Corrigé du DS3

1. Décidabilité

1. On suppose par l'absurde qu'il existe un programme P qui prend en entrée p et x et renvoie vrai ssi p s'arrête sur l'entrée x.
Construisons alors P' de la manière suivante:

P' prend en entrée un programme p:

il lance P sur p et p
si la réponse est vrai alors il boucle (par exemple avec une commande while(true){})
et si la réponse est faux alors il renvoie vrai.

Ainsi P' sur P' s'arrête ssi la réponse de P sur P' et P' est fausse càd ssi P' ne s'arrête pas sur P'! Ceci est contradictoire et donc P n'existe pas.

2. Supposons par l'absurde qu'il existe un programme P qui prend en entrée p et qui ~~s'arrête~~ renvoie vrai ssi celui-ci s'arrête sur au moins 10 arguments.

Construisons alors un programme P' qui résout le problème de l'arrêt.
Ceci étant impossible, on en déduira que P n'existe pas.

P' sur l'entrée p x :

- on construit le programme f dont l'entrée est y et qui lance p sur x puis renvoie y.
- on renvoie le résultat de P sur f

On remarque que si p s'arrête sur x alors f est une instance positive de ARRÊT, >, 10 car s'arrête sur toutes ses entrées et donc P' renverra vrai sur p x comme le fait P sur f.
Dans le cas contraire, f ne s'arrête sur aucune entrée et donc P sur f renvoie faux.
On a bien P' sur p et x renvoie vrai ssi p s'arrête sur x i.e P' résout le problème de l'ARRÊT.

3. Supposons par l'absurde l'existence de P qui résout SOMME et construisons comme précédemment P' qui résout ARRÊT :

P' sur p et x

- on construit un programme f qui prend en entrée a, b et c et qui lance p sur x puis renvoie vrai ssi a+b=c
- on renvoie le résultat de P sur f

On remarque alors que p s'arrête sur x ssi f est une instance positive de SOMME et donc P' est bien un programme qui décide ARRÊT.

2. Jeux dans un graphe :

Q1. joueur 1 : $f_1 : \{1\} \to \{0,2\}$ ou $f_1' : \{1\} \to \{0,2\}$
$1 \to 0$ $\quad$ $1 \to 2$

joueur 2 : $f_2 : \{0,2\} \to \{1\}$
$0 \to 1$
$2 \to 1$

Le joueur 1 a deux stratégies ≠ alors que le joueur 2 en a une seule.

Q2. 0 est une position gagnante pour J_2.
Si 1 suit f_1' alors on remarque que la partie ne passera jamais dans l'état 0 qui est alors une position gagnante pour J_1 ; on remarque que l'état 2 est aussi une position gagnante pour J_1.

Ainsi, $R_1 = \{1, 2\}$ avec f_1' ↑ stratégie gagnante dans les deux cas. $\quad$ $R_2 = \{0\}$ avec f_2 ↑ stratégie gagnante

Q3. J_2 jouera toujours suivant f_2.
Si J_1 suit f_1 alors une partie ne passera pas une infinité de fois en 2 et s'il suit f_1', elle ne passera pas une infinité de fois en 0 donc aucune position n'est gagnante pour J_1.

De plus, si on considère la partie 0-1-2-0-1-2-0-1-2... qui suit la stratégie f_2, on remarque qu'il n'y a pas non plus de stratégie gagnante pour J_2 depuis ces sommets et le jeu n'est donc pas positionnel.

Q4. Supposons que le jeu est positionnel et soient f_1 et f_2 les stratégies associées :

Supposons par l'absurde que $R_1 \cap R_2 \neq \emptyset$ et soit $s \in R_1 \cap R_2$

Supposons que $s \in S_1$

Construisons la partie suivante $s_0 = s$
$s_1 = f_1(s)$
$s_2 = f_2(s_1)$ etc...

càd $s_0 = s$ et $\forall i \geq 0$ $s_{2i+1} = f_1(s_{2i})$
et $s_{2i+2} = f_2(s_{2i+1})$

Cette partie est donc une f_1-partie mais aussi une f_2-partie. Elle doit être gagnante pour les joueurs 1 et 2 car $s_0 = s \in R_1 \cap R_2$ ce qui est absurde.

Le cas $s \in S_2$ est analogue

```
let transpose g =
  let n = Array.length g in
  let res = Array.make n [] in
  for i = 0 to n-1 do
    parcours g.(i) i res;
  done;
  res
```

vec:

```
let rec parcours l s tab = match l with
  | [] -> ()
  | x::next -> tab.(x) <- s::tab.(x);
               parcours next s tab
```

Il existe une stratégie gagnante depuis s ssi il existe un chemin de s vers un sommet de T dans G ou de manière équivalente un chemin d'un sommet de T vers s dans G transposé.
Dans ce cas, lorsqu'on a un chemin de t vers s alors pour tout sommet x sur ce chemin, $f(x)$ est le prédecesseur de x sur le chemin et donc un parcours en profondeur lancé depuis chaque sommet de T et mettant à jour le tableau conviendra et aura la complexité souhaitée.

```
let strategie g tab =
    let n = ARRAY.length g in
    let res = ARRAY.make n (-1) in
    let gt = transpose g in
    let rec parcours p s =
        if (res.(s) == -1) then (
            if (tab.(s)) then
                begin
                res.(s) ← n;
                List.iter (parcours s) gt.(s);
            else begin
                end
                res.(s) ← p;
                List.iter (parcours s) gt.(s);
                end
        )
    in
        for i = 0 to n-1 do
            if (tab.(s)) then
                parcours (-1) s;
        done;
        res
```

Complexité : transpose est en $O(|S|+|A|)$ et on parcourt une fois chaque liste d'adjacence : $O(|A|)$ plus $O(|S|)$ pour initialiser les tableaux.

Q7:

$(Attr_i(X))_i$ est une suite croissante pour l'inclusion incluse dans S un ensemble fini, elle est donc stationnaire d'où le résultat.

Q8: $R_1 = Attr(T_1)$

$R_2 = S \setminus Attr(T_1)$

Stratégies gagnantes :

Pour J_1 : Soit $s \in R_1 \cap S_1$ alors il existe n t.q $s \in A_n(T_1)$ mais $s \notin A_{n-1}(T_1)$ et il existe t t.q $(s,t) \in A$ et $t \in A_n(T_1)$.
On pose $f(s) = t$

si $s \in R_2$, il peut jouer n'importe quoi

Pour J_2 : Soit $s \in R_2 \cap S_2$ alors $\forall n$, $s \notin A_n(T_1)$ càd qu'il existe $t \notin A_{n-1}(T_1)$ t.q $(s,t) \in A$
On prend un tel $t \notin A_k(T_1)$ t.q $(s,t) \in A$ comme image de s par la stratégie.

Q10: Il existe une stratégie gagnante depuis s ssi il existe $t \in T$ t.q qu'il existe un chemin de s à t et un cycle ($\neq \emptyset$) contenant t.

Pour identifier cela on peut commencer par utiliser Kosaraju afin de récupérer les CFC de taille > 1, tous les sommets qu'elles contiennent sont sur des cycles $\neq \emptyset$ et tous les sommets accessibles depuis les sommets ainsi sélectionnés dans G^T sont donc des positions gagnantes.