Mathématiques Discrètes 1
& Informatique Théorique

ESIAL 1^{ere} année

Livret pédagogique : cours et exercices d'entrainement

Tony Bourdier
Version 9.0

École Supérieure d'Informatique et Applications de Lorraine
193 avenue Paul Muller
CS 90172 – 54602 Villers-lès-Nancy

tony.bourdier@loria.fr
www.tony-bourdier.fr

Table des matières

Première partie

Fondements, algèbre de Boole et théorie des langages

Chapitre 1

Fondements

1.1 Raisonnement par récurrence

1.1.1 Définition

Le raisonnement par récurrence est une forme de raisonnement visant à démontrer une propriété portant sur tous les entiers naturels, ou plus généralement sur tous les entiers supérieurs à un entier donné. Le raisonnement par récurrence simple consiste à démontrer séparément les points suivants :

- Une propriété est vérifiée par un entier k (par exemple 0 ou 1) : c'est l'**initialisation** de la récurrence
- Si la propriété est vérifiée par un entier, alors elle doit être vérifiée par son successeur, c'est-à-dire, l'entier qui le suit : il s'agit de l'**hérédité**.

Une fois cela établi, on en déduit que la propriété est vraie pour tous les entiers supérieurs à k.

Remarque 1.1 : Le raisonnement par récurrence établit une propriété importante liée à la structure des entiers naturels : celle d'être construits à partir de 0 en itérant le passage au successeur.

On peut formaliser le raisonnement par récurrence simple de la façon suivante :

Définition 1.2 : Raisonnement par récurrence simple :
Soit $\mathcal{P}$ une propriété définie sur $\mathbb{N}$:

$$\begin{array}{rl} \text{si} & \begin{cases} \mathcal{P}(0) \\ \forall n \in \mathbb{N},\ \mathcal{P}(n) \Rightarrow \mathcal{P}(n+1) \end{cases} \\ \text{alors} & \forall n \in \mathbb{N},\ \mathcal{P}(n) \end{array}$$

1.1.2 Règles de bonne conduite

Pour démontrer une propriété $\mathcal{P}$ pour tout $n \in \mathbb{N}$ par récurrence, on procède comme suit :

- On dit que l'on démontre $\mathcal{P}$ par récurrence sur n
- On démontre $\mathcal{P}(0)$
- On suppose que $\mathcal{P}(n)$ est vraie, c'est l'**hypothèse de récurrence** (**H.R.**), et l'on démontre que, sous cette hypothèse de récurrence, $\mathcal{P}(n+1)$ est vraie.
- On conlue proprement ! Autrement dit, on conclut par une phrase de la forme « La propriété est vraie pour 0 et est héréditaire, elle est donc vraie pour tout $n \in \mathbb{N}$ » ou encore « $\mathcal{P}(0)$ est vraie et $\forall n,\ \mathcal{P}(n) \Rightarrow \mathcal{P}(n+1)$ donc $\forall n \in \mathbb{N},\ \mathcal{P}(n)$ ».

1.1.3 Exemple

On cherche à démontrer que $\forall n \in \mathbb{N}^*,\ n^2 = \sum_{k=1}^{n}(2k-1)$.

On note $\mathcal{P}$ la propriété définie par $\mathcal{P}(n)$: « $n^2 = \sum_{k=1}^{n}(2k-1)$ »

- $\mathcal{P}(1)$ est vraie car $\sum_{k=1}^{1}(2k-1) = 2-1 = 1 = 1^2$
- Soit $n \geq 1$, on suppose (hypothèse de récurrence) que $\mathcal{P}(n)$ est vraie et on souhaite montrer que sous cette hypothèse $\mathcal{P}(n+1)$ est vraie. On a :

$$\begin{aligned}\sum_{k=1}^{n+1}(2k-1) &= \sum_{k=1}^{n}(2k-1) + (2(n+1)-1)\\ &= \sum_{k=1}^{n}(2k-1) + 2n + 1\\ &= \text{(H.R.) } n^2 + 2n + 1\\ &= (n+1)^2\end{aligned}$$

Donc $\forall n \geq 1,\ \mathcal{P}(n) \Rightarrow \mathcal{P}(n+1)$

La propriété $\mathcal{P}$ est vraie au rang 1 et se transmet du rang n au rang $n+1$, elle est donc vraie pour tout $n \geq 1$.

