Programme de colle 1

Thibault Mabillotte

2 Questions de Cours exigibles

1. Preuve de la déterminisation d'un automate

Soit $\mathcal{A} = (Q, I, F, \delta)$ un automate fini non déterministe qu'on suppose sans ϵ-transition. Montrons qu'on peut déterminiser $\mathcal{A}$ c'est-à-dire trouver un automate $\widehat{\mathcal{A}}$ déterministe tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\widehat{\mathcal{A}})$.

Posons $\widehat{\mathcal{A}} = \left(\mathcal{P}(Q), I, \{P \subset Q \,|\, P \cap F \neq \emptyset\}, \widehat{\delta}\right)$. On définit alors la fonction de transition $\widehat{\delta}$ par :

$$\forall P \subset \mathcal{P}(Q),\ \forall a \in \Sigma,\ \widehat{\delta}(P, a) = \{q \in Q \,|\, \exists p \in P,\ q \in \delta(p, a)\} = \bigcup_{p \in P} \delta(p, a)$$

Ici, $P \subset Q$ est une partie de Q donc un état de $\widehat{\mathcal{A}}$.

Montrons par récurrence que $\forall m \in \Sigma^*,\ \forall P \subset Q,\ \widehat{\delta}^*\left(P, m\right) = \bigcup\limits_{p \in P} \delta^*(p, m)$.

Soit $n \in \mathbb{N}$. On pose H_n : "$\forall m \in \Sigma^*,\ |m| = n,\ \forall P \subset Q,\ \widehat{\delta}^*\left(P, m\right) = \bigcup\limits_{p \in P} \delta^*(p, m)$"

Pour $n = 0$. Soit $m \in \Sigma^*$ tel que $|m| = 0$. On a directement que $m = \epsilon$. Soit $P \subset Q$. D'une part, $\widehat{\delta}^*\left(P, \epsilon\right) = P$ et d'autre part, $\bigcup_{p \in P} \delta^*\left(p, \epsilon\right) = \bigcup_{p \in P}\{p\} = P$. H_0 est donc vraie.

Supposons que H_n soit vraie et montrons H_{n+1}. Soit $m \in \Sigma^*$ tel que $m = ua$ avec $|u| = n$. Soit $P \subset Q$. On a alors :

$$\widehat{\delta}^*\left(P, m\right) = \widehat{\delta}^*\left(P, ua\right) = \widehat{\delta}^*\left(\widehat{\delta}^*\left(P, u\right), a\right)$$

Par hypothèse de récurrence, $\widehat{\delta}^*\left(P, u\right) = \bigcup\limits_{p \in P} \delta^*(p, u)$ d'où :

$$= \widehat{\delta}^*\left(\bigcup_{p \in P} \delta^*(p, u), a\right) = \widehat{\delta}\left(\bigcup_{p \in P} \delta^*(p, u), a\right)$$

Par définition de $\widehat{\delta}$ on trouve :

$$= \bigcup_{q \in \bigcup_{p \in P} \delta^*(p,u)} \delta(q, a) = \bigcup_{p \in P}\left(\bigcup_{q \in \delta^*(p,u)} \delta(q, a)\right)$$

Or, par définition d'un automate fini non déterministe, $\bigcup\limits_{q \in \delta^*(p,u)} \delta(q, a) = \delta^*(p, ua)$. On a donc finalement :

$$\widehat{\delta}^*\left(P, m, =\right) \bigcup_{p \in P} \delta^*(p, m)$$

Ainsi, H_{n+1} est vraie. Par principe de récurrence, la propriété est vraie pour tout $n \in \mathbb{N}$.

Finalement, on a l'équivalence suivante :

$$m \in \mathcal{L}(\widehat{\mathcal{A}}) \iff \widehat{\delta}^*\,(I, m) \in F' \iff \widehat{\delta}^*\,(I, m) \cap F \neq \emptyset \iff \left(\bigcup_{i \in I} \delta^*(i, m)\right) \cap F \neq \emptyset$$

$$\iff \exists i \in I,\ \delta^*(i, m) \cap F \neq \emptyset \iff m \in \mathcal{L}(\mathcal{A})$$

Il existe donc bien un automate déterministe $\widehat{\mathcal{A}}$ tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\widehat{\mathcal{A}})$.

