Définition 4.1 - *jeu sans mémoire*

On appelle *jeu sans mémoire* un jeu dans lequel à tout instant de la partie, il est possible de déterminer si un joueur a gagné ou si un coup est valide, *indépendament des précédents coups joués*.

Définition 4.2 - *jeu à information complète*

On appelle *jeu à information complète* un jeu dans lequel il n'y *aucune information cachée* que les joueurs ne puissent *savoir ou prévoir*.

Définition 4.3 - *graphe associé à un jeu à deux joueurs*

Un jeu à deux joueurs J_1 et J_2, sans mémoire, à information complète peut être *représenté par un graphe orienté biparti* :

$$G = (S, A) \quad \text{où } S = S_1 \sqcup S_2,\ A \subset (S_1 \times S_2) \cup (S_2 \times S_1)$$

Les sommets de S_1 sont appelés les *états contrôlés par* J_1 et ceux de S_2 les *états conrôlés par* J_2.

Définition 4.4 - *coup possible pour un joueur*

Soit un jeu à deux joueurs J_1 et J_2, sans mémoire, à information complète, de graphe associé $G = (S_1 \sqcup S_2, A)$. Pour $a \neq b$ dans $\{1; 2\}$, on appelle l'arc $(s_a, s_b) \in A \cap (S_a \times S_b)$ *un coup possible pour* J_a *depuis l'état* s_a *vers un état* s_b *contrôlé par* J_b.

Définition 4.5 - *jeu d'accessiblité*

Un jeu à deux joueurs J_1 et J_2, sans mémoire, à information complète de graphe associé $G = (S_1 \sqcup S_2, A)$ est dit *d'accessibilité* si toute partie du jeu prend fin dès lors qu'un joueur atteint un état dit *final* : il en existe trois types :

1. les états gagnants pour J_1, dont l'ensemble est appelé *condition de gain* F_1

2. les états gagnants pour J_2, dont l'ensemble est appelé *condition de gain* F_2

3. les états de match nul, dont l'ensemble est appelé F_0

Nécessairement, ces trois ensembles sont deux à deux disjoints.

Définition 4.6 - *partie partielle*

Soit $G = (S, A)$ le graphe associé à un jeu d'accessiblité. On appelle *partie partielle du jeu* tout chemin de G partant de l'état inital de G à un état quelconque de G. *L'ensemble des parties partielles du jeu est noté* S^w.

Définition 4.7 - *stratégie*

Soit $G = (S, A)$ le graphe associé à un jeu d'accessiblité. Une stratégie est une *application $\varphi : S^w \to S$ qui à une partie partielle $(s_0, \dots, s_p)$ associe un sommet la prolongeant* : $\Big(s_p,\, \varphi\big((s_0, \dots, s_p)\big)\Big) \in A$. On dit alors qu'*un joueur suit une stratégie*.
Pour un jeu sans mémoire, l'image d'une stratégie ne dépend que du dernier sommet de la partie partielle : $\varphi\big((s_0, \dots, s_p)\big) = \varphi(s_p)$ moralement.

Définition 4.8 - *stratégie gagnante*

Soit $G = (S, A)$ le graphe associé à un jeu d'accessiblité. Une *stratégie φ est gagnante depuis un état s* lorsque depuis s, le joueur qui la suit gagne peu importe le choix de l'adversaire.

Définition 4.9 - *suite convergeant vers l'attracteur*

Soit $G = (S, A)$ le graphe associé à un jeu d'accessiblité. Pour $a \neq b$ dans $\{1; 2\}$, on définit $(\mathcal{A}_j^a)_{j \in \mathbb{N}}$ *l'ensemble des positions permettant au joueur J_a de gagner en au plus j coups*. On a bien sûr $\mathcal{A}_0^a = F_a$, puis pour gagner en au plus $j \in \mathbb{N}^*$ coups :

- ou bien J_a peut gagner en au plus $j - 1$ coups,
- ou bien J_a dispose d'un coup le permettant ensuite de gagner en au plus $j - 1$ coups,
- ou bien enfin, J_b se trouvant sur une position non finale, ne dispose que de coups permettant ensuite à J_a de gagner en au plus $j - 1$ coups.