1.1.4 Récurrence d'ordre k

1.1.4.1 Récurrence d'ordre 2

Définition 1.3 : Raisonnement par récurrence d'ordre 2 :
Soit $\mathcal{P}$ une propriété définie sur $\mathbb{N}$:

$$\begin{array}{rl}\text{si} & \left\{\begin{array}{l}\mathcal{P}(0)\\ \mathcal{P}(1)\\ \forall n \in \mathbb{N},\ (\mathcal{P}(n-1) \wedge \mathcal{P}(n)) \Rightarrow \mathcal{P}(n+1)\end{array}\right.\\ \text{alors} & \forall n \in \mathbb{N},\ \mathcal{P}(n)\end{array}$$

1.1.4.2 Récurrence d'ordre k

Définition 1.4 : Raisonnement par récurrence d'ordre k :
Soit $\mathcal{P}$ une propriété définie sur $\mathbb{N}$:

$$\begin{array}{rl}\text{si} & \left\{\begin{array}{l}\bigwedge_{i=0}^{k-1}\mathcal{P}(i)\\ \forall n \in \mathbb{N},\ \left(\bigwedge_{i=0}^{k-1}\mathcal{P}(n-i)\right) \Rightarrow \mathcal{P}(n+1)\end{array}\right.\\ \text{alors} & \forall n \in \mathbb{N},\ \mathcal{P}(n)\end{array}$$

Remarque 1.5 : Il faut bien prendre garde au fait que l'initialisation d'une démonstration par récurrence d'ordre $k > 1$ n'est pas constituée uniquement de la vérification de $\mathcal{P}(0)$ mais de $\mathcal{P}(0)$ et $\mathcal{P}(1)$ et ... et $\mathcal{P}(k-1)$!

1.1.5 Récurrence forte

Il est parfois nécessaire, dans des raisonnements par récurrence, d'utiliser une version plus forte pour l'hérédité.

Définition 1.6 : Raisonnement par récurrence forte :
Soit $\mathcal{P}$ une propriété définie sur $\mathbb{N}$:

$$\begin{array}{rl} \text{si} & \left\{ \begin{array}{l} \mathcal{P}(0) \\ \forall n \in \mathbb{N},\ (\forall k \leq n,\ \mathcal{P}(k)) \Rightarrow \mathcal{P}(n+1) \end{array} \right. \\ \text{alors} & \forall n \in \mathbb{N},\ \mathcal{P}(n) \end{array}$$

1.2 Induction

1.2.1 Ensembles définis inductivement

On peut généraliser le principe des preuves par récurrence à des ensembles autres que $\mathbb{N}$. Il suffit ces ensembles soient inductifs, c'est-à-dire des ensembles pour lesquels on a un moyen de construction (des eléments de « base » et des « constructeurs ») :
Soit E un ensemble. On défini inductivement un sous-ensemble X de E lorsque l'on se donne des règles de construction des éléments de X, règles que l'on sépare en deux types de règles :

i) **les règles de bases** : qui indiquent les éléments qui sont dans X

ii) **les règles inductives** : qui donnent un moyen de construire les éléments de X à partir de ceux déjà construits.

