Complexité de Communication

Iordanis Kerenidis Frédéric Magniez David Xiao

24 janvier 2014

Résumé

Imaginons un scénario où deux protagonistes (ou joueurs), Alice et Bob, souhaitent calculer de manière collaborative une fonction $f(x, y)$ alors que x est détenue par Alice et y par Bob. La complexité de communication de la fonction f est le nombre de bits qu'Alice et Bob doivent s'échanger afin de pouvoir calculer $f(x, y)$. Les autres ressources, telles que temps et espace mémoire, sont ici ignorées afin de se focaliser uniquement sur la communication requise.

Plus formellement, Alice et Bob s'échangent des bits à tour de rôle en suivant un protocole, qui définit à chacun instant qui doit envoyer le prochain bit, ou si le protocole est terminé. Lorsque le protocole est terminé, la valeur de $f(x, y)$ doit alors être connue des deux joueurs, ou bien d'un seul des deux joueurs dans le contexte de communication unilatérale (où seule Alice envoie un message à Bob). Ce scénario modélise une communication parfaite, où chaque message arrive à son destinataire sans être perdu ou modifié. Ce sera le cas de tout ce chapitre.

L'étude de la complexité de communication permet d'unifier des preuves de bornes inférieures dans plusieurs domaines dont les automates, les machines de Turing, les circuits VLSI, les arbres de décision, les branching programs ou les OBDDs, et plus récemment les algorithmes de streaming ou bien encore le property testing.

Toutes les notions et résultats présentés se généralisent aux relations, mais nous ne considèrerons que les fonctions.

Nous regroupons ci-dessous les principales fonctions dont la complexité de communication sera étudiée. Notons dorénavant $[n] = \{1, 2, \ldots, n\}$ pour tout entier n.

$$
\begin{aligned}
\textsc{Index}_n : \quad & \{0,1\}^n \times [n] \to \{0,1\} \\
& (x, i) \mapsto x_i. \\
\textsc{Equality}_n : \quad & \{0,1\}^n \times \{0,1\}^n \to \{0,1\} \\
& (x, y) \mapsto \begin{cases} 1, & \text{si } x = y\,; \\ 0, & \text{sinon.} \end{cases} \\
\textsc{Disjointness}_n : \quad & \{0,1\}^n \times \{0,1\}^n \to \{0,1\} \\
& (x, y) \mapsto \begin{cases} 1, & \text{si } x_i = y_i = 1 \text{ pour un } i \in [n]\,; \\ 0, & \text{sinon.} \end{cases} \\
\textsc{GreaterThan}_n : \quad & \{0,1\}^n \times \{0,1\}^n \to \{0,1\} \\
& (x, y) \mapsto \begin{cases} 1, & \text{si } x > y \text{ (ordre lexicographique)}\,; \\ 0, & \text{sinon.} \end{cases}
\end{aligned}
$$

Enfin, dans tout ce chapitre les logarithmes seront pris en base 2.

Protocole 1 – Protocole unilatéral pour Maximum

```
- Alice: Envoyer  m := max(x_1, x_2, ..., n)  à Bob
- Bob: Calculer   out := max(m, y_1, y_2, ..., n)
```

1 Protocoles déterministes

1.1 Communication unilatérale

Définition 1.1 (Protocole déterministe unilatéral)**.** *Soient X, Y deux ensembles finis. Un* protocole déterministe unilatéral π *entre deux joueurs, Alice et Bob, est la description d'une conversation unilatérale d'Alice vers Bob comme suit. Pour toute entrée $x \in X$ d'Alice et entrée $y \in Y$ de Bob :*

1. *Alice envoie un message $m(x)$ de ℓ bits à Bob, où $m : X \to \{0,1\}^\ell$;*
2. *Bob calcule la* sortie *$out(m, y)$, où out est définie sur $\{0,1\}^\ell \times Y$.*

La complexité (de communication) $\mathsf{C}(\pi, x)$ *de π sur l'entrée x est la taille en bits du message $m(x)$ d'Alice. La* complexité (de communication) $\mathsf{C}(\pi)$ *de π est la valeur maximale de sa complexité de communication pour toutes les entrées $x \in X$.*

Remarquons que la taille ℓ du message d'Alice est décidée par le protocole, et non pas par l'entrée x d'Alice. En particulier, des scénarios où Bob pourrait tirer de l'information de la "non-réception" d'un message d'Alice, ou encore de la taille du message d'Alice, ne sont pas autorisés.

Puisque m est définie par l'entrée x, la sortie out du protocole est donc aussi une fonction de (x, y) qu'on pourra noter $out(x, y)$ au lieu de $out(m, y)$.

Comme pour les algorithmes, un protocole π *calcule* une fonction f si $out(x, y) = f(x, y)$ pour toutes entrées (x, y). Etant donnée une fonction f, définissions la *complexité de communication déterministe unilatérale* par la quantité

$$\overrightarrow{\mathsf{D}}(f) = \min\{\mathsf{C}(\pi) : \text{protocole } \pi \text{ déterministe unilatéral calculant } f\}.$$

Puisqu'Alice peut toujours se contenter d'envoyer la totalité de son entrée x à Bob, on peut déjà établir que

$$\overrightarrow{\mathsf{D}}(f) \leq \lceil \log |X| \rceil.$$

Exemple 1.2. *Soit $X = Y = [n]^n$. Définissons $f : X \times Y \to [n]$ par $f(x_1, \ldots, x_n, y_1, \ldots, y_n) = \max(x_1, \ldots, x_n, y_1, \ldots, y_n)$. Le protocole 1 calcule f avec complexité de communication $\lceil \log n \rceil$.*

Comme nous le voyons dans l'exemple précédent, $\overrightarrow{\mathsf{D}}(f)$ peut être beaucoup plus petite que le majorant $\lceil \log |X| \rceil$. Afin de caractériser $\overrightarrow{\mathsf{D}}(f)$, introduisons la notion de *matrice de communication* M_f de f. Cette matrice est indexée par $x \in X$, pour les lignes, et par $y \in Y$, pour les colonnes, et satisfait

$$(M_f)_{x,y} = f(x, y).$$

Théorème 1.3. *Soient X, Y deux ensembles finis, et f une fonction définie sur $X \times Y$.*

$$\overrightarrow{\mathsf{D}}(f) = \lceil \log(\textit{nombre de lignes distinctes dans } M_f) \rceil.$$

Protocole 2 – Protocole unilatéral optimal

- Préparation :
  Soit $L_1, L_2, \ldots, L_k$ les lignes sans multiplicité de M_f
  ordonnées arbitrairement (ordre connu d'Alice et Bob)
- Alice :
  Soit $i \in [k]$ l'entier tel que $L_i = (M_f(x,y))_{y \in Y}$
  Envoyer $m := i$ à Bob
- Bob : Calculer $out := L_m[y]$

Démonstration. Posons $k =$ nombre de lignes distinctes dans M_f, et considérons un protocole π quelconque pour f. Nous commençons par montrer qu'Alice doit pouvoir envoyer au moins k messages différents lorsque x varie, c'est-à-dire que $\mathsf{C}(\pi) \geq \lceil \log(k) \rceil$. Pour cela, montrons que si deux lignes x et x' de M_f sont distinctes alors nécessairement $m(x) \neq m(x')$. Par contraposition, si $m(x) = m(x')$ alors pour toute entrée $y \in Y$ le protocole calcule $out(m(x), y) = out(m(x'), y)$. Puisque π calcule f, il s'en suit que $f(x,y) = f(x',y)$, pour tout $y \in Y$, et donc que les lignes x et x' de M_f sont identiques.

Pour montrer $\overrightarrow{\mathsf{D}}(f) \leq \lceil \log k \rceil$, nous construisons un protocole π pour f tel que $\mathsf{C}(\pi) = \lceil \log k \rceil$. Soit L l'ensemble des lignes de M_f. Par hypothèse, l'ensemble L est de taille k. Le protocole 2 calcule alors correctement f et est de complexité annoncée. □

Une conséquence de ce résultat est la caractérisation des complexités suivantes, dont nous laissons les preuves en exercice :

Lemme 1.4.
$$\overrightarrow{\mathsf{D}}(\textsc{Equality}_n) = \overrightarrow{\mathsf{D}}(\textsc{Index}_n) = \overrightarrow{\mathsf{D}}(\textsc{Disjointness}_n) = n.$$

1.2 Cas général

Lorsque l'intéraction entre les deux joueurs est permise, la sortie du protocole doit pouvoir être calculée par chacun des deux joueurs. Un protocole est alors défini comme suit.