2. Preuve de la construction du produit cartésien de deux automates

Définition : Soient $\mathcal{A}_1 = (Q_1, q_0^1, F_1, \delta_1)$ et $\mathcal{A}_2 = (Q_2, q_0^2, F_2, \delta_2)$ deux automates finis déterministes qu'on suppose complets. On définit l'automate produit de la façon suivante :

$$\mathcal{A} = \big(Q_1 \times Q_2, (q_0^1, q_0^2), F, \delta\big) \text{ avec } F \subset Q_1 \times Q_2 \text{ et } \delta : \begin{array}{ccc} (Q_1 \times Q_2) \times \Sigma & \to & Q_1 \times Q_2 \\ (q_1, q_2), a & \mapsto & (\delta_1(q_1, a), \delta_2(q_2, a) \end{array}$$

Propriété : Soient $\mathcal{A}_1 = (Q_1, q_0^1, F_1, \delta_1)$ et $\mathcal{A}_2 = (Q_2, q_0^2, F_2, \delta_2)$ deux automates finis déterministes complets et $\mathcal{A}$ l'automate produit. On a alors que :

$$\forall (q_1, q_2) \in Q_1 \times Q_2, \forall m \in \Sigma^*,\ \delta^*\big((q_1, q_2), m\big) = \big(\delta_1^*(q_1, m), \delta_2^*(q_2, m)\big)$$

Démonstration : On procède par récurrence sur la taille des mots.

Soit $n \in \mathbb{N}$. Posons H_n : "$\forall m \in \Sigma^*,\ |m| = n,\ \forall (q_1, q_2) \in Q_1 \times Q_2,\ \delta^*\big((q_1, q_2), m\big) = \big(\delta_1^*(q_1, m), \delta_2^*(q_2, m)\big)$"

Pour $n = 0$. Soit $m \in \Sigma^*$ tel que $|m| = 0$. On a directement que $m = \epsilon$. Soit $(q_1, q_2) \in Q_1 \times Q_2$. D'une part, $\delta^*\left((q_1, q_2), \epsilon\right) = (q_1, q_2)$. D'autre part, $\big(\delta_1^*(q_1, \epsilon), \delta_2^*(q_2, m)\big) = (q_1, q_2)$. H_0 est donc vraie.

Supposons que H_n soit vraie et montrons H_{n+1}. Soit $m \in \Sigma^*$ tel que $m = au$ avec $|u| = n$. Soit $(q_1, q_2) \in Q_1 \times Q_2$. On a alors :

$$\delta^*\left((q_1, q_2), au\right) = \delta^*\left(\delta\big((q_1, q_2), a\big), u\right) = \delta^*\left(\big(\delta_1(q_1, a), \delta_2(q_2, a)\big), u\right)$$

Par principe de récurrence on trouve :

$$= \Big(\delta_1^*\big(\delta_1(q_1, a), u\big), \delta_2^*\big(\delta_2(q_2, a), u\big)\Big) = \Big(\delta_1^*\big(q_1, au\big), \delta_2^*\big(q_2, au\big)\Big)$$

On a donc que $\delta^*\left((q_1, q_2), m\right) = (\delta_1^*(q_1, m), \delta_2^*(q_2, m))$. Ainsi, H_{n+1} est vraie donc par principe de récurrence, la propriété est vraie pour tout $n \in \mathbb{N}$.

Propriété : Soient L_1 et L_2 deux langages reconnaissables. Alors $L_1 \cap L_2$ et $L_1 \cup L_2$ sont reconnaissables.

Démonstration : Soient L_1 et L_2 deux langages reconnaissables. Il existe alors $\mathcal{A}_1$ et $\mathcal{A}_2$ deux automates finis déterministes complets qui reconnaissent respectivement L_1 et L_2. Notons F_1 et F_2 l'ensemble des états finaux de $\mathcal{A}_1$ et $\mathcal{A}_2$.