> **Exemple 2.1 :** On considère l'ensemble E des suites finies, non vides, de 0 et de 1. $E = \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$. On définit inductivement l'ensemble X_d par :
>
> *i*) 1 est dans X_d
>
> *ii*) si x est dans X_d alors la suite x suivie de 0 est dans X_d
>
> X_d est alors l'ensemble des suites finies composées d'un 1 suivi de 0 : $\{1, 10, 100, \ldots\}$

Plus formellement :

Définition 2.2 : Soit E un ensemble, une définition inductive d'une partie X et E est la donnée :

i) d'une partie B de E

ii) d'un ensemble $\mathcal{O}p$ d'opérations sur les éléments de E

X est alors le plus petit ensemble vérifiant :

i) **base** : tous les éléments de B appartiennent à X ($B \subset X$)

ii) **induction** : pour toute opération ϕ de $\mathcal{O}p$ d'arité $m \in \mathbb{N}$ et $x_1, \ldots, x_m \in X$:

$$\phi(x_1, \ldots, x_m) \in X$$

Remarque 2.3 : L'ensemble de départ E vérifie les deux étapes de base et d'induction mais n'est pas nécessairement le plus petit. Donc X est le plus petit sous-ensemble de E contenant B et clos (= fermé = stable) pour les opérations de $\mathcal{O}p$.

Exemple 2.4 : X_d peut être défini inductivement par :

i) base : $B = \{1\}$

ii) induction : $\mathcal{O}p = \{AjouteZero : x \mapsto x.0\}$

X_d est alors le plus petit ensemble vérifiant

i) base : 1 dans X_d

ii) induction : si x dans X_d alors $AjouteZero(x) = x.0$ est dans X_d.

Exemple 2.5 : L'ensemble des expressions logique est le plus petit ensemble noté $\mathcal{E}$ tel que :

i) base : $true$ et $false \in \mathcal{E}$

ii) induction : $\forall e, e_1, e_2 \in \mathcal{E}$: $\begin{cases} e_1 + e_2 \\ e_1 \times e_2 \\ \neg e \end{cases} \in \mathcal{E}$

soit $\mathcal{B} = \{true, false\}$ et $\mathcal{O}p = \{+, \times, \neg\}$.

Théorème 2.6 : Tout élément d'un ensemble défini inductivement peut s'obtenir à partir d'éléments de base en appliquant un nombre fini d'étapes inductives.

▷ **Démonstration :** *Il suffit de considérer la suite d'ensembles suivants :*

i) $X_0 = B$

ii) $X_{n+1} = X_n \cup \{\phi(x_1, \ldots, x_k) \mid x_1, \ldots, x_k \in X_n \text{ et } \phi \in \mathcal{O}p\}$

On montre alors que $X = \bigcup_{n \geq 0} X_n$. *Pour tout élément* x *de* X, *il existe* $n \in \mathbb{N}$ *tel que* $x \in X_n$. *Le plus petit des* n *tels que* $x \in X_n$ *donne le nombre d'étapes inductives qu'il faut appliquer aux éléments de base pour obtenir* x. ■

Remarque 2.7 : Pour chaque élément d'un ensemble X construit inductivement, on peut dresser un arbre de construction dont :
- les feuilles sont des éléments de la base B
- les noeuds sont des éléments de $\mathcal{O}p$
- la hauteur[1] sera le nombre d'étapes inductives à appliquer pour obtenir cet élément

1.2.2 Fonctions définies inductivement

Définition 2.8 : Soit X un ensemble défini inductivement à partir de $(B, \mathcal{O}p)$. Définir une fonction $f : X \to F$ inductivement consiste à :

i) **base** : définir les valeurs $f(x) \in F$ pour tout $x \in B$

[1] En considérant qu'une feuille est un arbre de hauteur 0.

ii) **induction** : pour toute opération $\phi \in \mathcal{O}p$ d'arité n, pour tout $x_1, \ldots, x_n \in X$, exprimer $f(\phi(x_1, \ldots, x_n))$ en fonction de $f(x_1), \ldots, f(x_n)$ (et éventuellement de $x_1, \ldots, x_n$ et $\phi(x_1, \ldots, x_n)$).