Définition 1.5 (Protocole déterministe (cas général))**.** *Soient X, Y deux ensembles finis. Un* protocole déterministe π *entre deux joueurs, Alice (A) et Bob (B), est la description d'une conversation entre Alice et Bob comme suit. Pour toute entrée $x \in X$ d'Alice et entrée $y \in Y$ de Bob :*

1. *Alice envoie un premier message $m_1(x)$ de ℓ_1 bits à Bob ;*
2. *Bob répond par un deuxième message $m_2(y, m_1)$ de $\ell_2(m_1)$ bits à Alice ;*
3. *Et ainsi de suite tant que le protocole n Pour $i \geq 1$, Alice envoie $m_{2i-1}(x, m_1, m_2, \ldots, m_{2i-2})$ de $\ell_{2i-1}(m_1, m_2, \ldots, m_{2i-2})$ bits, et Bob $m_{2i}(y, m_1, m_2, \ldots, m_{2i-1})$ de $\ell_{2i}(m_1, m_2, \ldots, m_{2i-1})$ bits.*
4. *La fin du protocole est décidée simultanément par les deux joueurs en fonction des messages envoyées $m_1, m_2, \ldots$ ainsi que de x pour Alice, et de y pour Bob.*
5. *A la fin du protocole, Alice et Bob peuvent calculer une même* sortie $out(x,y)$*, qui dépend des messages échangés ainsi que de x pour Alice, et de y pour Bob.*

Protocole 3 – Protocole pour Médiane

```
- Partir de a = 1, b = n
- Pour i allant de 1 à ⌈log n⌉
        Alice et Bob: Calculer v = ⌈(a + b)/2⌉
        Alice: Calculer et envoyer m_{2i−1} := |{i : x_i ≥ v}|
        Bob:
                Calculer k := m_{2i−1} + |{i : x_i ≥ v}|
                Si k ≤ n alors envoyer m_{2i} := 0
                Sinon envoyer m_{2i} := 1
        Alice et Bob:
                Si m_{2i} = 0 alors affecter b := v
                Sinon affecter a := v
- Fin du protocole
        La sortie du protocole est out := v
```

La transcription $m(x, y)$ *de* π *sur l'entrée* (x, y) *est la concaténation des messages échangés* $m(x, y) := m_1 m_2 \ldots$. *La* complexité (de communication) $\mathsf{C}(\pi, x, y)$ *de* π *sur l'entrée* (x, y) *est la taille en bits de la transcription correspondante. La* complexité (de communication) $\mathsf{C}(\pi)$ *de* π *est la valeur maximale de sa complexité de communication pour toutes les entrées* $(x, y) \in X \times Y$.

Exemple 1.6. *La valeur médiane d'une famille* $z_1, z_2, \ldots$ *d'éléments de* $[n]$ *est définie par*

$$\text{médiane}(z) = \min\{v \in [n] : |\{i : z_i \geq v\}| \leq |\{i : z_i \leq v\}|\}.$$

Pour $X = Y = [n]^n$, *le protocole 3 permet de calculer par une recherche dichotomique la valeur médiane de* $(x, y) \in X \times Y$ *avec une complexité de communication de* $\lceil \log n \rceil (\lceil \log n \rceil + 1)$.

La notion de complexité de communication est alors définie de manière analogue au cas de communication unilatérale :

$$\mathsf{D}(f) = \min\{\mathsf{C}(\pi) : \text{protocole } \pi \text{ déterministe calculant } f\}.$$

Une propriété fondamentale des protocoles de communication est que les entrées (x, y) menant à la même transcription forment un rectangle : Un *rectangle* est un ensemble R tel que pour tout $(x, y) \in R$ et $(x', y') \in R$, alors $(x, y') \in R$ et $(x', y) \in R$. Un corollaire est que la sortie d'un protocole ne dépend en fait que de sa transcription. Il est donc possible à quiconque qui écoute la communication entre Alice et Bob de calculer la fonction sans même connaître x ou y !

Lemme 1.7. *Soit* π *un protocole déterministe sur* $X \times Y$. *Soit une paire d'entrées* (x, y) *et* (x', y') *de* π *ayant même transcription* t. *Alors* (x', y) *et* (x, y') *ont aussi même transcription, de plus cette transcription est* t.

En particulier, les sorties de π *sur les entrées* (x, y), (x', y), (x, y') *et* (x', y') *sont identiques.*

Preuve (esquisse). Considérons deux paires d'entrées (x, y) et (x', y') ayant même transcription $t = m(x, y) = m(x', y')$. Posons $t = t_1 t_2 \ldots$ avec $t_i = m_i(x, y)$. Par récurrence sur i, on montre que le i-ème message de π sur (x', y) est t_i, et de façon similaire sur (x, y').

Puisque la sortie $out(x, y)$ ne dépend que de la transcription et x, ou de la transcription et y, on en déduit que $out(x, y) = out(x', y)$, puis que $out(x', y) = out(x', y')$, soit au final que $out(x, y) = out(x', y')$. □

Nous verrons que cette propriété permet de montrer des bornes inférieures importantes, mais commençons par des bornes simples et générales sur $\mathsf{D}(f)$.

Lemme 1.8. *Soit f une fonction sur $X \times Y$. Alors la complexité de communication déterministe de f satisfait*

$$\lceil \log |f(X \times Y)| \rceil \leq \mathsf{D}(f) \leq \lceil \log(\min\{|X|, |Y|\}) \rceil + \lceil \log |f(X \times Y)| \rceil.$$

Démonstration. Pour toute fonction f sur $X \times Y$, le majorant est obtenu en considérant le protocole qui consiste pour Alice à envoyer x à Bob, et pour Bob à calculer et envoyer $f(x, y)$ à Alice. Ce protocole échange $\lceil \log(|X|) \rceil + \lceil \log(|f(X \times Y)|) \rceil$ bits. On peut aussi demander à Bob d'envoyer y à Alice, et laisser Alice prendre la décision (dans ce cas m_1 est par convention vide). La complexité de communication déterministe de f satisfait donc toujours

$$\mathsf{D}(f) \leq \lceil \log(\min\{|X|, |Y|\}) \rceil + \lceil \log(|f(X \times Y)|) \rceil.$$

Inversement, puisque la sortie d'un protocole déterministe P ne dépend que de sa transcription, il faut qu'il ait au moins autant de transcriptions que de sorties possibles et donc $\mathsf{D}(f)$ satisfait aussi

$$\lceil \log(|f(X \times Y)|) \rceil \leq \mathsf{D}(f).$$

□

Remarquons qu'un protocole P induit un pavage de M_f en rectangles *monochromatiques*, c'est-à-dire où les valeurs de M_f sont constantes.

Lemme 1.9. *Soit π un protocole de communication pour f. Alors M_f admet une partition en au plus $2^{\mathsf{C}(\pi)}$ rectangles monochromatiques.*

Démonstration. Pour chaque transcription t possible de π, définissons le sous-ensemble $R_t = \{(x, y) \in X \times Y : m(x, y) = t\}$, et montrons que ces sous-ensembles forment une partition de M_f en rectangles monochromatiques, dont le nombre est bien au plus $2^{\mathsf{C}(\pi)}$.

Tout d'abord, d'après la lemme 1.7, il s'agit bien de rectangles. De plus, la sortie de π ne dépend que de son transcript, et cette sortie calcule f, donc ces rectangles sont bien monochromatiques. Enfin, ces rectangles sont par définition couvrants et d'intersections deux à deux vides. □

Une première conséquence de ce pavage esr la méthode suivante pour prouver des bornes inférieures.

Théorème 1.10. *Soient X, Y deux ensembles finis. Soient f une fonction définie sur $X \times Y$ et $S \subseteq X \times Y$ tels que*

1. *f est constante sur S ;*
2. *tout $(x, y), (x', y') \in S$ satisfait $f(x, y) \neq f(x', y)$ ou $f(x, y) \neq f(x, y')$.*

Alors

$$\mathsf{D}(f) \geq \lceil \log(1 + |S|) \rceil.$$

Démonstration. Soit $z = f(x, y)$, pour un élément $(x, y) \in S$ quelconque, la valeur que f prend sur S. Chaque paire d'éléments $(x, y), (x', y') \in S$ est nécessairement dans un rectangle monochromatique (pour la valeur z) différent de M_f à cause de l'hypothèse (2). Donc il faut au moins $|S|$ rectangles pour former une partition des coefficients de M_f ayant pour valeur z. Puisque la fonction f n'est pas constante, il faudra donc au moins un rectangle monochromatique de plus pour obtenir tous les coefficients de M_f.