On considère $\mathcal{A}$ l'automate produit de $\mathcal{A}_1$ et $\mathcal{A}_2$ tel que l'ensemble de ses états initiaux soient $F = F_1 \times F_2$. On a alors :

$$m \in \mathcal{L}(\mathcal{A}) \iff \delta^*\left((q_0^1, q_0^2), m\right) \in F = F_1 \times F_2 \iff \left(\delta_1^*(q_0^1, m), \delta_2^*(q_0^2, m)\right) \in F_1 \times F_2$$

$$\iff \delta_1^*(q_0^1, m) \in F_1 \text{ et } \delta_2^*(q_0^2, m) \in F_2 \iff m \in L_1 \text{ et } m \in L_2$$

On trouve donc que $m \in \mathcal{L}(\mathcal{A}) \iff m \in L_1 \cap L_2$. Donc $L_1 \cap L_2 = \mathcal{L}(\mathcal{A})$. Ainsi, $L_1 \cap L_2$ est reconnaissable.

On considère $\mathcal{A}$ l'automate produit de $\mathcal{A}_1$ et $\mathcal{A}_2$ tel que l'ensemble de ses états initiaux soient $F = (Q_1 \times F_2) \cup (Q_2 \times F_2)$. On a alors :

$$m \in \mathcal{L}(\mathcal{A}) \iff \delta^*\left((q_0^1, q_0^2), m\right) \in F = (Q_1 \times F_2) \cup (Q_2 \times F_2)$$

$$\iff \delta^*\left((q_0^1, q_0^2), m\right) \in Q_1 \times F_2 \text{ ou } \delta^*\left((q_0^1, q_0^2), m\right) \in Q_2 \times F_1$$

$$\iff \left(\delta_1^*(q_0^1, m), \delta_2^*(q_0^2, m)\right) \in Q_1 \times F_2 \text{ ou } \left(\delta_1^*(q_0^1, m), \delta_2^*(q_0^2, m)\right) \in F_1 \times Q_2$$

Comme les automates $\mathcal{A}_1$ et $\mathcal{A}_2$ sont complets, on a que les assertions $\delta_1^*(q_0^1, m) \in Q_1$ et $\delta_2^*(q_0^2, m) \in Q_2$ sont toujours vraies. Ainsi :

$$\iff \delta_2^*(q_0^2, m) \in F_2 \text{ ou } \delta_1^*(q_0^1, m) \in F_1 \iff m \in L_2 \text{ ou } m \in L_1$$

On a alors que $m \in \mathcal{L}(\mathcal{A}) \iff m \in L_1 \cup L_2$. Autrement dit, $L_1 \cup L_2 = \mathcal{L}(\mathcal{A})$. Ainsi, $L_1 \cup L_2$ est reconnaissable.

3. Formule des attracteurs

Définition : Soit un jeu (G, s, T_1, T_2) avec $G = (S_1 \sqcup S_2, A)$ bipartite. On définit la suite des attracteurs pour le joueur i, notée $(A_n(i))_{n\in\mathbb{N}}$, par :

$$\left\{\begin{array}{lcl} A_0(i) & = & T_i \\ A_{n+1}(i) & = & A_n(i) \cup \{s \in S_i \,|\, \exists v \in A_n(i),\ (s,v) \in A\} \cup \{s \in S_{3-i} \,|\, \forall v \in S,\ (s,v) \in A \Rightarrow v \in A_n(i)\} \end{array}\right.$$

