Mathématiques pour l'informatique 1
notes de cours sur la première partie

L1 – Université Paris-Est, Marne-la-Vallée

Cyril Nicaud

Organisation

- Ce demi-cours est composé de 6 séances de cours et 6 séances de TD, de deux heures chacune.
- Il y a un TD toutes les deux semaines, tout le long du semestre.
- Le cours est en deux parties : 3 séances au début du semestre, et 3 autres au milieu du semestre.
- Pour ne pas rater une séance de cours ou de TD, renseignez-vous au secrétariat ou consultez ma page internet : `http://igm.univ-mlv.fr/~nicaud/L1.html`
- Le cours est sanctionné par un examen à la fin du semestre. La majeure partie des questions, mais pas toutes, ressemblent aux exercices faits en TD.

Le polycopié

- Le polycopié est juste une aide, la référence reste dans tout les cas le cours magistral : si une notion est abordée dans le cours mais n'est pas dans le polycopié, elle peut tomber à l'examen.
- Le polycopié est très factuel, je n'y décris ni les motivations, ni les intuitions, ni les conséquences. Il n'y a presque pas d'exemples non plus : il y a le cours magistral pour ça.
- Le polycopié vient d'être rédigé, il y aura donc inévitablement des coquilles et des erreurs. Si vous en repérez une, même si vous n'êtes pas sûr, envoyez-moi un email que je vérifie : `cyril.nicaud@univ-mlv.fr`
- **Le polycopié couvre les trois premiers cours seulement**, le deuxième polycopié sera disponible plus tard dans le semestre.

CHAPITRE I

Rappels mathématiques

Ce chapitre traite des préliminaires mathématiques sur les ensembles, les applications et les relations. Notions que vous avez déjà vues mais dont on aura besoin (et qu'il faut de toute façon maîtriser quand on fait des mathématiques ou de l'informatique).

I.1 Ensembles

Soit E un ensemble, on note $x \in E$ le fait que x est un élément de E et $x \notin E$ le fait qu'il ne l'est pas. On note $\emptyset$, l'*ensemble vide*, l'unique ensemble qui ne contient aucun élément.

Définition (inclusion) On dit qu'un ensemble E est *inclus* dans un ensemble F, noté $E \subset F$, quand pour tout $x \in E$, on a aussi $x \in F$. On dit alors que E est un *sous-ensemble* de F.

► On a toujours $\emptyset \subset E$ et $E \subset E$.

Théorème I.1 (double inclusion) *Deux ensembles E et F sont égaux si et seulement si $E \subset F$ et $F \subset E$.*

Définition (ensemble des parties) Soit E un ensemble. On note $\mathcal{P}(E)$ l'ensemble de tous les sous-ensembles de E, que l'on appelle l'*ensemble des parties de E* :

$$\mathcal{P}(E) = \{X \mid X \subset E\}.$$

Définition (produit cartésien) Soit E et F deux ensembles. On note $E \times F$ le *produit cartésien* de E et de F, qui est l'ensemble des couples dont la première coordonnée est dans E et la seconde dans F :

$$E \times F = \{(x, y) \mid x \in E \text{ et } y \in F\}.$$

On note habituellement E^2 au lieu de $E \times E$.

► Attention, si $E \neq F$ alors $E \times F \neq F \times E$.

► Similairement, pour un entier $k \geq 2$ et k ensembles $E_1, \ldots E_k$, on note

$E_1 \times E_2 \times \dots E_k$ l'ensemble des k-uplets $(x_1, \dots, x_k)$ tel que pour tout $i \in \{1, \dots k\}$, $x_i \in E_i$.

▶ Pour un entier $k \geq 2$ et un ensemble E, on note E^k l'ensemble

$$\underbrace{E \times \dots \times E}_{k \text{ fois}}.$$

Définition (opérations ensembllistes) Soit A et B deux sous-ensembles de E. On note :

- $\overline{A}$ (ou A^c) le *complémentaire* de A : $\overline{A} = \{x \in E \mid x \notin A\}$.
- $A \cup B$ l'*union* de A et B : $A \cup B = \{x \in E \mid x \in A \text{ ou } x \in B\}$.
- $A \cap B$ l'*intersection* de A et B : $A \cap B = \{x \in E \mid x \in A \text{ et } x \in B\}$.
- $A \setminus B$ la *différence ensembliste* de A par B : $A \setminus B = \{x \in A \mid x \notin B\}$.