Exemple 2.9 : On considère l'ensemble $\mathcal{E}$ de l'exemple 2.5. On définit la fonction $eval : \mathcal{E} \to \{0, 1\}$ de la façon suivante :

i) base : $eval(true) = 1$, $eval(false) = 0$

ii) induction : $\forall e, e_1, e_2 \in \mathcal{E}$: $\begin{cases} eval(e_1 + e_2) = max(eval(e_1), eval(e_2)) \\ eval(e_1 \times e_2) = min(eval(e_1), eval(e_2)) \\ eval(\neg e) = 1 - eval(e) \end{cases}$

Exemple 2.10 : Soit $Liste$ l'ensemble défini inductivement de la façon suivante :

- *base :* $\forall x \in \mathbb{N},\ x \in Liste$
- *induction :* $\forall q \in Liste, x \in \mathbb{N}$:

$$x :: q \in Liste$$

On veut définir les fonctions $Maximum : Liste \to \mathbb{N}$ et $Somme : Liste \to \mathbb{N}$ qui retournent respectivement le maximum et la somme des éléments d'une liste.

- $Maximum : Liste \to \mathbb{N}$ est défini inductivement par :
  - *i*) base : $\forall x \in \mathbb{N},\ Maximum(x) = x$
  - *ii*) induction : $\forall q \in Liste, x \in \mathbb{N} :\ Maximum(x :: q) = max(x, Maximum(q))$
- $Somme : Liste \to \mathbb{N}$ est défini inductivement par :
  - *i*) base : $\forall x \in \mathbb{N},\ Somme(x) = x$
  - *ii*) induction : $\forall q \in Liste, x \in \mathbb{N} :\ Somme(x :: q) = x + Somme(q)$

1.2.3 Preuve par induction structurelle

Il s'agit d'une généralisation de la preuve par récurrence.

Théorème 2.11 : Soit X un ensemble défini inductivement à partir d'une base B et d'un ensemble d'opérateurs $\mathcal{O}p$. On veut démonter une propriété $\mathcal{P}$ vraie pour tout élément de X, autrement dit : $\forall x \in X,\ \mathcal{P}(x)$. Si les deux conditions sont vérifiées :

i) **base** : $\mathcal{P}(x)$ est vraie pour tout x de B

ii) **induction** : pour tout ϕ de $\mathcal{O}p$ d'arité k, pour tout $x_1, \ldots, x_k \in X$, si $\mathcal{P}(x_1), \ldots, \mathcal{P}(x_k)$ sont vraies, alors $\mathcal{P}\big(\phi(x_1, \ldots, x_k)\big)$ est vraie.

alors $\mathcal{P}$ vraie pour tout élément de X.

Remarque 2.12 : On s'aperçoit alors que la preuve par récurrence sur un entier $n \in \mathbb{N}$ est une preuve par induction structurelle sur l'ensemble des entiers que l'on défini inductivement par :

i) **base** : $B = \{0\}$

ii) **induction** : $\mathcal{O}p = \{succ\}$ où $succ : n \mapsto n+1$

Exemple 2.13 : On reprend l'exemple 2.9. On souhaite démontrer que $\forall e \in \mathcal{E},\ eval(e) \in \{0,1\}$. On note $\mathcal{P}(e)$ la propriété « $eval(e) \in \{0,1\}$ ».

i) base : $eval(true) = 1 \in \{0,1\}$ et $eval(false) = 0 \in \{0,1\}$ donc pour tous les éléments e de la base, $\mathcal{P}(e)$ est vraie.

ii) induction : soient $e, e_1, e_2 \in \mathcal{E}$ tels que $\mathcal{P}(e),\ \mathcal{P}(e_1)$ et $\mathcal{P}(e_2)$ soient vérifiées. On cherche à vérifier $\mathcal{P}(e_1 + e_2)$, $\mathcal{P}(e_1 \times e_2)$ et $\mathcal{P}(\neg e)$.