4. Pseudo-code de l'algorithme minmax alpha-beta et exemple

Pseudo-code

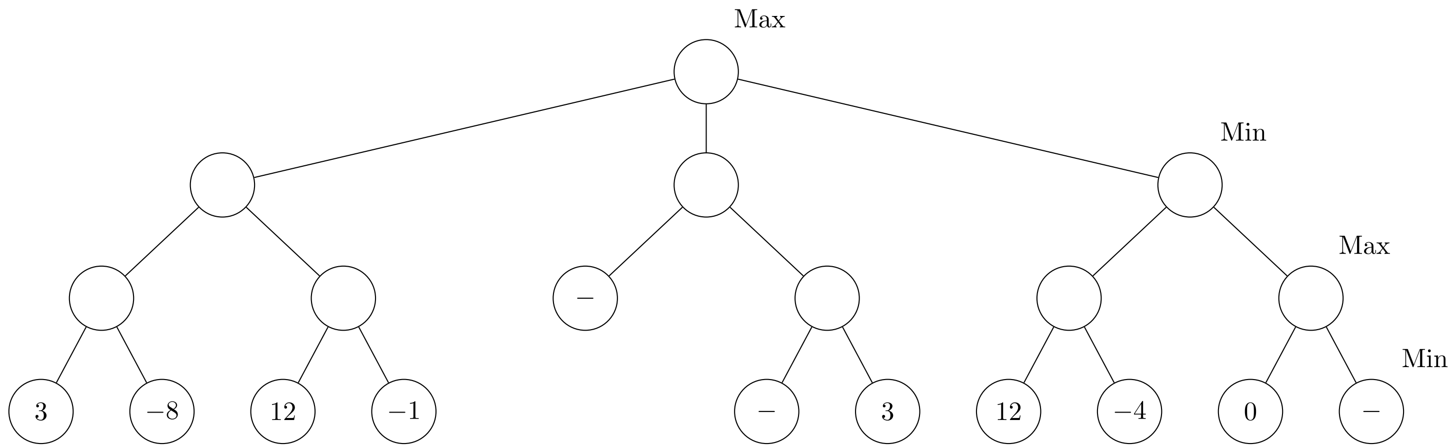
Algorithm 1 Algorithme minmax avec élagage alpha-beta

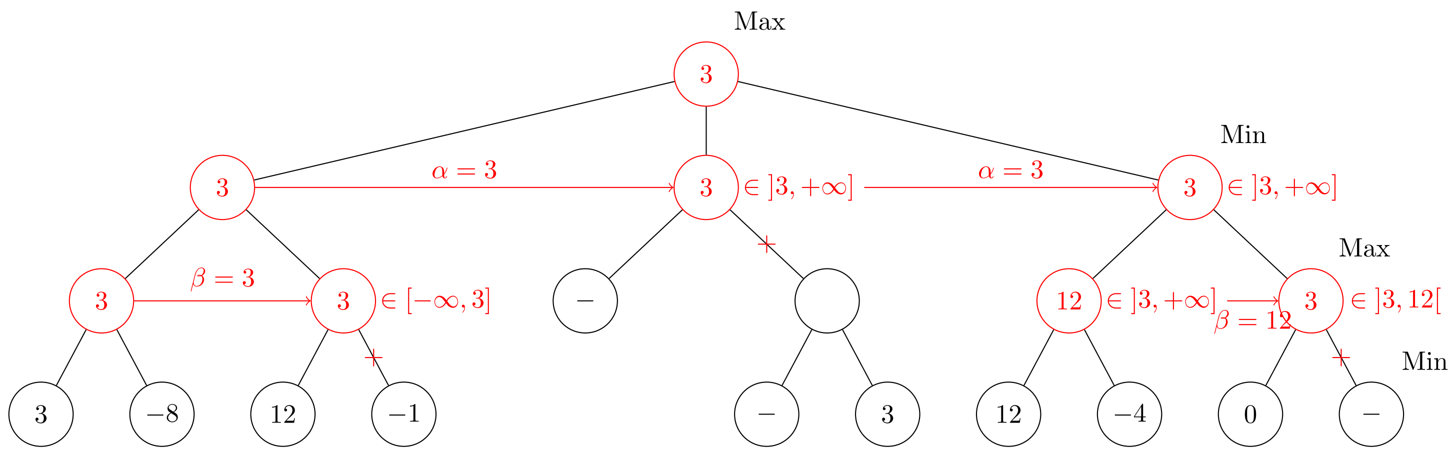
Require: $s \in S$, $p \in \mathbb{N}$, $\alpha = -\infty$, $\beta = +\infty$

```
procedure MinMax(s, p, α, β)
    if s est terminal then
        if s ∈ T_1 then
            return +∞
        else if s ∈ T_2 then
            return −∞
        else
            return 0
        end if
    end if
    if p = h_max then
        return h(s)
    else if s ∈ S_1 then
        for all voisin s' de s do
            x ← MinMax(s', p + 1, α, β)
            if x ≥ β then
                return β
            end if
            if x > α then
                α ← x
            end if
        end for
        return α
    else
        for all voisin s' de s do
            x ← MinMax(s', p + 1, α, β)
            if x ≤ α then
                return α
            end if
            if x < β then
                β ← x
            end if
        end for
        return β
    end if
end procedure
```