▶ L'union et l'intersection sont commutatives et associatives : $A \cup B = B \cup A$ et $(A \cup B) \cup C = A \cup (B \cup C)$. La différence symétrique n'est ni commutative, ni associative.

Théorème I.2 (distributivité) *L'intersection est distributive sur l'union et l'union est distributive sur l'intersection : pour tout $A, B, C \in \mathcal{P}(E)$,*

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C),$$
$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$$

Théorème I.3 (de Morgan) *Pour tout $A, B \in \mathcal{P}(E)$, on a*

$$\overline{A \cap B} = \overline{A} \cup \overline{B},$$
$$\overline{A \cup B} = \overline{A} \cap \overline{B}.$$

▶ On peut faire une intersection ou une union d'une famille (finie ou infinie) d'ensembles : si pour tout $i \in \mathcal{I}$, E_i est un sous-ensemble de E, on définit

$$\bigcup_{i \in \mathcal{I}} E_i = \{x \in E \mid \exists i \in \mathcal{I},\ x \in E_i\},$$
$$\bigcap_{i \in \mathcal{I}} E_i = \{x \in E \mid \forall i \in \mathcal{I},\ x \in E_i\}.$$

Définition (partition) Soit E un ensemble et $P = \{E_i\}_{i \in \mathcal{I}}$ une famille de sous-ensembles de E. On dit que P est une *partition* de E quand P vérifie :

- pour tout $i \in \mathcal{I}$, $E_i \neq \emptyset$;
- pour tout $i, j \in \mathcal{I}$, si $i \neq j$ alors $E_i \cap E_j = \emptyset$;
- $\bigcup_{i \in \mathcal{I}} E_i = E$.

I.2 Fonctions et applications

Définition (fonction) Soient E et F deux ensembles. Une *fonction* f de $E \longrightarrow F$ (E dans F) est un sous-ensemble de $E \times F$ tel que pour tout $x \in E$, il existe au plus un $y \in F$ tel que $(x, y) \in f$. Quand il existe, on note cet y par $f(x)$. L'élément x est alors l'*antécedent* de y et y est l'*image* de x.

Définition (domaine de definition) Le domaine de définition d'une fonction f de $E \longrightarrow F$, noté $\mathrm{Dom}(f)$ est l'ensemble des éléments de $x \in E$ qui ont une image par f.

Définition (application) Une application f de $E \longrightarrow F$ est une fonction de $E \longrightarrow F$ telle que $\mathrm{Dom}(f) = E$. On note F^E l'ensemble des applications de E dans F.

▶ Si f est une fonction de $E \longrightarrow F$, en se restreignant à $\mathrm{Dom}(f)$ on obtient une application de $\mathrm{Dom}(f) \longrightarrow F$.

Définition (classification) Soit f une application de E dans F. On dit que

- f est *injective* quand $\forall x, x' \in E$, $f(x) = f(x') \Rightarrow x = x'$;
- f est *surjective* quand $\forall y \in F$, $\exists x \in E$, $f(x) = y$;
- f est *bijective* quand elle est à la fois injective et surjective.

Définition (identité) Soit E un ensemble. On note Id_E l'*identité* sur E qui est l'application de $E \longrightarrow E$ telle que pour tout $x \in E$, $\mathrm{Id}_E(x) = x$.

Définition (composition) Soient E, F et G trois ensembles, f une application de F^E et g une application de G^F. La *composée* de f par g, notée $g \circ f$ est l'application de $E \to G$ définie pour tout $x \in E$ par $g \circ f(x) = g(f(x))$.

▶ La composition est associative mais pas commutative, même si les ensembles de départ et d'arrivée sont compatibles.

Théorème I.4 (bijections) *Soit E et F deux ensembles. Une application f de $E \longrightarrow F$ est une bijection si et seulement si il existe une application g de $F \longrightarrow E$ telle que $f \circ g = Id_F$ et $g \circ f = Id_E$. On dit alors que g est l'*application réciproque *de f et on la note f^{-1}.*

I.3 Relations binaires

Définition (relation binaire) Soit E un ensemble. Un *relation binaire* $\mathcal{R}$ sur E est un sous-ensemble de $E \times E$. Si $(x, y) \in \mathcal{R}$, on le note $x\mathcal{R}y$.