En conclusion :

$$\begin{aligned}\mathcal{A}_j^a = {} & \mathcal{A}_{j-1}^a \cup \\ & \Big\{s \in S_a,\ \exists t \in \mathcal{A}_{j-1}^a,\ (s, t) \in A\Big\} \cup \\ & \Big\{s \in S_b,\ s \text{ non final et } \forall t \in S,\ (s, t) \in A \implies t \in \mathcal{A}_{j-1}^a\Big\}\end{aligned}$$

Définition 4.10 - *attracteur d'un joueur*

Soit $G = (S, A)$ le graphe associé à un jeu d'accessiblité. Pour $a \neq b$ dans $\{1; 2\}$, la suite $(\mathcal{A}_j^a)_{j \in \mathbb{N}}$ est croissante pour l'inclusion ($\mathcal{A}_0^a \subset \mathcal{A}_1^a \subset \dots$) et majorée par S. C'est d'après le théorème de la limite monotone une suite convergente, on note $\mathcal{A}^a$ *sa limite, appelée attracteur du joueur J_a* :

$$\mathcal{A}^a = \lim_{j \to +\infty} \mathcal{A}_j^a$$

Il s'agit de l'ensemble des positions gagnantes pour le joueur J_a.

Théorème 4.11 - *de Zermelo*

Dans un jeu à **deux joueurs fini** (à parties finies), à **information complète** et **sans match nul**, pour tout état du jeu, il existe une stratégie gagnante pour l'un des joueurs partant de cet état.

Définition 4.12 - *heuristique pour MinMax*

Dans le cadre de l'algorithme MinMax appliqué à un jeu d'accessibilité de joueurs J_1 et J_2 de graphe associé $G = (S, A)$, *une fonction heuristique pour faire gagner J_1* doit vérifier :

$$\begin{aligned} h : S &\longrightarrow \mathbb{Z} \\ s &\longmapsto \begin{cases} M & \text{si } s \in F_1 \\ -M & \text{si } s \in F_2 \\ h(s) \in [\![-M+1,\, M-1]\!] & \text{sinon} \end{cases} \end{aligned}$$

où M est un entier naturel assez grand pour modéliser l'infini.

Implémentation - *MinMax classique - obtention du score*

- **Entrée** :
  - $\mathcal{A}$ l'arbre associé au jeu
  - n un noeud de $\mathcal{A}$ (état du jeu)
  - p la profondeur d'exploration
  - $h : S \to \mathbb{Z}$ une heuristique
- **Sortie** : score du sommet n

```
MinMax(𝒜, n, p, h) :
    si n est final ou p = 0 :
        renvoyer h(n)
    sinon :
        si n ∈ S1:
            res = −∞ // ou tout minorant de h
            pour tout fils f de n:
                v = MinMax(𝒜, f, p−1, h)
                res = max(v,res)
            renvoyer res
        si n ∈ S2:
            res = +∞ // ou tout majorant de h
            pour tout fils f de n:
                v = MinMax(𝒜, f, p−1, h)
                res = min(v,res)
            renvoyer res
```

Implémentation - *MinMax - meilleur coup possible*

- **Entrée** :
  - $\mathcal{A}$ l'arbre associé au jeu
  - n un noeud de $\mathcal{A}$ (état **non final** du jeu)
  - p la profondeur d'exploration
  - $h : S \to \mathbb{Z}$ une heuristique
- **Sortie** : f le fils de n correspondant au meilleur coup (au score le plus souhaitable)

```
score = MinMax(𝒜, n, p, h)
pour tout f fils de n :
    v = MinMax(𝒜, f, p-1, h)
    si v == score :
    renvoyer f
```

Implémentation - *MinMax avec élagage Alpha-Bêta - obtention du score*

L'appel initial se fait avec $\alpha = -\infty$ et $\beta = +\infty$.

- **Entrée** :
  - $\mathcal{A}$ l'arbre associé au jeu
  - n un noeud de $\mathcal{A}$ (état du jeu)
  - p la profondeur d'exploration
  - $h : S \to \mathbb{Z}$ une heuristique
  - α et β bornant le score de n
- **Sortie** : score de n

```
AlphaBeta(𝒜, n, p, h, α, β) :
    si n est final ou p = 0 :
        renvoyer h(n)
    sinon :
        si n ∈ S1:
            a = α
            pour tout fils f de n:
                v = AlphaBeta(𝒜, f, p−1, h, a, β)
                a = max(v,a)
                si a >= β // ⟦α, β⟧ est vide ou un singleton : on élague
                    renvoyer a
            renvoyer a
        si n ∈ S2:
            b = β
            pour tout fils f de n:
                v = AlphaBeta(𝒜, f, p−1, h, α, b)
                b = min(v,b)
                si b <= α // ⟦α, β⟧ est vide ou un singleton : on élague
                    renvoyer b
            renvoyer b
```