Exemple :

Exemple résolu d'utilisation du minmax avec élagage alpha-beta. $-$ et $+$ tiennent pour $-\infty$ et $+\infty$.





5. Code de la structure union find avec toutes les optimisations

En Ocaml

```
type union_find = {n : int ; link : int array ; rank : int array }

let create (n : int) : union_find =
  {n = n ; link = Array.init n (x->x) ; rank = Array.make n 0}

let rec find (uc : union_find) (i : int) : int =
  if uc.link.(i) = i then i
  else
        let r_i = find uc uc.link.(i) in
        uc.link.(i) <- r_i;
        r_i

let union (uc : union_find) (i : int) (j : int) : unit =
  let r_i = find uc i in
  let r_j = find uc j in
  if r_i <> r_j then
        begin
          if uc.rank.(r_i) > uc.rank.(r_j) then
                uc.link.(r_j) <- r_i
          else if uc.rank.(r_j) > uc.rank.(r_i) then
                uc.link.(r_i) <- r_j
          else
                uf.link.(r_i) <- r_j ;
                uf.rank.(r_j) <- uf.rank.(r_j) + 1 ;
        end
```

En C

```
struct union_find {
  int n ;
  int* link ;
  int* rank ;
}

typdef struct union_find union_find

union_find* create(int n) {
  union_find* uf = malloc(sizeof(union_find)) ;
  uf->n = n ;
  uf->link = malloc(sizeof(int)*n) ;
  uf->rank = malloc(sizeof(int)*n) ;
  for (int i = 0; i<n ; i+=1) {
    uf.link[i] = i ;
    uf.rank[i] = 0 ;
  }
  return uf ;
}

int find(union_find* uf, int i) {
  int p = uf->link[i] ;
  if (p==i) {
    return i ;
  }
  else {
    int r = find(uf,p) ;
    uf->link[i] = r ;
    return r ;
  }
}

void union(union_find* uf, int i, int j) {
  int r_i = uf->link[i] ;
  int r_j = uf->link[j] ;
  if (r_i != r_j) {
    if (uf->rank[r_i] > uf->rank[r_j]) {
      uf->link[r_j] <- r_i ;
    }
    elif (uf->rank[r_j] > uf->rank[r_i]) {
      uf->link[r_i] <- r_j ;
    }
    else {
      uf->link[r_i] <- r_j ;
      uf->rank[r_j] <- uf->rank[r_j] + 1 ;
    }
  }
}
```

6. Preuve du lemme de l'étoile

Lemme (de l'étoile) : Soit L un langage.

Si L est reconnaissable alors $\exists N \in \mathbb{N},\ \forall m \in L,\ |m| \geq N,\ \exists u, v, w \in \Sigma^*,\ m = uvw,\ v \neq \epsilon,\ |uv| \leq N,\ \forall k \in \mathbb{N},\ uv^k w \in L$

Démonstration : Soit L un langage reconnaissable.

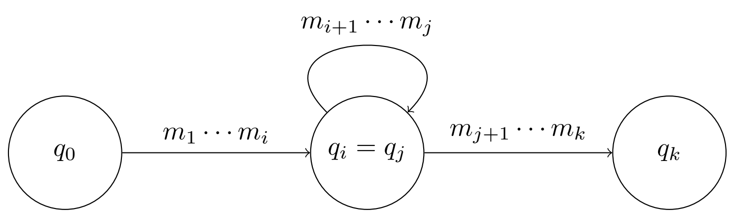
L est reconnaissable donc il existe un automate complet déterministe $\mathcal{A} = (Q, q_0, F, \delta)$ tel que $L = \mathcal{L}(\mathcal{A})$. On choisit de poser N le nombre d'états de $\mathcal{A}$.