Définition (classification) Soit $\mathcal{R}$ une relation binaire sur E. On dit que $\mathcal{R}$ est :

- *réflexive* si pour tout $x \in E$, $x\mathcal{R}x$;
- *irréflexive* si pour tout $x \in E$, $x \not\mathcal{R} x$;
- *symétrique* si pour tout $x, y \in E$, $x\mathcal{R}y \Rightarrow y\mathcal{R}x$;
- *antisymétrique* si pour tout $x, y \in E$, $x\mathcal{R}y$ and $y\mathcal{R}x \Rightarrow x = y$;
- *transitive* si pour tout $x, y, z \in E$, $x\mathcal{R}y$ and $y\mathcal{R}z \Rightarrow x\mathcal{R}z$.

Définition (relation d'équivalence) Une *relation d'équivalence* est une relation réflexive, symétrique et transitive.

Définition (classe d'équivalence) Soit $x \in E$ et $\mathcal{R}$ une relation d'équivalence sur E. La *classe d'équivalence* de x, notée $[x]_\mathcal{R}$ est l'ensemble des éléments de E en relation avec x : $[x]_\mathcal{R} = \{y \in E \mid x\mathcal{R}y\}$.

Théorème I.5 *Soit $\mathcal{R}$ une relation d'équivalence sur E. L'ensemble des classes d'équivalences par $\mathcal{R}$ est une partition de E. Réciproquement, si P est une partition de E, la relation "est dans la même part" est une relation d'équivalence sur E.*

Définition (ordre) Un *ordre* est une relation réflexive, antisymétrique et transitive.

Définition (ordre total) Un ordre est dit *total* quand pour tout $x, x' \in E$, $x\mathcal{R}x'$ ou $x'\mathcal{R}x$.

Définition (ordre strict) Un *ordre strict* est une relation irréflexive, antisymétrique et transitive.

CHAPITRE II

Mots et Langages

II.1 Mots

Définition (alphabet) Un *alphabet* est un ensemble fini non vide dont les éléments sont appelés des *lettres*.

Ex : $B = \{0, 1\}$ l'alphabet binaire, $A = \{a, b, c\}$, $L = \{a \cdots, z\}$.

Définition (mot vide) Le symbole ε dénote ce qu'on appelle le *mot vide*. Pour tout ensemble A, on considère par convention que $A^0 = \{\varepsilon\}$, l'ensemble contenant uniquement le mot vide.

▶ Attention, le mot vide n'est pas l'ensemble vide. Et $\{\varepsilon\} \neq \emptyset$ non plus, puisque le premier est de cardinal un et le second de cardinal zéro.

Définition (mot) Pour tout alphabet A, on pose A^* l'ensemble défini par

$$A^* = \bigcup_{n \geq 0} A^n.$$

Un élément de A^* est appelé un *mot* sur l'alphabet A.

▶ Attention l'exposant $*$ n'a pas le même sens que quand en mathématiques on écrit, par exemple, $\mathbb{N}^*$: A^* est l'ensemble des mots sur l'alphabet A alors que $\mathbb{N}^*$ est l'ensemble des entiers naturels privé de l'élément 0. Normalement le contexte rend la notation sans ambiguïté.

▶ Un mot est donc soit le mot vide, soit un n-uplet d'éléments de A. Par exemple, $u = (a, a, b, c, a)$ est un mot de $\{a, b, c\}^*$. Par simplification, on va l'écrire $u = aabca$. C'est juste un changement d'écriture, les propriétés des mots sont les mêmes que celles des n-uplets.

Définition (longueur) Soit $u \in A^*$. La *longueur* de u est l'unique ℓ tel que $u \in A^\ell$. On note $|u|$ la longueur de u. Pour tout $n \in \mathbb{N}$, on note A^n l'ensemble des mots de longueur n.

▶ Soit $u \in A^*$. On note $u = u_1 \dots u_n$ pour dire que u est un mots de longueur n égal au n-uplet

$(u_1, \ldots, u_n)$, où les u_i sont dans A. Par convention $u = \varepsilon$ quand $n = 0$.

