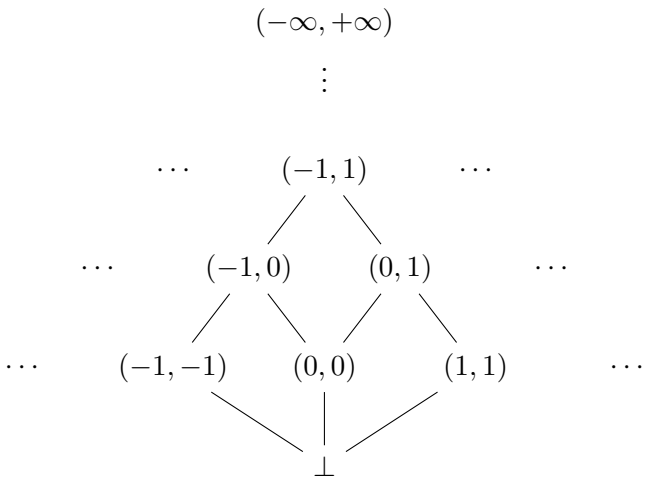
Domaines des intervalles et élargissement

L'objectif de ce TP est d'implémenter le domaine des intervalles et quelques élargissements.

Le domaine des intervalles

Ce domaine permet de borner les variables à chaque point de programme.
Treillis $(\mathscr{D}^\sharp, \sqsubseteq^\sharp)$ avec $\mathscr{D}^\sharp = \bot \cup \{(n_1, n_2) \in (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{+\infty\}) \mid n_1 \leq n_2\}$.



$$\begin{aligned}
\gamma(-\infty, +\infty) &=]\!] -\infty, +\infty [\![\\
\gamma(-\infty, n) &=]\!] -\infty, n]\!] \\
\gamma(n, +\infty) &= [\![n, +\infty [\![\\
\gamma(n_1, n_2) &= [\![n_1, n_2]\!] \\
\gamma(\bot) &= \emptyset
\end{aligned}$$

1. Implémenter ce domaine dans l'analyseur fourni (voir dans le TP précédent comment implémenter un nouveau domaine en partant du fichier `src/domains/dummy.ml`). On gardera pour l'instant les valeurs par défaut pour les fonctions `widening`, `sem_times`, `sem_div`, `backsem_times` et `backsem_div`.

   Indications :
   — On utilisera le type suivant :

   ```
   type t = Bot | Itv of int option * int option
   ```

   où `None` représente $\pm\infty$ et `Some` n la borne finie n [1].
   — Il pourra s'avérer pratique d'étendre certaines fonctions des entiers sur $\mathbb{Z} \cup \{-\infty\}$ ou $\mathbb{Z} \cup \{+\infty\}$. Exemple pour "$\leq$" :

   ```
   (* Extension de <= à Z U {-oo}. *)
   let leq_minf x y = match x, y with
     | None, _ -> true (* -oo <= y *)
     | _, None -> false (* x > -oo (x != -oo) *)
     | Some x, Some y -> x <= y

   (* Extension de <= à Z U {+oo}. *)
   let leq_pinf x y = match x, y with
     | _, None -> true (* x <= +oo *)
   ```

1. Pour rappel, le type `option`, disponible par défaut en OCaml, est défini comme suit : `type 'a option = None | Some of 'a`.

```
  | None, _ -> false (* +oo > y (y != +oo) *)
  | Some x, Some y -> x <= y
```

— On pourra utiliser la petite fonction suivante pour maintenir l'invariant de type $n_1 \leq n_2$ lorsqu'on crée des intervalles :

```
let mk_itv o1 o2 = match o1, o2 with
  | None, _ | _, None -> Itv (o1, o2)
  | Some n1, Some n2 -> if n1 > n2 then Bot else Itv (o1, o2)
```

2. Tester le domaine sur le programme suivant (fichier `examples/ex09.tiny`) :

```
i=0;
while (i < 10) {
  ++i;
}
```

puis sur le même programme en remplaçant 10 par 1 000 000. Qu'observe t-on ? (utiliser l'option `-v 2` si on ne voit aucune différence)

3. Pour remédier au problème, implémenter l'élargissement vu en cours :

$$x^\sharp \triangledown y^\sharp = \begin{cases} [\![a, b]\!] & \text{si } x^\sharp = [\![a, b]\!], y^\sharp = [\![c, d]\!], c \geq a, d \leq b \\ [\![a, +\infty[\![& \text{si } x^\sharp = [\![a, b]\!], y^\sharp = [\![c, d]\!], c \geq a, d > b \\ ]\!-\infty, b]\!] & \text{si } x^\sharp = [\![a, b]\!], y^\sharp = [\![c, d]\!], c < a, d \leq b \\ ]\!-\infty, +\infty[\![& \text{si } x^\sharp = [\![a, b]\!], y^\sharp = [\![c, d]\!], c < a, d > b \\ y^\sharp & \text{si } x^\sharp = \bot \\ x^\sharp & \text{si } y^\sharp = \bot \end{cases}$$

4. Tester le nouveau domaine sur les programmes précédents puis sur le programme suivant (fichier `examples/ex10.tiny`) :

```
i = 0; j = 0;
while (i < 10) {
  if (i <= 0) {
    ++i;
    j = i;
  } else {
    ++i;
  }
}
```

Quel intervalle obtient t-on pour la variable `j` ? Proposer un nouvel élargissement permettant d'obtenir la réponse exacte $[\![0, 1]\!]$ (indice : cet élargissement est dit *avec retard*).

Question subsidiaire : que se passe t-il dans votre domaine si le programme analysé contient des expressions telles celles de `examples/ex08.tiny` ? On peut essayer d'y remédier à l'aide du module `InfInt` fourni (documentation : `src/doc/InfInt.html`).