Dessin de circuits élémentaires

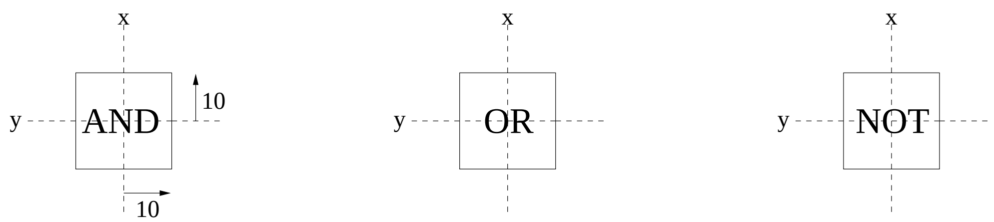
On se propose ici de représenter graphiquement des circuits élémentaires construits à partir de portes logiques AND, OR et NOT. Pour cela, on définit le type `circuit` des formules logiques correspondant à de tels circuits :

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

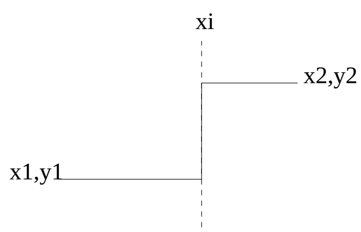
Le constructeur `Entree` correspond à un fil d'entrée du circuit, c'est-à-dire à une variable propositionnelle libre de la formule correspondante.

1 Préliminaires

□ Ecrire des fonctions traçant, à une position `x,y` donnée, les portes AND, OR et NOT, de la manière suivante :



□ Ecrire une fonction `ligne_brisee` traçant une ligne brisée d'un point `x1,y1` à un point `x2,y2` en effectuant un coude sur la colonne `xi`, de la manière suivante :



□ Ecrire une fonction `hauteur` calculant la hauteur d'un circuit (c'est-à-dire sa hauteur en tant qu'arbre), une fonction `largeur` calculant sa largeur et une fonction `parcours : circuit -> string list` collectant toutes les entrées d'un circuit par un parcours préfixe.

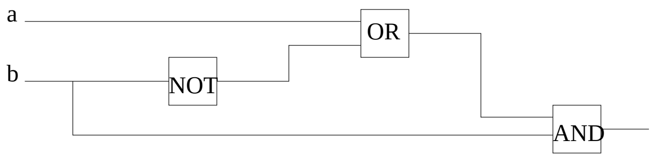
2 Dessin d'un circuit

Etant donné un circuit c à représenter graphiquement, on peut commencer par calculer sa hauteur h et l'ensemble $E = e_1, \ldots, e_n$ de ses entrées par un parcours préfixe. On choisit d'orienter le circuit de la gauche vers la droite : les entrées seront représentées sur une même colonne, à gauche, et les différentes profondeurs de l'arbre sur différentes colonnes, le sommet de l'arbre (le résultat du circuit) se trouvant tout à droite. Il y aura donc h colonnes (une pour les entrées, et $h-1$ pour les différentes profondeurs de l'arbre).

Exemple : le circuit $c = (a + \bar{b})b$, représenté par l'arbre

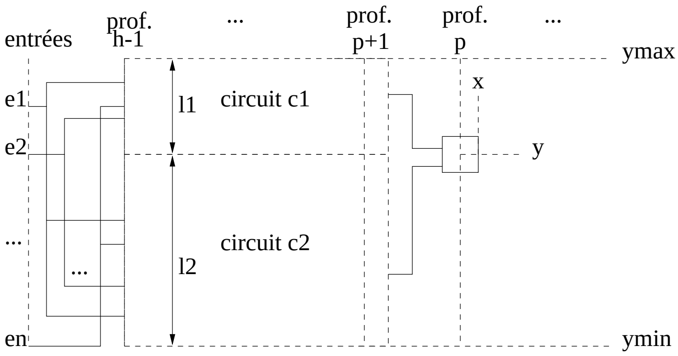
```
AND (OR (Entree "a", NOT (Entree "b")), Entree "b")
```

sera tracé de la manière suivante :



L'idée est d'écrire une fonction de dessin récursive, prenant en argument la profondeur p dans l'arbre, des bornes $ymin$ et $ymax$ fixant les limites des ordonnées pour le tracé, et le sous-arbre T à tracer. Cette fonction renvoie en résultat le point de sortie du circuit qu'elle a tracé.

Cette fonction se rappelle récursivement sur les sous-arbes de T, avec des paramètres $ymin$ et $ymax$ calculés suivants les largeurs de ces sous-arbres (l_1 et l_2 sur le schéma). Les points de sortie renvoyés permettent alors de relier ces circuits à la porte logique définissant T, par des lignes brisées. Lorsque l'on doit connecter un arbre à une entrée (constructeur `Entree`) la position de cette entrée peut être calculée directement, et on trace alors une ligne brisée partant de cette entrée (on choisira des colonnes différentes pour les coudes de ces lignes brisées). Ceci est illustré sur le schéma suivant :



☐ Ecrire une fonction `dessine_circuit` réalisant cette méthode de dessin, et testez la sur le circuit additionneur 1-bit.