Cependant, tout protocole π pour f induit une partition en $2^{\mathsf{C}(\pi)}$ rectangles monochromatiques, d'où le résultat. □

Le théorème 1.10 permet de montrer que

Lemme 1.11.

$$\mathsf{D}(\textsc{Equality}_n) = \mathsf{D}(\textsc{Disjointness}_n) = n + 1.$$

Démonstration. La majoration est obtenue par le protocole trivial du Lemme 1.8. Pour obtenir la minoration il suffit de considérer pour $\textsc{Equality}_n$ l'ensemble $S = \{(x, x) : x \in \{0, 1\}^n\}$, et pour $\textsc{Disjointness}_n$ l'ensemble $S = \{(x, \overline{x}) : x \in \{0, 1\}^n\}$, où $\overline{x}$ est obtenu en prenant la négation de tous les bits de x. Dans les deus cas, l'ensemble S est de taille 2^n et satisfait les hypothèses du théorème 1.10. Par conséquent, la quantité $\lceil \log(1 + 2^n) \rceil = n + 1$ est un minorant de chacune des complexités. □

Cet autre lemme permet d'établir une borne inférieure sur $\mathsf{D}(f)$ à partir de propriétés algébriques de M_f. Notons $\delta_z(f)$ la fonction qui prend la valeur 1 en (x, y) si $z = f(x, y)$ et la valeur 0 sinon.

Théorème 1.12. *Soient X, Y deux ensembles finis, et $f : X \times Y \mapsto Z$.*

$$\mathsf{D}(f) \geq \left\lceil \log \left(\sum_{z \in Z} \operatorname{rang}(M_{\delta_z(f)}) \right) \right\rceil.$$

Démonstration. Fixons un protocole π pour f. Etant donnée une transcription t possible de π, appelons $R_t = \{(x, y) \in X \times Y : m(x, y) = t\}$. Nous interprèterons aussi R_t comme la matrice booléenne indexée par $x \in X$ et $y \in Y$, et prenant la valeur 1 en $(x, y) \in R_t$ et 0 sinon. D'après le lemme 1.7, nous savons que R_t est un rectangle, et que toute entrée de R_t mène vers une même sortie $out(t)$. Pour $z \in Z$, définissons $S_z := \{t : out(t) = z\}$. Alors

$$M_{\delta_z(f)} = \sum_{t \in S_z} R_t.$$

Le théorème du rang nous dit alors que

$$\operatorname{rang}(M_{\delta_z(f)}) \leq \sum_{t \in S_z} \operatorname{rang}(R_t) = |S_z|,$$

où l'égalité provient du fait que chaque R_t est non nul, et donc de rang au moins 1, pour $t \in S_z$.

La preuve se conclut en observant que $\sum_{z \in Z} |S_z|$ représente le nombre de transcriptions différentes du π peut transcrire lorsque son entrée (x, y) parcours $X \times Y$. Par conséquent sa complexité est au moins

$$\mathsf{C}(\pi) \geq \left\lceil \log \left(\sum_{z \in Z} |S_z| \right) \right\rceil.$$

□

Pour le problème $\textsc{Equality}_n$, le théorème 1.12 nous permet de redémontrer que

$$\mathsf{D}(\textsc{Equality}_n) \geq n+1.$$

En effet, observons que $M_{\delta_1(f)}$ est la matrice identité de taille $2^n \times 2^n$, et est donc de rang n. De plus $M_{\delta_0(f)}$ n'est pas nulle, et donc $\mathsf{D}(\textsc{Equality}_n) > n$, ou encore $\mathsf{D}(\textsc{Equality}_n) \geq n+1$ puisque $\mathsf{D}(\textsc{Equality}_n)$ est une valeur entière.

2 Protocoles probabilistes

Lorsque les joueurs, Alice et Bob, ont accès à une source aléatoire, le protocole devient probabiliste : son exécution, sa transcription et sa sortie peuvent dépendre des choix aléatoires d'Alice et Bob. La notion de calcul d'une fonction f par un protocole est alors relâchée comme pour les algorithmes en tolérant probabilité d'erreur au plus ε, pour $0 < \varepsilon < 1/2$, c'est-à-dire en imposant que la sortie du protocole soit correcte avec probabilité au moins $(1-\varepsilon)$, pour chaque entrée (x, y). Cette probabilité d'erreur, pour une entrée (x, y) fixée, ne dépend donc que des choix aléatoires réalisés par les joueurs.

La source aléatoire peut être de deux natures différentes :

— Soit elle est propre à chaque joueur, c'est-à-dire les bits aléatoires r_A d'Alice sont indépendants des bits aléatoires r_B de Bob. On parle alors d'*aléa privé* ;

— Soit elle est commune aux deux joueurs, c'est-à-dire que les bits aléatoires d'Alice sont identiques à ceux de Bob ($r_A = r_B$). On parle alors d'*aléa public*.

La complexité d'un protocole π sur une entrée (x, y) dépend donc aussi des choix aléatoires faits. On définit alors simplement $\mathsf{C}(\pi)$ comme la valeur maximale de la complexité de communication de π pour toutes les entrées et choix aléatoires possibles.

Etant donnée une fonction f, les notions de complexités des protocoles probabilistes sont alors définies comme suit :

— Complexité de communication probabiliste à aléa privé :

$$\mathsf{R}_\varepsilon(f) = \min\left\{\mathsf{C}(\pi) : \begin{array}{l}\text{protocole probabiliste } \pi \text{ calculant } f \\ \text{avec aléa privé et erreur bornée } \varepsilon\end{array}\right\};$$

— Complexité de communication probabiliste à aléa public :

$$\mathsf{R}^{\mathsf{pub}}_\varepsilon(f) = \min\left\{\mathsf{C}(\pi) : \begin{array}{l}\text{protocole probabiliste } \pi \text{ calculant } f \\ \text{avec aléa privé } \textbf{et} \text{ public, et erreur bornée } \varepsilon\end{array}\right\}.$$

Ces complexités satisfont

$$\mathsf{R}^{\mathsf{pub}}_\varepsilon(f) \leq \mathsf{R}_\varepsilon(f) \leq \mathsf{D}(f).$$

On procède de façon analogue pour les communications unilatérales probabilistes en définissant $\overrightarrow{\mathsf{R}}^{\mathsf{pub}}_\varepsilon(f)$ et $\overrightarrow{\mathsf{R}}_\varepsilon(f)$.

Exemple 2.1. *Alors que* $\mathsf{D}(\textsc{Equality}_n) = n+1$*, les protocoles 4 et 5 permettent d'établir que*

$$\mathsf{R}(\textsc{Equality}_n) \leq \lceil \log(n/\varepsilon) \rceil + 1 \quad \textit{et} \quad \mathsf{R}^{\mathsf{pub}}_\varepsilon(\textsc{Equality}_n) \leq \lceil \log(1/\varepsilon) \rceil.$$

Ces protocoles renvoient toujours '$x = y$' lorsque $x = y$. Leurs complexités vérifient l'inégalité ci-dessus. Lorsque $x \neq y$, le protocole 4 se trompe avec probabilité au plus ε car le polynôme

Protocole 4 – Protocole à aléa privé pour $\textsc{Equality}_n$

- Alice :
  - Choisir un nombre premier p quelconque tel que $n/\varepsilon \leq p \leq 2n/\varepsilon$
  - Choisir uniformément au hasard un entier a tel que $0 \leq a < p$
  - Calculer $u = \sum_{i=1}^{n} x_i a^{n-i} \mod p$
  - Envoyer $m_1 := (p, a, u)$ à Bob
- Bob :
  - Calculer $v = \sum_{i=1}^{n} y_i a^{n-i} \mod p$
  - Si $u = v$, stopper et déclarer '$x = y$'
  - Sinon stopper et déclarer '$x \neq y$'

Protocole 5 – Protocole à aléa partagé pour $\textsc{Equality}_n$

- Alice et Bob :
  - Choisir $k := \lceil 1/\varepsilon \rceil$ mots $r^1, r^2, \ldots, r^k$ de n bits uniformément au hasard
- Alice :
  - Calculer $u_t = \bigoplus_{i=1}^{n} x_i r_i^t$, pour $t = 1, 2, \ldots, k$
  - Envoyer $m_1 := u$ à Bob
- Bob :
  - Calculer $v_t = \bigoplus_{i=1}^{n} y_i r_i^t$, pour $t = 1, 2, \ldots, k$
  - Si $u = v$, stopper et déclarer '$x = y$'
  - Sinon stopper et déclarer '$x \neq y$'

$P = \sum_{i=1}^{n}(x_i - y_i)a^{n-i}$ s'annule alors en au plus n points (car lorsque p est premier, l'anneau $\mathbb{Z}_p$ est un corps), et que a est pris au hasard dans un ensemble de taille au moins n/ε. Pour le protocole 5, observons que si $w \in \{0,1\}^n$ et $w \neq 0^n$, alors $\bigoplus_{i=1}^{n} w_i r_i = 0$ avec probabilité $1/2$, lorsque r est choisi uniformément au hasard dans $\{0,1\}^n$. Par conséquent, $w := (x_i \oplus y_i)_{i=1,2,\ldots,n}$ satisfait $w \neq 0^n$ lorsque $x \neq y$. De plus $u = v$ si et seulement si $\bigoplus_{i=1}^{n} w_i r_i^t = 0$, pour $t = 1, 2, \ldots, k$. Donc le protocole 5 se trompe avec probabilité $1/2^k \leq \varepsilon$.

