Déterminisation des automates


```
(* corrigé du TD Automate 3 *)
(* travail sur des automates non-déterministes *)
(* par Loïc Le Loarer *)

(* le type automate *)
type ('a,'b) automate = { alphabet : 'b list ; init : 'a list ;
 term : 'a list ; trans : ('a * 'b * 'a) list};;

(* Question 1 : la fonction ... *)
let rec ajoute c f = function
  | [] -> [c,[f]]
  | (c',l) :: r when c' = c -> (c,insere f l) :: r
  | h :: r -> h :: (ajoute c f r) ;;

let nouv_trans etats trans =
  it_list
    (fun ntrans (i,c,f) -> if mem i etats then ajoute c f ntrans else ntrans)
    [] trans ;;

let determinise a =
  let trans = ref []
  and etats_restant = ref [a.init]
  and etats = ref [a.init] in
  while !etats_restant <> [] do
    let etat :: r = !etats_restant in
    etats_restant := r;
    let nouv_trans = nouv_trans etat a.trans in
    do_list
      (function (c,f) -> begin
        trans := (etat,c,f) :: !trans;
        if not (mem f !etats) then begin
          etats := f :: !etats;
          etats_restant := f :: !etats_restant
        end
      end) nouv_trans
  done;
  { alphabet = a.alphabet; init = [a.init]; trans = !trans;
    term = it_list
      (fun l c -> if (intersect c a.term)<>[] then insere c l else l)
      [] !etats } ;;

let determinise a =
  let rec aux etats trans = function
    | etat :: r ->
        let nouv_trans = nouv_trans etat a.trans in
        let etats_restant,etats,trans' =
          it_list
            (fun (r,e,t) (c,f) ->
              let t' = (etat,c,f) :: t in
              if not(mem f e) then (f :: r),(f :: e),t' else r,e,t')
            (r,etats,trans) nouv_trans in
        aux etats trans' etats_restant
    | [] -> { alphabet = a.alphabet; init = [a.init]; trans = trans;
            term = it_list
              (fun l c -> if (intersect c a.term)<>[] then insere c l else l)
              [] etats } in
  aux [a.init] [] [a.init] ;;

determinise auto1;;
```