► Tout lettre $a \in A$ peut être vue comme un mot de longueur 1 composé d'uniquement cette lettre. Attention, ε n'est pas une lettre.

Théorème II.1 (égalité des mots) *Deux mots $u = u_1 \ldots u_n$ et $v = v_1 \ldots v_m$ sont égaux si et seulement si $n = m$ et pour tout $i \in \{1, \ldots, n\}$, $u_i = v_i$.*

Définition (longueur en a) Pour toute lettre $a \in A$ et pour tout mot $u \in A^*$, on note $|u|_a$ la *longueur en a de u*, qui est le nombre d'occurrences de la lettre a dans le mot u.

II.2 Opérations

Définition (concaténation) Soient $u = u_1 u_2 \cdots u_n$ et $v = v_1 v_2 \cdots v_m$ deux mots sur l'alphabet A. On note $u \cdot v$ (ou encore uv) la *concaténation de u et de v*, qui est le mot de taille $n + m$ défini par :

$$u \cdot v = u_1 \cdots u_n v_1 \cdots v_m.$$

Théorème II.2 *La concaténation sur A^* est associative et admet pour élément neutre le mot vide ε. Elle n'est pas commutative.*

Théorème II.3 *Pour tout $u, v \in A^*$ on a :*

- $|uv| = |u| + |v|$;
- $\forall a \in A,\ |uv|_a = |u|_a + |v|_a$.

Théorème II.4 (simplification) *Si u, v, w sont trois mots tels que $uw = vw$ alors $u = v$. De même, si $wu = wv$ alors $u = v$. On dit qu'on peut* simplifier à gauche et à droite.

► Une opération n'est pas toujours simplifiable, par exemple la composition des applications de $\mathbb{R}$ dans $\mathbb{R}$ ne l'est pas.

Définition (miroir) Le *miroir* d'un mot $u = u_1 \cdots u_n$, noté $\tilde{u}$ est le mot $\tilde{u} = \tilde{u}_1 \cdots \tilde{u}_n$ vérifiant :

$$\forall i \in \{1, \ldots, n\},\ \tilde{u}_i = u_{n+1-i}.$$

Ex : Pour $u = bonjour$ on a $\tilde{u} = ruojnob$.

Théorème II.5 *Pour tout $u \in A^*$, $|\tilde{u}| = |u|$. Pour tout $u \in A^*$, $\tilde{\tilde{u}} = u$. Pour tout $u, v \in A^*$, $\widetilde{uv} = \tilde{v}\tilde{u}$.*

II.3 Découpage des mots

Définition (préfixe) On dit qu'un mot v est un *préfixe* d'un mot u s'il existe un mot $w \in A^*$ tel que $u = vw$.

Définition (suffixe) On dit qu'un mot v est un *suffixe* d'un mot u sur A s'il existe un mot $w \in A^*$ tel que $u = wv$.

Définition (facteur) On dit qu'un mot v est un *facteur* d'un mot u sur A s'il existe deux mots $w, w' \in A^*$ tel que $u = wvw'$.

► Attention dans les définitions ci-dessus, on peut très bien prendre $w = \varepsilon$ et/ou $w' = \varepsilon$.

Définition (sous-mot) Soit $u = u_1 \cdots u_n$ un mot de A^*. Un *sous-mot* v de u est un mot v de longueur m tel qu'il existe une injection strictement croissante ϕ de $\{1, \cdots, m\}$ dans $\{1, \cdots, n\}$ tel que :

$$v = u_{\phi(1)} u_{\phi(2)} \cdots u_{\phi(m)}.$$

▶ Informellement on obtient un sous-mot de u en supprimant des lettres de u et en ne changeant pas l'ordre de celles qui restent. Si $u = informatique$, $v = faq$ est un sous-mot de u, mais pas $w = fini$.

Théorème II.6 *Pour tout $u \in A^*$, u et ε sont toujours des préfixes, des suffixes, des facteurs et des sous-mots de u.*

II.4 Ordres sur les mots

Pour toute cette partie on suppose que l'alphabet A est totalement ordonné (on donne un ordre sur les lettres).