Exemple 2.2. *Un autre exemple un peu plus évolué et interactif séparant* D *et* R *est obtenu pour la fonction* $\textsc{GreaterThan}_n$. *Le théorème 1.12 permet d'établir que* $\mathsf{D}(\textsc{GreaterThan}_n) = n + 1$. *Nous verrons que* $\overrightarrow{\mathsf{R}}(\textsc{GreaterThan}_n) = \Omega(n)$. *En revanche que* $\mathsf{R}(\textsc{GreaterThan}_n) = O(\log^2 n)$. *Cette dernière borne est obtenue par un protocole procédant à une recherche dichotomique du plus grand indice i tel que $x[1, i-1] = y[1, i-1]$. Une fois cet indice connu, il suffit en effet pour Alice et Bob et comparer leurs valeurs respectives x_i et y_i. A chaque étape de la recherche dichotomique, Alice et Bob doivent comparer x et y sur un intervalle de positions communes. Cette étape est réalisé en utilisant le protocole 4 où p satisfait $n^2/\varepsilon \leq p \leq 2n^2/\varepsilon$, de sorte que l'accumulation des erreurs est au plus ε.*

Obtenir des minorants de la complexité de communication probabiliste est beaucoup plus complexe. Nous verrons plus loin une méthode générique pour cela. En attendant voici quelques premiers résultats obtenus relativement simplement. Nous verrons plus loin comment supprimer le facteur logarithmique du lemme 2.3.

Lemme 2.3.

$$\overrightarrow{\mathsf{R}}_{\frac{1}{3n}}(\textsc{Index}_n) \geq n, \quad \overrightarrow{\mathsf{R}}_{\frac{1}{3}}(\textsc{Index}_n) \geq n/\log n.$$

Démonstration. La deuxième minoration se déduit de la première comme suit. Etant donné un protocole π pour $\textsc{Index}_n$ avec erreur $1/3$, il suffit d'exécuter $\Theta(\log n)$ fois π, mais avec des choix aléatoires différents, pour obtenir une erreur au plus $1/(3n)$. En effectuant ces répétitions en parallèle le protocole reste uniliatéral.

Pour montrer la première minoration, nous allons montrer qu'il est possible de dérandomiser tout protocole π pour $\textsc{Index}_n$ d'erreur $1/(3n)$, tout en conservant sa complexité. Le lemme 1.4 permet alors de conclure. Soient r_A et r_B les bits aléatoires respectifs d'Alice et Bob. Le protocole déterministe est décrit par le protocole suivant :

- Alice choisit r_A tel que $\Pr_{r_A,r_B}[\exists j \in [n] : P(m(x,r_A),j,r_B) \neq x_j]$ est minimal.
- Alice envoie $t := m(x,r_A)$ à Bob.
- Bob calcule la valeur majoritaire de $out(t,i,r_B)$, lorsque r_B parcourt tous les choix aléatoires possibles.

Par hypothèse, pour tout $x \in \{0,1\}^n$ et $j \in [n]$,

$$\Pr_{r_A,r_B}[out(m(x,r_A),j,r_B) \neq x_j] \leq \frac{1}{3n}.$$

Par sommation (union-bound), on obtient pour tout $x \in \{0,1\}^n$,

$$\Pr_{r_A,r_B}[\exists j \in [n] : out(m(x,r_A),j,r_B) \neq x_j] \leq \frac{1}{3}.$$

Donc l'aléa r_a choisi par Alice et le message $t = m(x,r_A)$ envoyé à Bob satisfont nécessairement

$$\Pr_{r_B}[\exists j \in [n] : out(t,j,r_B) \neq x_j] \leq \frac{1}{3}.$$

Le message t permet de conclure pour toute entrée $i \in [n]$ de Bob. En effet, pour la valeur particulier $j = i$, l'inégalité précédente entraîne

$$\Pr_{r_B}[out(t,i,r_B) \neq x_i] \leq \frac{1}{3}.$$

Ainsi x_i est bien la valeur majoritaire de $out(t,i,r_B)$ lorsque r_B parcourt tous les choix aléatoires possibles de Bob. A noter que cette étape, ainsi que celle du choix de r_A, sont potentiellement longues mais déterministes et ne nécessitent aucune communication. □

3 Applications

3.1 Automates, OBDD et arbres de décision

Lemme 3.1. *Soit L un langage de $\{0,1\}^*$ reconnu par un automate déterministe A à q états. Soit $n \geq 1$ un entier quelconque, et soit $f_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ la fonction définie par $f_n(x,y) = 1$ si $xy \in L$, et $f_n(x,y) = 0$ sinon. Alors*

$$q \geq 2^{\overrightarrow{\mathsf{D}}(f_n)-1}.$$

Démonstration. Supposons donné un automate A pour L à q états. Alors le protocole suivant calcule f_n pour tout $(x,y) \in \{0,1\}^n \times \{0,1\}^n$. Il est de plus déterministe et unilatéral.

- Alice simule l'automate A sur x et envoie l'état courant après la lecture de x à Bob.
- Bob termine la simulation de A en partant de l'état reçu de Bob et en lisant y.
- Bob décide si $xy \in L$ à partir de l'état final de A, et peut donc calculer la valeur $f_n(xy)$.

Puisqu'il suffit de $\lceil \log q \rceil$ à Alice pour transmettre l'état courant de l'automate à Bob, nous avons montré que $\overrightarrow{\mathsf{D}}(f_n) \leq \lceil \log q \rceil \leq 1 + \log q$. □

Exemple 3.2. *La première application du lemme 3.1 est que le langage $L = \{xx : x \in \{0,1\}^*\}$ n'est pas régulier.*

Exemple 3.3. *Une autre application permet de montrer que la déterminisation d'un automate (non-déterministe) nécessite d'accroître exponentiellement le nombre d'états dans certain cas. En effet, le langage $L_n = \{xy : x, y \in \{0,1\}^n, x \neq y\}$ nécessite au moins 2^{n-1} états pour être reconnu par un automate déterministe, alors qu'il est reconnaissable par un automate non-déterministe à $O(n)$ états.*

Il est aussi possible d'appliquer ces techniques pour les arbres de décision, les branching programs et les OBDDs.

3.2 Algorithmes de streaming

Dans un contexte de données massives, les algorithmes de streaming expriment le fait qu'il n'est pas systématiquement possible, pour un programme, de charger la totalité d'une entrée en mémoire vive. Dans une telle situation, l'accès à l'entrée est uniquement séquentiel comme sur un disque dur ou sur une bande, à l'opposé de l'accès direct qu'autorise la mémoire vive ou RAM (random-access memory).

Les algorithmes de streaming à une passe et de mémoire $s(n)$ traitent un flux de données de taille n comme le ferait un automate à $2^{s(n)}$ états. En effet, pour les algorithmes de streaming, une *passe* sur une entrée $w \in \Sigma^n$, pour un alphabet donné Σ, signifie que w est donnée comme un *flux* $w_1, w_2, \ldots, w_n$, qui arrive de façon séquentielle, c'est-à-dire lettre par lettre dans cet ordre.

Pour simplifier, nous supposerons en général que la longueur n est connue en avance. Voici maintenant la définition formelle d'un algorithme de streaming, qu'il soit déterministe ou probabiliste.

Définition 3.4 (Algorithme de streaming)**.** *Un* algorithme de streaming *à $p(n)$ passes et de complexité en espace $s(n)$ est un algorithme qui, étant donné un flux $w \in \Sigma^n$ donné en entrée,*

— *réalise $p(n)$ passes séquentielles sur le flux w ;*

— *utilise une mémoire de taille au plus $s(n)$ bits tout en parcourant w.*

Il est d'usage de définir aussi la complexité en temps d'un algorithme de streaming, c'est-à-dire le temps qu'il met à traiter une lettre du flux avant de lire la suivante. Néanmoins, nous nous attacherons ici uniquement à étudier la complexité en espace, et plus précisément le compromis espace et nombre de passes. Nos bornes inférieures ne s'en retrouveront que plus fortes puisque indépendantes de la complexité en temps des algorithmes. De plus, les algorithmes de streaming que nous construirons peuvent tous être implémentés de sorte que chaque lettre est traitée en temps $\mathrm{polylog}(n)$.