Soit $m \in L$ un mot quelconque de L tel que $|m| \geq N$. En particulier, si un tel mot n'existe pas dans L la propriété est vérifiée. Notons $m = m_1 \cdots m_k$ avec $k \geq n$.

Comme $m \in L$, l'automate $\mathcal{A}$ reconnait le mot m. Donc $\delta^*(q_0, m) \in F$. On pose alors :

$$\forall i \in [\![1, k]\!],\ q_i = \delta^*(q_0, m_1 \cdots m_i)$$

De ce fait, $\mathrm{card}\,(Q) = N$ et $\mathrm{card}\,(\{q_0, q_1, \cdots, q_k\}) = k + 1$. Par principe des tiroirs, il existe donc $i, j \in [\![1, N]\!]$ tels que $i \neq j$ et $q_i = q_j$. On suppose sans perte de généralité que $i < j$ et on a :



On choisit de poser $u = m_1 \cdots m_i$, $v = m_{i+1} \cdots m_j$ et $w = m_{j+1} \cdots m_k$. On trouve donc que $q_i = \delta^*(q_0, u)$, $q_j = q_i = \delta^*(q_i, v)$ et $q_k = \delta^*(q_j, w) \in F$. On a donc bien que $m = uvw$, $|uv| \leq N$ car $j \leq N$ et $v \neq \epsilon$ car $i < j$.

On montre par récurrence que $\forall p \in \mathbb{N},\ \delta^*(q_i, v^p) = q_i$. Ainsi si $p \in \mathbb{N}$:

$$\delta^*(q_0, uv^p w) = \delta^*\left(\delta^*(q_0, u), v^p w\right) = \delta^*(q_i, v^p w) = \delta^*\left(\delta^*\left(q_i, u^p\right), w\right) = \delta^*(q_i, w) = q_k \in F$$

On a donc que $uv^p w \in \mathcal{L}(\mathcal{A})$. Autrement dit, $uv^p w \in L$. Ce qui démontre le lemme de l'étoile.

7. Montrer que le langage $L = \{a^n b^n \,|, n \in \mathbb{N}\}$ n'est pas reconnaissable à l'aide du lemme de l'étoile

Contraposée du lemme de l'étoile : Soit L un langage.

Si $\forall N \in \mathbb{N},\ \exists m \in L,\ |m| \geq N,\ \forall u, v, w \in \Sigma^*,\ m = uvw,\ v \neq \epsilon,\ |uv| \leq N,\ \exists k \in \mathbb{N},\ uv^k w \notin L$ alors L n'est reconnaissable

Exercice : Soit $L = \{a^n b^n \,|, n \in \mathbb{N}\}$. Montrons à l'aide du lemme de l'étoile que L n'est pas reconnaissable.

Utilisons la contraposée du lemme de l'étoile.
Soit $N \in \mathbb{N}$ un entier quelconque.
On fixe $m = a^N b^N$. D'une part, $m \in L$ et, d'autre part, $|m| = 2N \geq N$.
Soient $u, v, w \in \Sigma^*$ tels que $m = uvw$, $v \neq \epsilon$ et $|uv| \leq N$. Ainsi, il existe $i, j \in [\![0, N]\!]$ tels que $i < N$, $0 < j$ et

$$u = a^i,\ v = a^j \text{ et } w = a^{N-i-j} b^N$$

On choisit $k = 0$ et on trouve que $av^k w = a^{N-j} b^N$. Or, $j \neq 0$ donc $N - j < N$. Autrement dit, $av^k w \notin L$.
Ainsi, il existe k tel que $uv^k w \notin L$. D'après le lemme de l'étoile, L n'est pas reconnaissable.

