```
(************************************************************************
************************************************************************
********
******** Concrete Categories, definition of Set
******** Language SML/NY
******** Author M. Bertol (c)
********
******** Release: Tue Aug 13 11:35:40 MET DST 1996                                10
********
************************************************************************
************************************************************************)

use"cat.sml";

use"simpleSet.sml";


                                                                                20
exception nonCompatiblePair;

datatype ('a)Set_Arrow = set_arrow of ('a Set) * ('a -> 'a) * ('a Set);          set_arrow

fun set_s(set_arrow(a,_,_)) = a;                                                 set's

fun set_t(set_arrow(_,_,a)) = a;                                                 set't

fun set_ident(a) = set_arrow(a, fn x => x, a);
                                                                                30
fun set_comp(set_arrow(c,g,d),set_arrow(a,f,b)) =                                set'comp
                  if set_eq(c,b) then set_arrow(a,fn x => g(f(x)), d)            set'eq
                  else raise nonCompatiblePair;


(*** instantiation of finite set ***)

val FinSet = cat(set_s, set_t, set_ident, set_comp);

                                                                                40
(*** datatype for coloring elelments ***)

datatype 'a Tag = just of 'a | pink of ('a Tag) | blue of ('a Tag) |             Tag
              pair of ('a Tag) * ('a Tag) | ttrue;


(*** instantiation of initial object ***)

exception undefined;
                                                                                50
val set_initial = (emptyset, fn x => set_arrow(emptyset, fn y => raise undefined, x));


(*** instantiation of coproduct ***)

fun set_coproduct(a,b) = let val sumab = union(mapset(pink)(a), mapset(blue)(b))    set'coproduct
                         val univ = fn (c,
                                        set_arrow(_, f, _),
                                        set_arrow(_, g, _)) =>
                            let val fg = fn pink x => f(x)                      60
                                        | blue x => g(x)
                                        | just x => raise undefined
```

```
                        in set_arrow(sumab,fg,c)
                        end
                  in ((sumab,
                      set_arrow(a,pink,sumab),
                      set_arrow(b,blue,sumab)),
                     univ)
                  end;

(*** instantiation of coequalizer ***)

fun set_coequalizer(set_arrow(s,f,b), set_arrow(_,g,_)) =                    set'coequalizer
   let val cat(_,_, id,comp) = FinSet
   in if is_empty(s) then ((b,id(b)), fn (_,j) => j)
      else if cardinality(s) = 1 then let val (y,_) = singleton_split(s)
                                      in if f(y) = g(y) then ((b,id(b)), fn (_,j) => j)
                                         else let val b' = remove(g(y),b)
                                                  fun ff(z) = if z=g(y) then f(y)
                                                              else z
                                              in((b', set_arrow(b,ff,b')),
                                                  fn (d', set_arrow(_,j,_))
                                                          => set_arrow(b',j,d'))
                                              end
                                      end
         else let val (a,a') = split(s)
                  val ((_,q),univ) = set_coequalizer(set_arrow(a,f,b),
                                                     set_arrow(a,g,b))
                  val ((d,r),univ') = set_coequalizer(comp(q, set_arrow(a',f,b)),
                                                      comp(q, set_arrow(a',g,b)))
              in ((d,comp(r,q)), fn (d',j) => univ'(d',univ(d',j)))
              end
   end


(*** instantiation of io_cp_ce_cat ***)

val io_cp_ce_FinSet = io_cp_ce_cat(FinSet, set_initial, set_coproduct, set_coequalizer);

val cocomplete_FinSet = cocomplete_cat(FinSet, finite_colimit(io_cp_ce_FinSet));


(*** instantiation of terminal object ***)

val set_terminal = let val t = singleton(ttrue)                              set'terminal
               in (t,fn a => set_arrow(a,fn x => ttrue,t))
               end


(*** instantiation of product ***)

fun set_times(a,b) = let val a_list = set_to_list(a)                         set'times
                     val b_list = set_to_list(b)
                     fun s_mult(s,[],l) = s
                       | s_mult(s,a::l',l) = let fun e_mult(r,x,[]) = r
                                                   | e_mult(r,x,y::l) =
                                                         e_mult(pair(x,y)::r,x,l)
                                             in s_mult(e_mult(s,a,l),l',l)
                                             end
                  in list_to_set(s_mult([],a_list,b_list))
                  end;


fun set_product(a,b) = let val a_cross_b = set_times(a,b)                    set'product
```

```
                    val proj_a = set_arrow(a_cross_b, fn pair(y,z) => y, a)
                    val proj_b = set_arrow(a_cross_b, fn pair(y,z) => z, b)
                    val univ = fn (p,
                                   set_arrow(_,f1,_),
                                   set_arrow(_,f2,_)) =>
                                          set_arrow(p,                          130
                                                  fn xy => pair(f1 xy, f2 xy),
                                                  a_cross_b)
                in ((a_cross_b,proj_a,proj_b),univ)
                end


(*** instantiation of equalizer ***)

fun set_equalizer(f,g) =                                                set'equalizer
    let val a = dom(FinSet)(f)                                          140
        fun filter(r,f1,g1,emptyset) = r
          | filter(r,f1,g1,s) = let val (x,s') = singleton_split(s)
                                in if f1(x)=g1(x) then filter(union(singleton(x),r),
                                                              f1,g1,s')
                                   else filter(r,f1,g1,s')
                                end
        val set_arrow(_,f_fn,_) = f
        val set_arrow(_,g_fn,_) = g
        val e = filter(emptyset,f_fn,g_fn,a)
    in ((e,set_arrow(e,fn x => x, a)),                                  150
        (fn (e1,h1) => set_arrow(e1,fn y => h1 y,e)))
    end


(*** instantiation of complete_cat ***)

val completeFinSet =                                                    completeFinSet
    complete_cat(FinSet,
                 dual_colimit(finite_colimit(io_cp_ce_cat(dual_Cat(FinSet),
                                                          set_terminal,     160
                                                          set_product,
                                                          set_equalizer))))
```