Exemple 3.5. *Etant donné un flux de n entiers deux à deux distincts de $[n+1]$, considérons le problème de déterminer l'unique entier x manquant, en une seule passe et avec mémoire $O(\log n)$.*

Une solution consister à remarquer que la somme S des entiers du flux est simplement la somme des entiers de 1 à $n+1$ moins l'entier x manquant. Le calcul de S se fait avec une mémoire $O(\log n)$ en lisant un à un les entiers du flux. La valeur de x est donc $\frac{(n+1)(n+2)}{2} - S$ à la fin de la lecture du flux.

Considérons maintenant le problème de déterminer dans un flux de n entiers $a_1, a_2, \ldots$ de $[m]$, la plus grande multiplicité de ces entiers. Ce problème revient à calculer le moment de fréquences F^∞. Le calcul des moments de fréquences a été l'une des premières applications des algorithmes de streaming. Le moment F^k d'ordre k est ainsi défini

$$F^k = \sum_{j=1}^{m} (f_j)^k, \quad \text{avec } f_j = |\{i : a_i = j\}|.$$

Ces moments peuvent être efficacement approchés en une passe par des algorithmes de streaming probabilistes avec mémoire polylog(nm) pour $k = 0, 2$ et même loglog(n) pour $k = 1$. En revanche pour $k > 2$ la mémoire requise est au moins de $\Omega(n^{1-2/k})$.

Certains de ces moments (par exemple F_2) a des applications importantes sur le réseau Internet afin de détecter d'éventuelles attaques DoS (denial of service attack). Nous allons plus tard relier le calcul de F^∞ en streaming à celui du calcul d'une autre fonction (INDEX) en complexité de communication.

D'abord, comme pour les automates, la complexité en mémoire d'un algorithme de streaming est simplement reliée à la complexité de communication comme suit.

Lemme 3.6. *Soit f une fonction définie sur $X \times Y$ avec $X = Y = \{0,1\}^n$. Alors tout algorithme de streaming à une passe doit utiliser une mémoire $s(n) \geq \overrightarrow{\mathsf{D}}(f)$ s'il est déterministe, et $s(n) \geq \overrightarrow{\mathsf{R}}_\varepsilon(f)$ s'il est probabiliste et d'erreur ε.*

De plus, tout algorithme de streaming à $p(n)$ passes doit utiliser une mémoire $s(n) \geq \mathsf{D}(f)/p(n)$ s'il est déterministe, et $s(n) \geq \mathsf{R}_\varepsilon(f)/p(n)$ s'il est probabiliste et d'erreur ε.

Ce lemme permet de montrer plusieurs compris mémoire et nombre de passes en streaming. Cependant, comme nous allons le voir, il est parfois nécessaire d'effectuer une réduction plus complexe, afin de transformer la fonction à calculer par un algorithme de streaming en une autre fonction plus adaptée à la complexité de communication.

Lemme 3.7. *Soit A un algorithme de streaming déterministe à une passe et de mémoire $s(n)$ permettant de calculer une valeur v telle que $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$ sur un flux de $n+1$ de $[2n]$. Alors $\overrightarrow{\mathsf{D}}(\textsc{Index}_n) \leq s(n)$.*

De plus, si A est probabiliste avec erreur ε, alors $\overrightarrow{\mathsf{R}}_\varepsilon(\textsc{Index}_n) \leq s(n)$.

Démonstration. En utilisant l'algorithme de streaming A, nous allons construire un protocole π unilatéral pour $\textsc{Index}_n$, où Alice reçoit l'entrée $x \in \{0,1\}^n$, et Bob l'entrée $i \in [n]$. La réduction est déterministe :

- Alice simule l'algorithme A sur le début du stream formé des entiers j tels que $x_j = 1$.
- Alice continue de simuler l'algorithme A sur des entiers $n < i' \leq 2n$ distincts jusqu'à avoir traité exactement n valeurs.
- Alice envoie la mémoire de l'algorithme A à Bob.
- Bob termine la simulation de l'algorithme A sur la fin du stream formée de l'unique entier i, et calcule la valeur v renvoyée par A.
- Si $v \geq 1/3$, alors Bob renvoie $out := 1$; sinon Bob renvoie $out := 0$

Pour prouver que le protocole a la même erreur que l'algorithme A (c'est-à-dire 0 si A est déterministe, et ε sinon), il suffit de remarque que $F^\infty = 1$ lorsque $x_i = 0$, et que $F^\infty = 2$ sinon. Lorsque v satisfait la promesse $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$, on conclue en observant que $v < 1/3$ lorsque $x_i = 0$, et $v > 1/3$ sinon. □

Puisque $\overrightarrow{\mathsf{D}}(\textsc{Index}_n)$ et $\overrightarrow{\mathsf{R}}_\varepsilon(\textsc{Index}_n)$ sont données par les Lemmes 1.4 et 2.3, nous obtenons le résultat suivant. Nous verrons plus loin comment supprimer le facteur logarithmique dû au minorant non optimal du lemme 2.3.

Théorème 3.8. *Tout algorithme de streaming probabiliste à une passe, de mémoire $s(n)$ et erreur $\varepsilon \leq 1/3$ permettant de calculer une valeur v telle que $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$ sur un flux de $n+1$ entiers de $[2n]$ nécessite une mémoire $s(n) \geq n/\log n$.*

Afin de généraliser ce résultat aux passes multiples, une réduction à un autre problème de communication doit être considérée. En effet, $\mathsf{D}(\textsc{Index}_n) = O(\log n)$. Nous considérons donc maintenant une nouvelle réduction à la fonction $\textsc{Disjointness}_n$.

Lemme 3.9. *Soit A un algorithme de streaming déterministe à $p(n)$ passes et de mémoire $s(n)$ permettant de calculer une valeur v telle que $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$ sur un flux de $2n$ entiers de $[3n]$. Alors $\mathsf{D}(\textsc{Disjointness}_n) \leq 2p(n)s(n)$.*

De plus, si A est maintenant probabiliste avec erreur $\varepsilon \leq 1/3$, alors $\mathsf{R}_\varepsilon(\textsc{Disjointness}_n) \leq 2p(n)s(n)$.

Démonstration. La preuve est très similaire à celle du lemme 3.7. La réduction est toujours déterministe :

- Alice simule l'algorithme A sur le début du stream formé des entiers i tels que $x_i = 1$, et complété avec des entiers $n < i' \leq 2n$ distincts afin d'obtenir n valeurs.
- Bob simule l'algorithme A sur la fin du stream formé des entiers i tels que $y_i = 1$, et complété avec des entiers $2n < i' \leq 3n$ distincts afin d'obtenir n valeurs.
- Au total le stream comporte $2n$ entiers de $[3n]$, les joueurs simulant à tour de rôle l'algorithme durant ses $p(3n)$ passes.
- A la fin de la simulation, Alice et Bob calculent la valeur v renvoyée par A.
- Si $v \geq 1/3$, alors ils décident $out := 1$; sinon $out := 0$

Au total $2p(n)s(n)$ bits sont échangés. Pour prouver que le protocole a la même erreur que l'algorithme A, il suffit de remarque que $F^\infty \leq 1$ lorsque $\textsc{Disjointness}_n(x, y) = 0$, et que $F^\infty = 2$ sinon. Lorsque v satisfait la promesse $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$, on conclue en observant que $v < 1/3$ lorsque $\textsc{Disjointness}_n(x, y) = 0$, et $v > 1/3$ sinon. □

Nous obtenons ainsi le résultat suivant à l'aide du lemme 2.3, que nous généraliserons plus loin au cas des algorithmes probabilistes.

Théorème 3.10. *Tout algorithme de streaming déterministe à $p(n)$ passe, de mémoire $s(n)$ et erreur $\varepsilon \leq 1/3$ permettant de calculer une valeur v telle que $|v - \mathrm{F}^\infty| < \mathrm{F}^\infty/3$ sur un flux de $2n$ entiers de $[3n]$ nécessite une mémoire $s(n) \geq (n+1)/2p(n)$.*

3.3 Machines de Turing

Pour une machine de Turing à plusieurs rubans, l'entrée est habituellement écrite sur un ruban à lecture uniquement. L'espace utilisé par la machine est alors l'ensemble des cellules sur lesquelles la machine a écrit. Le temps est défini de manière standard par le nombre de transitions effectuées par la machine. Il est possible d'exprimer des compromis espace-temps à l'aide de la complexité de communication, comme nous allons le voir avec l'exemple du palindrome.

