```
(***************************************************************************)
(*                         Dessin de circuits                              *)
(***************************************************************************)

type circuit = AND of circuit * circuit
             | OR  of circuit * circuit
             | NOT of circuit
             | Entree of string ;;

#open "graphics";;

let W = size_x () ;;
let H = size_y () ;;
let LP = 10 ;;

let dessine_porte x y texte =
  set_color blue ;
  moveto (x-LP) (y-LP) ;
  lineto (x+LP) (y-LP) ;
  lineto (x+LP) (y+LP) ;
  lineto (x-LP) (y+LP) ;
  lineto (x-LP) (y-LP) ;
  moveto (x-LP+2) (y-5) ;
  draw_string texte
;;

let dessine_AND x y =
  dessine_porte x y "AND"
;;
let dessine_OR x y =
  dessine_porte x y "OR"
;;
let dessine_NOT x y =
  dessine_porte x y "NOT"
;;


let ligne_brisee (x1,y1) (x2,y2) xi =
  set_color red ;
  moveto x1 y1 ;
  lineto xi y1 ;
  lineto xi y2 ;
  lineto x2 y2
;;


let rec hauteur = function
    AND (c1,c2) -> 1 + max (hauteur c1) (hauteur c2)
  | OR (c1,c2)  -> 1 + max (hauteur c1) (hauteur c2)
  | NOT c       -> 1 + (hauteur c)
  | Entree _    -> 0
;;


let rec largeur = function
    AND (c1,c2) -> (largeur c1) + (largeur c2)
  | OR (c1,c2)  -> (largeur c1) + (largeur c2)
  | NOT c       -> largeur c
  | Entree _    -> 1
;;


let entrees c =
```



```
  let rec collect acc = function
      AND (c1,c2) -> let acc' = collect acc c1 in collect acc' c2
    | OR  (c1,c2) -> let acc' = collect acc c1 in collect acc' c2
    | NOT c       -> collect acc c
    | Entree s    -> if mem s acc then acc else s::acc
  in rev (collect [] c)
;;


let place s l =
  let rec place_rec i = function
      [] -> raise Not_found
    | s'::reste -> if s=s' then i else place_rec (succ i) reste
  in place_rec 1 l
;;

let dessine_circuit c =
  let h = hauteur c in
  let l = largeur c in
  let e = entrees c in
  let ne = list_length e in
  let ligne_entree s =
    let j = place s e in (ne-j+1) * ( H / (ne+1))  in
  let colonne p =
    (h-p+1) * (W / (h+2))  in
  let xi_entree s =
    let j = place s e in (colonne h) + j * W / ((h+2)*(ne+1))  in

  let rec dessine_rec p ymin ymax = function
      AND (c1,c2) ->
        let x = colonne p in
        let l1,l2 = largeur c1, largeur c2 in
        let pasy = (ymax-ymin) / (l1+l2) in
        let ymin1,ymax1 = ymin + (l2) * pasy, ymax in
        let ymin2,ymax2 = ymin, ymin + l2 * pasy in
        let y = (ymax2 + ymin1) / 2 in
        dessine_AND x y ;
        let xc1,yc1 = dessine_rec (succ p) ymin1 ymax1 c1 in
        let xi = (match c1 with Entree s -> xi_entree s | _ -> (xc1+x) / 2) in
        ligne_brisee (xc1,yc1) (x-LP,y+5) xi ;
        let xc2,yc2 = dessine_rec (succ p) ymin2 ymax2 c2 in
        let xi = (match c2 with Entree s -> xi_entree s | _ -> (xc2+x) / 2) in
        ligne_brisee (xc2,yc2) (x-LP,y-5) xi ;
        x+LP,y

    | OR  (c1,c2) ->
        let x = colonne p in
        let l1,l2 = largeur c1, largeur c2 in
        let pasy = (ymax-ymin) / (l1+l2) in
        let ymin1,ymax1 = ymin + (l2) * pasy, ymax in
        let ymin2,ymax2 = ymin, ymin + l2 * pasy in
        let y = (ymax2 + ymin1) / 2 in
        dessine_OR x y ;
        let xc1,yc1 = dessine_rec (succ p) ymin1 ymax1 c1 in
        let xi = (match c1 with Entree s -> xi_entree s | _ -> (xc1+x) / 2) in
        ligne_brisee (xc1,yc1) (x-LP,y+5) xi ;
        let xc2,yc2 = dessine_rec (succ p) ymin2 ymax2 c2 in
        let xi = (match c2 with Entree s -> xi_entree s | _ -> (xc2+x) / 2) in
        ligne_brisee (xc2,yc2) (x-LP,y-5) xi ;
        x+LP,y

    | NOT c       ->
        let y = (ymin + ymax) / 2 in
```

```
        let x = colonne p in
        dessine_NOT x y ;
        let xc,yc = dessine_rec (succ p) ymin ymax c in
        let xi = (match c with Entree s -> xi_entree s | _ -> (xc + x) / 2) in
        ligne_brisee (xc, yc) (x-LP, y) xi ;
        x+LP, y

    | Entree s    ->
        colonne h, ligne_entree s

  in
  set_color black ;
  do_list (fun s -> moveto ((colonne h)-10) (ligne_entree s) ;
               draw_string s) e ;
  dessine_rec 0 10 (H-10) c
;;


let disj1 = OR ( AND (NOT (Entree "a"), Entree "b"),
                 AND (Entree "a", NOT (Entree "b")) ) ;;
let disj2 = OR ( AND (Entree "a", Entree "b"),
                 AND (NOT (Entree "a"), NOT (Entree "b")) ) ;;
let add1 = OR ( AND (disj1, NOT (Entree "cin")),
                AND (disj2, Entree "cin") ) ;;

let essai c =
  clear_graph () ;
  dessine_circuit c
;;
```