8. Montrer que le langage $L = \{a^n b^n \,|, n \in \mathbb{N}\}$ n'est pas reconnaissable à l'aide de la méthode des résiduels

Exercice : Soit $L = \{a^n b^n \,|, n \in \mathbb{N}\}$. Montrons à l'aide de la méthode des résiduels que L n'est pas reconnaissable.

Supposons qu'il existe $\mathcal{A} = (Q, q_0, F, \delta)$ un automate fini déterministe et complet tel que $L = \mathcal{L}(\mathcal{A})$. Pour montrer que L n'est pas reconnaissable, on va montrer que $\mathcal{A}$ n'existe pas. Pour cela, on prouve que quelque soit $n \in \mathbb{N}$, $\operatorname{card}(Q) \geq n$ c'est-à-dire que $\operatorname{card}(Q) = +\infty$ ce qui est impossible car $\mathcal{A}$ est un automate fini déterministe.

Soit $n \in \mathbb{N}$. On considère la famille F contenant $u_0 = \epsilon$, $u_1 = a$, $u_2 = a^2$, $\ldots$ et $u_{n-1} = a^{n-1}$. On remarque alors que pour tout $i, j \in [\![0, n-1]\!]$:

$$u_i b^i \in L \text{ et } u_j b^i \notin L$$

Supposons par l'absurde que $\operatorname{card}(Q) < n$. Comme la famille F est constituée de n élément distincts, par principe des tiroirs, il existe $i, j \in [\![0, n-1]\!]$ tels que $i \neq j$ et $\delta^*(q_0, u_i) = \delta^*(q_0, u_j)$. On trouve alors que :

$$\delta^*\left(q_0, u_i b^i\right) = \delta^*\left(\delta^*\left(q_0, u_i\right), b^i\right) = \delta^*\left(\delta^*\left(q_0, u_j\right), b^i\right) = \delta^*\left(q_0, u_j b^i\right)$$

On trouve ainsi que $\delta^*\left(q_0, u_i b^i\right) = \delta^*\left(q_0, u_j b^i\right)$. Pourtant $\delta^*(q_0, u_i) \in F$ et $\delta^*\left(q_0, u_j b^i\right) \notin F$. On trouve une absurdité, donc $\operatorname{card}(Q) \geq n$. D'après ce qui a été écrit plus haut, L n'est pas reconnaissable.

9. Le complémentaire d'un graphe reconnaissable est reconnaissable

Théorème : Soit L un langage. Si L est reconnaissable alors $\overline{L}$ est reconnaissable.

Preuve : Soit L un langage reconnaissable.

L est reconnaissable donc il existe un automate complet détermiste $\mathcal{A} = (Q, q_0, F, \delta)$ tel que $L = \mathcal{L}(\mathcal{A})$.

$\mathcal{A}$ est complet donc quelque soit $m \in \Sigma^*$, il existe un état q de $\mathcal{A}$ tel que $q = \delta^*(q_0, m)$. On a alors l'équivalence suivante :

$$m \notin \mathcal{L}(\mathcal{A}) \Longleftrightarrow \exists q \in Q \backslash F,\ \delta^*(q_0, m) = q$$

On pose alors $\mathcal{A}' = (Q, q_0, F' = Q\backslash F, \delta)$. Ainsi :

$$m \notin \mathcal{L}(\mathcal{A}) \Longleftrightarrow \exists q \in Q\backslash F,\ \delta^*(q_0, m) = q \Longleftrightarrow \exists q \in F', \delta^*(q_0, m) = q \Longleftrightarrow m \in \mathcal{L}(\mathcal{A}')$$

Autrement dit, $m \notin L \Longleftrightarrow m \in \mathcal{L}(\mathcal{A}')$. Donc $\overline{L} = \mathcal{L}(\mathcal{A}')$. L'automate $\mathcal{A}'$ reconnait $\overline{L}$ donc le complémentaire de L est reconnaissable.