Théorème 3.11. *Soit M une machine de Turing à plusieurs rubans reconnaissant le langage $PAL = \{vv^R : v \in \{0,1\}^*\}$ en temps $T(n)$ et espace $S(n)$ toute entrée de taille $4n$ donnée sur un ruban en lecture uniquement. Alors*

$$T(n) \times S(n) = \Omega(n^2).$$

Démonstration. Considérons une entrée w de la forme $w = x0^{2n}y^R$, où $x, y \in \{0,1\}^n$. Alors $w \in PAL$ si et seulement si $x = y$. Nous allons utiliser M pour construire un protocole déterministe π pour $\textsc{Equality}_n$ de complexité $O(T(n)\times S(n)/n)$. Le résultat découle alors du fait que $\mathsf{D}(\textsc{Equality}_n) = n + 1$. Voici le protocole où Alice reçoit l'entrée $x \in \{0,1\}^n$ et Bob l'entrée $y \in \{0,1\}^n$:

- Alice simule la machine M tant que la tête de lecture n'arrive pas sur un bit de y.
- Bob fait de même tant que la tête de lecture n'arrive pas sur un bit de x.
- Alice commence la simulation, et envoie l'état de M lorsqu'elle doit donner la main à Bob.
- Bob fait de même.
- Le joueur qui termine la simulation de M annonce le résultat à l'autre joueur.

Chaque message nécessite d'envoyer l'état des têtes de lecture de M, soit un nombre constant de bits, ainsi que le contenu des cellules écrites sur chacun des rubans. Par définition $S(n)$ est justement le nombre maximal de cellules sur lesquelles M écrit durant son exécution. Donc il faut au plus au plus $O(S(n))$ bits pour décrire l'état de M.

De plus, un message est envoyé lors que la tête de lecture franchit au moins une fois chacun des $2n$ bits '0' séparant x de y^R. Ce franchissement prend un temps au moins $2n$. Donc au plus $T(n)/(2n)$ messages sont envoyés, en plus du dernier message qui est un simple bit.

Au final, P a donc pour complexité $\mathsf{C}(P) = O(S(n) \times T(n)/n)$.

□

4 Complexité d'information

Jusqu'à maintenant, nous mesurions la complexité de communication par le nombre de bits transmis. Nous allons ici étudier la complexité de communication sous un autre pointe devue à l'aide de la théorie de l'information. Nous utiliserons la notion d'entropie définie par Shanon. Toutes les variables aléatoires seront supposées à support fini.

4.1 Rappels de théorie de l'information

4.1.1 Entropie

Soit une variable aléatoire X. L'entropie de Shanon caractérise la quantité "d'incertitude" que X possède. Intuitivement, si X est uniformément distribuée, alors X est très incertain, alors qu'il n'y a aucune incertitude si X est constante.

Définition 4.1. *L'*entropie *d'une variable X est notée $H(X)$ et définie par :*

$$H(X) := \sum_{x \in \mathrm{supp}(X)} \Pr[X = x] \log \tfrac{1}{\Pr[X=x]} = \mathop{\mathrm{E}}_{x \sim X} \left[\log \tfrac{1}{\Pr[X=x]}\right]$$

Notons U_n une variable aléatoire uniformément distribuée sur n bits, et 0 une variable aléatoire constante de valeur 0. On observe alors que :

$$H(U_n) = \sum_{x \in \{0,1\}^n} \Pr[U_n = x] \log \frac{1}{\Pr[U_n = x]} = \sum_{x \in \{0,1\}^n} 2^{-n} \cdot n = n,$$
$$H(0) = 1 \cdot \log 1 = 0.$$

Ces équations nous disent qu'une chaine uniformément aléatoire de n bits contient n bits d'incertitude, alors qu'une variable aléatoire constante n'a aucune incertitude.

L'entropie vérifie les propriétés suivantes, pour toutes variables aléatoires X, Y (éventuellement corélées) :

Non-négative : $H(X) \geq 0$. Il est donc impossible d'avoir une incertitude "négative".

Sous-additive : $H(XY) \leq H(X)+H(Y)$, avec égalité si et seulement si X et Y sont indépendants. La quantité d'incertitude dans XY est donc au plus la somme des quantités d'incertitude prises séparément dans chacune des variables séparément.

Maximale pour la distribution uniforme : $H(X) \leq \log |S|$, lorsque le support de X est inclus dans un ensemble fini S ($\mathrm{supp}(X) \subseteq S$), avec égalité si et seulement si X est uniforme sur S.

Nous utiliserons aussi parfois l'*entropie binaire*. Pour $p \in [0, 1]$, notons B_p la variable aléatoire de Bernoulli telle que $\Pr[B_p = 1] = p$. Alors nous définissons :

$$h(p) = H(B_p) = p \log \tfrac{1}{p} + (1 - p) \log \tfrac{1}{1-p} \tag{1}$$

4.1.2 Entropie conditionnelle

L'*entropie conditionnelle* mesure la quantité d'incertitude qu'il reste dans une variable aléatoire, malgré la donnée d'une autre variable aléatoire (éventuellement corrélée avec la première).

Définition 4.2. *Soient X, Y deux variables aléatoires (éventuellement corrélées). L'*entropy conditionnelle *de X étant donnée Y est notée $H(X \mid Y)$ et définie par :*

$$H(X \mid Y) := \mathop{\mathrm{E}}_{y \sim Y}[H(X \mid Y = y)] = \sum_{y \in \mathrm{supp}(Y)} \Pr[Y = y] \times H(X \mid Y = y).$$

De manière similaire à l'entropie, l'entropie conditionnelle est toujours non-négative et sous-additive. Elle satisfait de plus les propriétés suivantes, pour toutes variables aléatoires X, Y :

Inégalité Data Processing : $H(X \mid Y) \leq H(X \mid f(Y))$, pour tout fonction f. La quantité d'incertitude ne peut pas décroître lorsque Y est modifié.

Maximale pour des variables indépendantes : $H(X \mid Y) \leq H(X)$, avec égalité si et seulement si X et Y son indépendants. Il s'agit d'une conséquence de la propriété précédente lorsque f est constante.

Chain Rule : $H(XY) = H(X)+H(Y \mid X)$. La quantité d'incertitude dans les variables jointes XY est égale à la somme des quantités d'incertitude dans X et dans Y sachant X.

4.1.3 Information mutuelle

Nous définissons maintenant la notion d'information mutuelle que nous utiliserons pour caractériser la complexité de communication des protocoles.

Définition 4.3. *L'*information mutuelle *entre deux variables aléatoires X, Y est notée $I(X;Y)$ et définie par :*

$$I(X;Y) := H(X) - H(X \mid Y).$$

Intuitivement, cette notion capture l'écart d'incertitude dans X qu'apporte la connaissance de Y, ou encore la quantité d'information que révèle Y sur X.

Cette quantité est symétrique :

$$I(X;Y) = H(Y) - H(Y \mid X).$$

La propriété Chain Rule permet aussi d'avoir une définition alternative :

$$I(X;Y) = H(X) + H(Y) - H(XY).$$

L'information mutuelle peut aussi être définie de manière conditionnelle :

$$I(X;Y \mid Z) := H(X \mid Z) - H(X \mid YZ).$$

Cette dernière notion capture la quantité d'information que révèle Y sur X lorsque Z est déjà connu. Elle vérifie comme précédemment les égalités suivantes :

$$\begin{aligned} I(X;Y \mid Z) &= H(Y \mid Z) - H(Y \mid XZ), \\ I(X;Y \mid Z) &= H(X \mid Z) + H(Y \mid Z) - H(XY \mid Z), \\ I(X;Y \mid Z) &= \mathop{\mathrm{E}}_{z \sim Z}[I(X;Y \mid Z = z)]; \end{aligned}$$

ainsi que les propriétés suivantes, pour toutes variables aléatoires X, Y, Z, V :

Non-négativé : $I(X;Y \mid Z) \geq 0$.

Minimale en cas d'indépendance : $I(X;Y \mid Z) = 0$ si et seulement X, Y sont indépendantes pour chaque valeur possible de Z fixée (indépendance conditionnée à Z).

Majorée par l'entropie : $I(X;Y \mid Z) \leq \min\{H(X \mid Z), H(Y \mid Z)\}$.

Chain rule : $I(X;VY \mid Z) = I(X;V \mid Z) + I(X;Y \mid VZ)$.

Enfin, l'inégalité Data Processing s'exprime de manière analogue pour l'information mutuelle, signifiant qu'il est impossible d'augmenter la quantité d'information en modifiant une variable aléatoire :

Lemme 4.4 (Inégalité Data Processing)**.** *Soient X, Y, Z des variables aléatoires. Pour toute fonction f :*

$$I(f(X,Z);Y \mid Z) \leq I(X;Y \mid Z).$$

4.2 Complexité d'information

Nous sommes maintenant prêts à définir la notion de complexité d'information d'un protocole de communication. Afin d'être le plus général possible, nous considérons maintenant simultanément de l'aléa public et privé. Ainsi chaque joueur dispose d'une source d'aléa public commune à l'autre joueur, et d'une source d'aléa privé qui lui est propre. Cette modification, a priori mineure, est importante pour prouver une borne inférieure pour DISJOINTNESS.