Définition (ordre lexicographique) Soient u et v deux mots de A^* on dit que u est plus petit que v pour l'*ordre lexicographique*, noté $u \leq v$ ou encore $u \leq_{\text{lex}} v$ quand :

- ou bien u est préfixe de v ;
- ou bien il existe $a < b$ dans A (pour l'ordre sur l'alphabet), il existe des mots w, u', v' dans A^*, tels que $u = wau'$ et $v = wbv'$.

▶ C'est l'ordre utilisé pour les dictionnaires.

Définition (ordre militaire) Soient u et v deux mots de A^* on dit que u est plus petit que v pour l'*ordre militaire*, noté $u \leq_{\text{mil}} v$, quand

- ou bien $|u| < |v|$,
- ou bien $|u| = |v|$ et $u \leq_{\text{lex}} v$.

▶ L'ordre lexicographique et l'ordre militaire sont des ordres totaux sur A^*. Les deux ont un élément minimal, qui est le mot vide.

II.5 Langages

Définition (langage) Un *langage* est un ensemble de mots. C'est donc un élément de $\mathcal{P}(A^*)$.

Définition (concaténation de langages) Soient X et Y deux langages, la *concaténation* de X et de Y, notée XY est l'ensemble des concaténations des mots de X par les mots de Y :

$$XY = \{x \cdot y \mid x \in X,\ y \in Y\}.$$

Théorème II.7 *La concaténation des langages est associative mais pas commutative. L'élément neutre est le langage réduit au mot vide $\{\varepsilon\}$.*

Définition (miroir d'un langage) Soit X un langage, le *miroir* noté $\widetilde{X}$ est l'ensemble des miroirs des mots de X :

$$\widetilde{X} = \{\tilde{x} \mid x \in X\}.$$

Définition (puissances) Par convention, pour tout langage X on pose $X^0 = \{\varepsilon\}$. On définit les *puissances d'un langage X*, pour tout $n \geq 1$ par

$$X^n = X^{n-1} X.$$

▶ Pour $n \geq 1$, $u \in X^n$ si et seulement si il peut s'écrire comme concaténation de n mots de X.

Définition (étoile) Soit X un langage, l'*étoile* de X (on dit aussi *l'étoile de Kleene de* X) est le langage

$$X^* = \bigcup_{n \geq 0} X^n.$$

▶ Le langage X^* contient toujours le mot vide.

CHAPITRE III

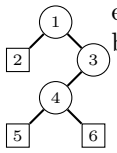
Codes, codages et compression

III.1 Arbres binaires

On reste informels dans cette partie. La formalisation mathématique de la notion d'arbre est bien entendu possible, mais n'apporte rien dans le cadre de ce cours, où on se sert des arbres pour décrire l'algorithme de compression de Huffman.

▶ Un *arbre binaire* $\mathcal{B}$ est composé de *nœuds*. Il y a un *nœud* particulier appelé la racine de $\mathcal{B}$. Tout nœud de $\mathcal{B}$ qui n'est pas la racine est soit le *fils gauche* soit le *fils droit* d'un autre nœud $\mathcal{B}$. La racine n'est fils d'aucun nœud. On demande de plus que tout nœud soit atteignable depuis la racine : si on prend la clotûre transitive de la relation "est fils de" sur les nœuds de $\mathcal{B}$, tout nœud est en relation avec la racine.

▶ Si un nœud a au moins un fils on dit que c'est un *nœud interne*, sinon on dit que c'est une *feuille*.



On représente la racine en haut et les feuilles en bas.

- 1 est la racine ;
- 2, 5 et 6 sont les feuilles ;
- 1, 3 et 4 sont des nœuds internes.

Définition (forêt) Une *forêt* est un ensemble d'arbres.

III.2 Codes

Définition (code) Un sous-ensemble non-vide X de A^* est un *code* quand tout mot de A^* s'écrit d'au plus une façon comme concaténation de mots de X : X est un code si et seulement si quand on a

$$x_1 \cdots x_n = y_1 \cdots y_m,$$

où les x_i et les y_i sont des mots de X, alors $n = m$ et pour tout $i \in \{1, \ldots n\}$, $x_i = y_i$.