10. Démonstration du théorème fondamental sur les arbres couvrants de poids minimal

Définition : Soient $G = (S, A, p)$ un graphe pondéré connexe, $S' \subset S$ et $e = (x, y) \in A$. On dit que l'arête e est sûre pour S' dans G si elle admet exactement une extrémité dans S' et qu'elle est de poids minimal parmi toutes les arêtes ayant exactement une extrémité dans S'.

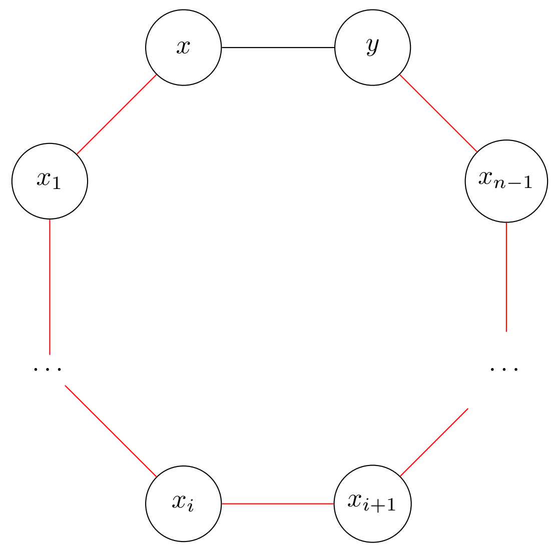
Théorème : Soit $G = (S, A, p)$ un graphe connexe pondéré par des poids distincts. En notant $T = (S, A_0)$ son unique arbre couvrant de poids minimal on a :

$$\forall S' \subset S,\ S' \neq \emptyset, \text{ si } (x, y) \text{ est sûre pour } S' \text{ dans } G \text{ alors } (x, y) \in A_0$$

Démonstration : Soient $S' \subset S$ tel que $S' \neq \emptyset$ et $e = (x, y)$ une arête sûre pour S' dans G. e est une arête sûre donc, sans perte de généralité, supposons que $x \in S'$ et $y \notin S'$.

On suppose par l'absurde que $e \notin A_0$.

$T = (S, A_0)$ est un arbre donc par définition T est connexe. Ainsi, il existe un unique chemin de x à y qu'on note $c : x = x_0 \to x_1 \to \cdots \to x_{n-1} \to x_n = y$. En particulier, $e = (x, y) \notin A_0$ donc c n'est pas réduit à l'arête e.



On note $i + 1 = \min\limits_{1 \leq j \leq n} \{x_j \notin S'\}$. Ce minimum existe car $x_n = y \notin S'$ et que j est minoré par 0. Par construction de ce minimum, $x_i \in S'$. Ainsi, l'arête $e' = (x_i, x_{i+1})$ possède exactement une extrémité dans S'. Comme e est une arête sûr pour S' dans G on a :

$$p(e) < p(e')$$

On considère alors le graphe $T' = (S, A'_0)$ avec $A'_0 = (A_0 \backslash \{e'\}) \cup \{e\}$. Comme $e \notin A_0$, $\text{card}\,(A_0) = \text{card}\,(A'_0)$. Par ailleurs, T' est connexe car il existe un chemin de x_i à x_{i+1} grâce à l'arête e. Cette assertion reste vrai pour tout les sommets du chemin c. Les sommets de S hors du chemin c ne sont pas affecté par la suppression de e'.

$T = (S, A_0)$ est un arbre donc $\text{card}\,(A_0) = \text{card}\,(S) - 1$. Ainsi, $\text{card}\,(A'_0) = \text{card}\,(A_0) = \text{card}\,(S) - 1$. Donc le graphe $T' = (S, A'_0)$ est connexe avec $\text{card}\,(A'_0) = \text{card}\,(S) - 1$ donc est acyclique. Autrement dit, T' est un arbre.

Le poids de l'arbre T' s'écrit alors :

$$p(T') = p(T) - p(e') + p(e)$$

Or $p(e') > p(e)$ donc :

$$p(T') = p(T) - p(e') + p(e) < p(T)$$

On trouve donc que $p(T') < p(T)$. Or T est de poids minimal. On trouve alors une absurdité donc $e \in A_0$.