Observer que, du point de vue de la complexité de communication, cet aléa mixte n'apporte rien de plus que l'aléa public, puisqu'Alice et Bob peuvent chacun générer leur aléa privé en utilisant une partie de leur aléa public. Nous utiliserons donc aussi les notations $\overrightarrow{\mathsf{R}}_{\varepsilon}^{\mathsf{pub}}$ et $\mathsf{R}_{\varepsilon}^{\mathsf{pub}}$ pour écrire les complexités de communication correspondantes à aléa public et privé. En revanche, du point de vue de la complexité d'information, le modèle est différent et rend cette complexité plus malléable.

Soit π un protocole de communication à aléa public et privé, et soit μ une distribution sur les entrées des joueurs. Notons XY des entrées distribués aléatoirement (et potentiellement corrélées) selon μ. Soit $XYRM$ la distribution jointe des entées, de l'aléa public R, et de la transcription M du protocole π exécuté sur la paire d'entrées (X, Y) avec l'aléa public R. Observer que l'aléa privé n'est ici pas explicité, et donc que M est une variable aléatoire, même lorsque XYR sont fixés.

Définition 4.5. *La* complexité d'information *d'un protocole π selon une distribution μ est notée $\mathsf{IC}^{\mu}(\pi)$ et définie par :*

$$\mathsf{IC}^{\mu}(\pi) := I(X; M \mid YR) + I(Y; M \mid XR).$$

Si π est unilatéral, alors la complexité d'information unilatérale *selon μ est notée $\overrightarrow{\mathsf{IC}}^{\mu}(\pi)$ et définir par :*

$$\overrightarrow{\mathsf{IC}}^{\mu}(\pi) := I(X; M \mid R).$$

La complexité d'information d'un protocole est alors facilement reliée à la complexité de communication du même protocole comme suit.

Lemme 4.6. *Soient π un protocole et μ une distribution. Alors, $\mathsf{IC}^{\mu}(\pi) \leq \mathsf{C}(\pi)$. Si de plus π est unilatéral, alors $\overrightarrow{\mathsf{IC}}^{\mu}(\pi) \leq \mathsf{C}(\pi)$.*

Preuve (esquisse). Puisque $\mathsf{C}(\pi)$ est un majorant de $H(M)$, il l'est aussi de $I(X; M \mid R)$, $I(X; M \mid YR)$ et $I(Y; M \mid XR)$. La deuxième partie du lemme est donc prouvée, alors que pour la première partie seule un majorant de $2\mathsf{C}(\pi)$ est ainsi obtenu.

Pour se débarrasser du facteur 2, il faut utiliser le fait que chaque message ne dépend que de l'entrée du joueur qui l'envoie, étant donné l'historique des messages précédents. Il s'agit de la propriété rectangle étendue au cas des protocoles probabilistes. Chaque message ne contribue donc qu'à un seul des deux termes qui apparaissent dans la définition de $\mathsf{IC}^{\mu}(\pi)$. Le résultat s'en suit après quelques lignes de calcul. □

4.3 Fonction Index

Nous allons prouver une borne inférieure sur la complexité de communication d'une fonction en fait légèrement plus difficile, appelée la fonction index augmentée AUG-INDEX$_n$ et définie par :

$$\text{AUG-INDEX}_n(x, (x_1, \ldots, x_{k-1}, k)) = x_k.$$

Cette fonction est donc comme la fonction $\textsc{Index}_n$, à l'exception que Bob reçoit aussi les valeurs $x_1, \ldots, x_{k-1}$ en plus de k. A priori ces valeurs ne peuvent pas aider Bob à déterminer x_k si tous les bits de x sont choisis indépendamment. C'est ce que nous allons formaliser en utilisant la théorie de l'information.

Puisque Index est effectivement plus facile que Aug-Index, prouver une borne inférieure sur la complexité de communication de Aug-Index entraîne directement une borne inférieure sur la complexité de communication de Index.

Théorème 4.7. *Pour tout* $0 < \varepsilon < 1/2$,

$$\overrightarrow{\mathsf{R}}_{\varepsilon}^{\mathsf{pub}}(\textsc{Aug-Index}_n) \geq (1 - h(\varepsilon))n.$$

Démonstration. Soit donc π un protocole unilatéral et d'erreur ε pour $\textsc{Aug-Index}_n$ qui minimise $\overrightarrow{\mathsf{R}}_{\varepsilon}^{\mathsf{pub}}(\textsc{Aug-Index}_n)$. D'après la discussion sur l'aléa privé et public, il est possible de choisir π tel que π est à aléa public uniquement. Nous allons minorer la complexité d'information de π pour la distribution μ uniforme sur les entrées x, k, ce qui donnera le résultat souhaité puisque

$$\overrightarrow{\mathsf{R}}_{\varepsilon}^{\mathsf{pub}}(\textsc{Aug-Index}_n) = \mathsf{C}(\pi) \geq \overrightarrow{\mathsf{IC}}^{\mu}(\pi).$$

Observer que la transcription de π consiste en un seul message M d'Alice vers Bob. Soient $XKRM$ les variables aléatoires correspondantes à l'échantillonnage des entrées selon μ, à l'échantillonnage de l'aléa public R, et à la génération du message correspondant M. Alors nous établissons :

$$\begin{aligned}
\overrightarrow{\mathsf{IC}}^{\mu}(\pi) &= I(X; M \mid R) \\
&= \sum_{k=1}^{n} I(X_k; M \mid X_1, \ldots, X_{k-1}) \text{ (chain rule)} \\
&= \sum_{k=1}^{n} I(X_k; M \mid X_1, \ldots, X_{k-1}, k) \\
&= \sum_{k=1}^{n} (H(X_k \mid X_1, \ldots, X_{k-1}, k) - H(X_k \mid M, X_1, \ldots, X_{k-1}, k)) \\
&= n - n \cdot H(X_K \mid M, X_1, \ldots, X_{K-1}, K) \text{ (independance)} \qquad (2)
\end{aligned}$$

Puisque π calcule $\textsc{Aug-Index}_n$, Bob doit pouvoir deviner la valeur X_K. Cela nous suggère que, conditionnée à M, l'entropie de X_K devrait être petite, sinon Bob se tromperait avec une trop grande probabilité. Ce raisonnement est formalisé par l'inégalité de Fano, qui est énoncée uniquement pour le cas des variables booléennes dans le Lemme 4.8.

Pour l'appliquer, observer que M et l'entrée de Bob, $X_1, \ldots, X_{Y-1}, Y$, déterminent la sortie $out(M, X_1, \ldots, X_{Y-1}, Y)$ du protocole (il n'y a pas d'aléa privé dans π). L'inégalité Data Processing nous permet donc d'établir :

$$H(X_Y \mid MX_1, \ldots, X_{Y-1}, Y) \leq H(X_Y \mid out(M, X_1, \ldots, X_{Y-1}, Y)).$$

L'inégalité de Fano (Lemme 4.8 avec $U = X_K$ et $V = out(\ldots)$) nous permet alors de majorer le terme de droite par $h(\varepsilon)$.

En intégrant ceci dans l'Équation 2, nous obtenons l'inégalité suivante qui conclue la preuve :

$$\overrightarrow{\mathsf{IC}}^{\mu}(\pi) \geq n(1 - h(\varepsilon)).$$

□

Voici l'énoncé de l'inégalité de Fano dans le cas booléen, ainsi que sa preuve.

Lemme 4.8 (Inégalité de Fano, cas booléen)**.** *Soient U, V des variables booléennes telles que* $\Pr[U \neq V] \leq \varepsilon$*, pour* $0 < \varepsilon < 1/2$*. Alors* $H(U \mid V) \leq h(\varepsilon)$*.*

Démonstration. Soit E l'évènement '$U \neq V$'. Alors par définition $\Pr[E] \leq \varepsilon$. Puisque UV déterminent conjointement E, et que EV détermine conjointement U, il vient que $I(U; E \mid V) = H(U \mid V) = H(E \mid V)$. De plus conditionner n'augmente pas l'entropie, et donc

$$H(U \mid V) = H(E \mid V) \leq H(E).$$

Puisque E est une variable de Bernoulli qui prend la valeur 1 avec probabilité $p \leq \varepsilon$, et que h est croissante sur l'intervalle $[0, 1/2]$, nous avons $H(E) = h(p) \leq h(\varepsilon)$, ce qui conclut la preuve. □

4.3.1 La fonction GreaterThan

A l'aide de la borne inférieure pour Aug-Index$_n$, nous allons montrer par réduction une borne inférieure pour la fonction GreaterThan$_n$.