▶ On n'impose pas que tout mot soit représentable comme concaténation de mots de X, mais que s'il l'est alors il l'est d'une unique façon.

Définition (code de longueur fixe) Soit X un langage qui n'est ni vide ni réduit au mot vide. Si tous les mots de X sont de même longueur $k \geq 1$, on dit que X est un *code de longueur fixe.*

Théorème III.1 *Un code de longueur fixe est un code.*

Ex : ASCII, UNICODE, code bitmap pour les images, *etc.* sont des codes de longueur fixe.

Définition (code préfixe) Un langage X non vide et non réduit au mot vide est un *code préfixe* quand aucun mot de X n'est préfixe d'un autre mot de X.

Théorème III.2 *Un code préfixe est un code.*

▶ Attention à la terminologie, on devrait dire "code sans préfixes".

▶ Les codes préfixes sont très utilisés pour faire de la compression de données.

Définition (code suffixe) Un langage X non vide et non réduit au mot vide est un *code suffixe* quand aucun mot de X n'est suffixe d'un autre mot de X.

Théorème III.3 *Un code suffixe est un code.*

▶ Il existe des langages qui ne sont dans aucune des catégories précédentes mais qui sont pourtant des codes.

III.3 Codage

On note $B = \{0, 1\}$ l'alphabet binaire.

Définition (codage) Un codage est une application injective de A^* dans B^*.

▶ Cette définition est trop générale en pratique, on ne peut pas stocker toutes les images de tous les mots.

Théorème III.4 *Soit $C \subset B^*$ un code et soit φ une application injective de A dans C. On étend φ (par morphisme) à tout $u \in A^*$ par $\varphi(\varepsilon) = \varepsilon$ et pour tout $u = u_1 \cdots u_n$ mot de A^* de longueur $n \geq 1$,*

$$\varphi(u) = \varphi(u_1) \cdots \varphi(u_n).$$

Alors φ est un codage.

III.4 Huffman

▶ L'idée de la compression de donnée est de représenter en machine un mot u de A^* par un codage φ et un mot v sur B^*, tels que $\varphi(u) = v$.

▶ Pour que cela soit efficace, il faut que

- l'encodage se calcule rapidement ;
- la représentation de (φ, v) soit compacte ;
- le décodage, pour retrouver u, se fasse rapidement.

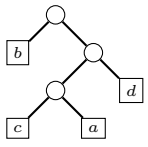
► L'algorithme de Huffman permet de réaliser un tel encodage en utilisant le théorème III.4 : on va construire un code préfixe C et le codage φ à partir du mot à compresser u, puis on en déduira v.

► **1. Initialisation :** on crée une forêt avec un arbre pour tout $a \in A$. Chacun de ces arbres est réduit à juste une racine. Le *poids* de la racine de l'arbre a est $|u|_a$.

► **2. Construction de l'arbre :** tant qu'il y a plus d'un arbre dans la forêt on répète l'opération suivante : sélectionner les deux arbres $\mathcal{A}$ et $\mathcal{B}$ qui ont les plus petits poids à la racine $p_{\mathcal{A}}$ et $p_{\mathcal{B}}$. Les changer en un seul arbre $\overset{\circ}{\underset{\mathcal{A}}{/}}\!\!\underset{\mathcal{B}}{\backslash}$, de poids $p_{\mathcal{A}} + p_{\mathcal{B}}$ à la racine.

► **3. Codage pour les lettres :** une fois l'arbre terminé, les lettres sont sur les feuilles. A chaque lettre $a \in A$, on associe le mot binaire $\varphi(a)$ obtenu en suivant le chemin allant de la racine à la feuille a, en mettant, de gauche à droite, un 0 quand on va à gauche et un 1 quand on va à droite.

Ex :



$\varphi(a) = 101$
$\varphi(b) = 0$
$\varphi(c) = 100$
$\varphi(d) = 11$

► La taille finale obtenue, $|\varphi(u)|$ vérifie :

$$|\varphi(u)| = \sum_{a \in A} |\varphi(a)| \cdot |u|_a.$$

► Si on utilise un code de longueur fixe, le mieux qu'on puisse faire c'est utiliser k bits par symbole, où k est l'unique entier tel que $2^{k-1} < |A| \leq 2^k$.