programming Language) is an abstract modeling language that helps model easily optimization problems that can be solved both with IBM CPLEX linear programming and IBM CPLEX constraint programming CPOptimizer (CPO)

Optimization can help any kind of business, which includes for sure Monkey Business.

With Constraint Programming within OPL CPLEX we can write

```
/*

Sam, who doesn't like bananas, likes sitting on the grass
The monkey who sat on the rock ate the apple. The monkey who ate the pear didn't
sit on the tree branch
Anna sat by the stream but she didn't eat the pear
Harriet didn't sit on the tree branch. Mike doesn't like oranges.

*/

using CP;

{string} monkeys={"Anna","Harriet","Mike","Sam"};
{string} fruits={"apple","banana","pear","orange"};
{string} spots={"grass","rock","stream","branch"};

int monkeysrank[m in monkeys]=ord(monkeys,m)+1;
int fruitsrank[f in fruits]=ord(fruits,f)+1;
int spotsrank[s in spots]=ord(spots,s)+1;

range rmonkeys=1..card(monkeys);
dvar int what[rmonkeys] in 1..card(fruits);
dvar int where[rmonkeys] in 1..card(spots);

dvar int whoeats[1..card(fruits)] in rmonkeys;
dvar int whosits[1..card(spots)] in rmonkeys;

subject to
{
  allDifferent(what);
  allDifferent(where);
  inverse(whoeats,what);
  inverse(whosits,where);


  what[monkeysrank["Sam"]]!=fruitsrank["banana"];
```

```
  where[monkeysrank["Sam"]]==spotsrank["grass"];
  whoeats[fruitsrank["apple"]]==whosits[spotsrank["rock"]];
  whoeats[fruitsrank["pear"]]!=whosits[spotsrank["branch"]];
  where[monkeysrank["Anna"]]==spotsrank["stream"];
  what[monkeysrank["Anna"]]!=fruitsrank["pear"];
  where[monkeysrank["Harriet"]]!=spotsrank["branch"];
  what[monkeysrank["Mike"]]!=fruitsrank["orange"];
}

execute
{
  for(var r in rmonkeys)
  {
  var m=Opl.item(monkeys,r-1);
  writeln(Opl.item(monkeys,r-1)," sits ",Opl.item(spots,where[r]-1)," and eats
",Opl.item(fruits,what[r]-1));
  }
}

/*

which gives

Anna sits stream and eats orange
Harriet sits rock and eats apple
Mike sits branch and eats banana
Sam sits grass and eats pear

*/
```

Whereas with Mathematical Programming we can write the following which works both with MIP and CP:

```
/*

Sam, who doesn’t like bananas, likes sitting on the grass
The monkey who sat on the rock ate the apple. The monkey who ate the pear didn’t
sit on the tree branch
Anna sat by the stream but she didn’t eat the pear
Harriet didn’t sit on the tree branch. Mike doesn’t like oranges.

*/

//using CP; // CP works fine too with this model, just uncomment this line

{string} monkeys={"Anna","Harriet","Mike","Sam"};
{string} fruits={"apple","banana","pear","orange"};
{string} spots={"grass","rock","stream","branch"};

dvar boolean where[monkeys][spots];
dvar boolean what[monkeys][fruits];

subject to
{
  // monkeys are at a given spot
  forall(m in monkeys) sum(s in spots) where[m][s]==1;
```

```
  // which are different
  forall(s in spots) sum(m in monkeys) where[m][s]==1;

  // monkeys eat a given fuit
  forall(m in monkeys) sum(f in fruits) what[m][f]==1;
  // which are different
  forall(f in fruits) sum(m in monkeys) what[m][f]==1;

  what["Sam"]["banana"]==false;
  where["Sam"]["grass"]==true;
  forall(m in monkeys)
    {
      what[m]["apple"]==where[m]["rock"];
      what[m]["pear"]+where[m]["branch"]<=1;
    }

  where["Anna"]["stream"]==true;
  what["Anna"]["pear"]==false;

  where["Harriet"]["branch"]==false;
  what["Mike"]["orange"]==false;
}

execute
{
  for(var m in monkeys)
  {
    write(m," sits ");
    for(var s in spots) if (where[m][s]==1) write(s);
    write(" and eats ");
    for(var f in fruits) if (what[m][f]==1) writeln(f);
   }
}

/*

which gives

Anna sits stream and eats orange
Harriet sits rock and eats apple
Mike sits branch and eats banana
Sam sits grass and eats pear

*/
```

NB :

Monkey Business is a very good movie by the Marx Brothers. (Not to be confused with Karl Marx)