Théorème 4.9. $\overrightarrow{\mathsf{R}}_{\varepsilon}^{\mathsf{pub}}(\text{GreaterThan}_n) \geq (1 - h(\varepsilon))n$.

Démonstration. Nous procédons par réduction, et montrons qu'à partir d'un protocole unilatéral pour GreaterThan$_n$, il est possible de construire un protocole unilatéral pour Aug-Index$_n$ de même complexité de communication et de même probabilité d'erreur.

Sur l'entrée x and $(x_1, \ldots, x_{k-1}, k)$ de Aug-Index$_n$, il suffit d'exécuter le protocole pour GreaterThan$_n$ sur l'entrée (x', y'), où $x' = x$ et $y' = (x_1, \ldots, x_{k-1}, 0^{n-k+1})$.

Alors $x' > y'$ (ordre lexicographique) si et seulement si $x_y = 1$. □

4.4 La fonction Disjointness

La fonction Disjointness est au cœur de nombreux résultats en complexité de communication, et nous allons prouver qu'elle a une complexité probabiliste linéaire pour la communication bilatérale en utilisant encore une fois des méthodes de théorie de l'information.

Théorème 4.10. $R_{\varepsilon}^{\text{pub}}(\text{Disjointness}_n) \geq \frac{1-2\sqrt{\varepsilon}}{2} \times n$.

Démonstration. Soit π un protocole d'erreur ε pour Disjointness$_n$ tel que $R_{\varepsilon}^{\text{pub}}(\text{Disjointness}_n) = \mathsf{C}(\pi)$. Encore une fois, nous pouvons supposer que π est sans aléa privé. Nous allons minorer la complexité d'information de π selon une distribution bien choisie, ce qui impliquera le théorème d'après 4.6.

Définissons la distribution μ sur $\{0,1\} \times \{0,1\}$ comme étant uniforme sur les trois éléments $(0,0), (0,1), (1,0)$. Soient X, Y des variables aléatoires sur $\{0,1\}^n \times \{0,1\}^n$ distribuées aléatoirement selon μ^n, c'est-à-dire telles que $(X_i, Y_i) \sim \mu$. Noter que XY sont corrélées et que Disjointness$(X, Y) = 0$ avec probabilité 1. Autrement dit, la distribution μ^n est *facile* pour Disjointness$_n$. Cependant, nous allons montrer que $\mathsf{IC}^{\mu}(\pi)$ est linéaire en n, mais en utilisant néanmoins que π calcule Disjointness$_n$ avec erreur ε sur *toutes les entrées*, incluant celles en dehors du support de μ^n.

Nous allons utiliser un argument de type somme directe pour montrer que π induit un protocole pour AND dont la complexité d'information est d'$1/n$ celle de π. Ici la fonction AND est simplement la fonction logique 'et' entre deux bits.

En général, un résultat de type *somme directe* établit que la quantité de ressources nécessaires pour calculer n instances indépendantes d'une fonction est au moins n fois la quantité nécessaire pour calculer une seule instance. Alors qu'établir une telle propriété pour la complexité de la communication probabiliste est toujours une des questions ouvertes les plus importantes de ce domaine, il est possible de la démontrer pour la complexité d'information.

Ensuite, nous conclurons alors à l'aide d'un résultat (non prouvé ici) qui établit qu'un protocole pour AND requiert une complexité d'information constante strictement positive (pour la distribution μ).

La première partie correspond au Lemme 4.11, et la deuxième au Lemme 4.12. En combinant ces deux lemmes, on obtient que le protocole π satisfait

$$\mathsf{IC}^{\mu^n}(\pi) \geq n \times \mathsf{IC}^{\mu}(\pi') \geq \frac{1-2\sqrt{\varepsilon}}{2} \times n.$$

□

Lemme 4.11. *Soit π un protocole à aléa public et erreur ε pour* $\textsc{Disjointness}_n$*. Alors il existe un protocole π' à aléa public et privé, et erreur ε pour* AND *tel que*

$$\mathsf{IC}^{\mu}(\pi') \leq \frac{1}{n} \times \mathsf{IC}^{\mu^n}(\pi)$$

Démonstration. Nous construisons π' sur toute entrée $(x, y) \in \{0,1\} \times \{0,1\}$ à partir de π comme suit :

1. Avec l'aléa public, Alice et Bob génèrent :
   (a) $j \leftarrow_{\mathrm{R}} [n]$, uniformément au hasard
   (b) $x_1, \ldots, x_{j-1} \sim_\mu X$, indépendamment.
   (c) $y_{j+1}, \ldots, y_n \sim_\mu Y$, indépendamment.
2. Avec son aléa privé, Alice génère $x_{j'} \sim_\mu (X \mid Y = y_{j'})$, pour chaque $j' \in [j+1, n]$.
3. Avec son aléa privé, Bob génère $y_{j'} \sim_\mu (Y \mid X = x_{j'})$, pour chaque $j' \in [1, j-1]$.
4. Alice assigne $x_j \leftarrow x$, et Bob assigne $y_j \leftarrow x$.
5. Alice et Bob simulent le protocole π on $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$, et renvoient la sortie de π.

Pour montrer que π' calcule AND avec erreur ε, observer que chaque $(x_{j'}, y_{j'})$ générés par le protocole, pour $j' \neq j$, le sont selon μ, et donc $x_{j'} \wedge y_{j'} = 0$. Par conséquent,

$$\begin{aligned}&\textsc{Disjointness}((x_1, \ldots, x_{j-1}, x, x_{j+1}, \ldots, x_n), (y_1, \ldots, y_{j-1}, y, y_{j+1}, \ldots, y_n))\\ &= \mathrm{AND}(x, y).\end{aligned}$$

En conclusion, la probabilité d'erreur de π' est au plus celle de π', et est donc majorée par ε.

Nous terminons maintenant la preuve par l'analyse de la complexité d'information de π' dont l'aléa public R correspond aux variables aléatoires $JX_1 \ldots X_{J-1} Y_{J+1} \ldots Y_n$. Par conséquent,

$$\begin{aligned}\mathsf{IC}^{\mu}(\pi') &= I(X; M' \mid YR) + I(Y; M' \mid XR)\\ &= I(X; M \mid YJX_1 \ldots X_{J-1} Y_{J+1} \ldots Y_n)\\ &\quad + I(Y; M \mid XJX_1 \ldots X_{J-1} Y_{J+1} \ldots Y_n)\end{aligned}$$

Concentrons nous sur le premier terme de la somme. Ecrivons X_J pour X et Y_J pour Y, puisque c'est ce que le protocole π' fait. Nous avons alors :

$$\begin{aligned}
& I(X_J; M \mid J X_1 \dots X_{J-1} Y_J \dots Y_n) \\
= \; & \frac{1}{n} \sum_{j=1}^{n} I(X_j; M \mid X_1 \dots X_{j-1} Y_j \dots Y_n) \\
= \; & \frac{1}{n} \sum_{j=1}^{n} H(X_j \mid X_1 \dots X_{j-1} Y_j \dots Y_n) - H(X_j \mid M X_1 \dots X_{j-1} Y_j \dots Y_n) \\
= \; & \frac{1}{n} \sum_{j=1}^{n} H(X_j \mid X_1 \dots X_{j-1} Y_1 \dots Y_n) - H(X_j \mid M X_1 \dots X_{j-1} Y_j \dots Y_n) \\
\leq \; & \frac{1}{n} \sum_{j=1}^{n} H(X_j \mid X_1 \dots X_{j-1} Y_1 \dots Y_n) - H(X_j \mid M X_1 \dots X_{j-1} Y_1 \dots Y_n) \\
= \; & \frac{1}{n} \sum_{j=1}^{n} I(X_j; M \mid X_1 \dots X_{j-1} Y_1 \dots Y_n) \\
= \; & \frac{1}{n} \times I(X_1 \dots X_n; M \mid Y_1 \dots Y_n),
\end{aligned}$$

où la première égalité provient de la définition de l'information mutuelle conditionnée par J ; la deuxième égalite de l'indépendance de X_j avec $Y_1, \dots Y_{j-1}$ lorsque conditionnée par $X_1 \dots X_{j-1} Y_j \dots Y_n$; l'inégalité provient de la non-croissance de l'entropie lorsqu'elle est conditionnée ; et enfin la dernière égalité découle de la propriété Chain Rule.

Un calcul analogue sur le deuxième terme de $\mathsf{IC}^{\mu}(\pi')$ permet alors de conclure la preuve. □

Lemme 4.12. *Tout protocole π' à aléa public et privé qui calcule* AND *avec erreur ε satisfait* $\mathsf{IC}^{\mu}(\pi') \geq \frac{1-2\sqrt{\varepsilon}}{2}$.

Ce lemme requiert un effort significativement plus technique que nous omettons ici.