```
(*******************************************************************
********************************************************************
********
******** Concrete Problem Solutions
******** Language SML/NY
******** Author M. Bertol (c)
********
******** Release: Tue Aug 13 11:35:40 MET DST 1996
********
********************************************************************
********************************************************************)


(*** computing a push out diagram in set ***)

datatype n_labels = x_a | x_b | x_c;
datatype e_labels = x_f | x_g;


val G = let val N = [x_a,x_b,x_c]
            val E = [x_f,x_g]
            fun src x_f = x_a
              | src x_g = x_a
            fun tgt x_f = x_b
              | tgt x_g = x_c
        in graph(N,E,src,tgt)
        end;

val a_set = union(singleton(just "A"),
                  union(singleton(just "B"),
                        union(singleton(just "C"),
                              union(singleton(just "D"),emptyset))))

val b_set = union(singleton(just "B"),
                  union(singleton(just "E"),
                        union(singleton(just "F"),
                              union(singleton(just "G"),emptyset))))

val c_set = union(singleton(just "C"),
                  union(singleton(just "D"),
                        union(singleton(just "E"),
                              union(singleton(just "F"),
                                    union(singleton(just "G"),emptyset)))))


fun f_fn(just"A") = just "C"
  | f_fn(just"B") = just "C"
  | f_fn(just"C") = just "D"
  | f_fn(just"D") = just "E"

fun g_fn(just"A") = just "B"
  | g_fn(just"B") = just "E"
  | g_fn(just"C") = just "B"
  | g_fn(just"D") = just "F"

fun nodes_to_sets (x_a) = a_set
  | nodes_to_sets (x_b) = b_set
  | nodes_to_sets (x_c) = c_set
```

```
fun edges_to_arrows(x_f) = set_arrow(a_set,f_fn,b_set)
  | edges_to_arrows(x_g) = set_arrow(a_set,g_fn,c_set)

val po_diagram = diagram(G,nodes_to_sets,edges_to_arrows)

let val cocomplete_cat(_,set_colimit) = cocomplete_FinSet in set_colimit(po_diagram) end



(*** definiing the category of (not only set) relations ***)

(*** only composition should be redefined ---- rest is inherited ***)

fun relation_compose(cC)((f',g'),(f,g)) = let val complete_cat(C,_) =cC          relation_compose
                                              val (p,p'),_ = pullback(cC)(g,f')
                                          in (compose(C)(f,p),compose(C)(g',p'))
                                          end;




(*** definiing the category of finite graphs ***)

val cross_product =
         cat_functor(FinSet,
                  fn a => set_times(a,a),
                  fn set_arrow(a,f,b) => set_arrow(set_times(a,a),
                                                   fn pair(x,y) =>
                                                         pair(f(x),f(y)),
                                                   set_times(b,b)),
                  FinSet)


fun I(A) = cat_functor(A, fn x=> x, fn x => x, A);
val graphCat = comma_cat(I(FinSet),cross_product);


val cocomplete_FinGraph = let val C = cocomplete_FinSet                            cocomplete_FinGraph
                    in  cocomplete_comma_cat(C,C)(cocontinous_I(FinSet),
                                                 cross_product)
                    end;
```