- $eval(e_1 + e_2) = max(eval(e_1), eval(e_2))$. Or, par hypothèse d'induction, $eval(e_1) \in \{0,1\}$ et $eval(e_2) \in \{0,1\}$. Le maximum de deux éléments de $\{0,1\}$ appartient aussi à $\{0,1\}$, donc $\mathcal{P}(e_1 + e_2)$ est vérifiée.
- $eval(e_1 \times e_2) = min(eval(e_1), eval(e_2))$. Or, par hypothèse d'induction, $eval(e_1) \in \{0,1\}$ et $eval(e_2) \in \{0,1\}$. Le minimum de deux éléments de $\{0,1\}$ appartient aussi à $\{0,1\}$, donc $\mathcal{P}(e_1 \times e_2)$ est vérifiée.
- $eval(\neg e) = 1 - eval(e)$. Par hypothèse d'induction, $eval(e) \in \{0,1\}$. Deux cas sont possibles :
  - $eval(e) = 1$ et ainsi $eval(\neg e) = 0 \in \{0,1\}$
  - $eval(e) = 0$ et ainsi $eval(\neg e) = 1 \in \{0,1\}$

  donc quelque soit e vérifiant $\mathcal{P}$, $\neg e$ vérifie $\mathcal{P}$.

Conclusion : la propriété est vérifiée pour les éléments de la base et est stable pour les opérations d'induction, elle est donc vraie pour tous les éléments de $\mathcal{E}$.

1.3 Travaux Dirigés

1.3.1 Exercice 1

Démontrez par récurrence que $\forall n \in \mathbb{N}$,

$$\sum_{i=0}^{n} q^i = \frac{1-q^{n+1}}{1-q} \quad \text{et} \quad \sum_{i=0}^{n}(2i+1) = (n+1)^2$$

1.3.2 Exercice 2

Soit *Liste* l'ensemble défini inductivement de la façon suivante :

- *Base :* $\forall x \in \mathbb{N},\ x \in Liste$
- *Induction :* $\forall q \in Liste, x \in \mathbb{N}$:

$$x :: q \in Liste$$

Soit $\lambda \in Liste$, on note $taille(\lambda)$ le nombre d'élements de λ et $queue(\lambda)$ la queue de λ.
Exemples : $taille(2) = 1,\ taille(4 :: 1 :: 2) = 3,\ queue(2) = 2,\ queue(4 :: 1 :: 2) = 2$.

1) Donnez une définition inductive de *taille*
2) Donnez une définition inductive de *queue*

1.3.3 Exercice 3

Soit *Pile* l'ensemble défini inductivement de la façon suivante :

- *Base :* $\forall x \in \mathbb{N},\ x \in Pile$

- *Induction :* $\forall q \in Pile, x \in \mathbb{N}$:

$$x \uparrow q \in Pile$$

Soit $\lambda \in Pile$, on note $prof(\lambda)$ le nombre d'élements contenus dans λ et $fond(\lambda)$ le fond de λ.
Exemples : $prof(2) = 1,\ prof(4 \uparrow 1 \uparrow 2) = 3,\ fond(2) = 2,\ fond(4 \uparrow 1 \uparrow 2) = 2.$

1) Donnez une définition inductive de $prof$
2) Donnez une définition inductive de $fond$

1.3.4 Exercice 4

On appelle ensemble des *arbres binaires* et on note AB le plus petit ensemble défini par :
- *Base :* $\forall n \in \mathbb{N},\ n \in AB$
- *Induction :* $\forall n \in \mathbb{N},\ \forall g, d \in AB,\ (g, n, d) \in AB$

Ce que l'on note également :

$$AB\ :=\ \mathbb{N}\ \mid\ (AB, \mathbb{N}, AB)$$

1) Pour tout élément a de AB, on note $h(a)$ la hauteur de a.
Exemple : $h(5) = 1,\ h\big(((4,2,3),1,5)\big) = 3,\ h\big((1,5,2),2,((4,2,3),1,5)\big) = 4$
Donnez la définition inductive de h.
2) Pour tout élément a de AB, on note $n(a)$ le nombre de noeuds constituant a.
Exemple : $n(5) = 1,\ n\big(((4,2,3),1,5)\big) = 5,\ n\big((1,5,2),2,((4,2,3),1,5)\big) = 9$
Donnez la définition inductive de n.
3) Montrez que $\forall k_1, k_2 \in \mathbb{N}$:

$$2^{k_1} + 2^{k_2} \leq 2.2^{max(k_1,k_2)}$$

et en déduire que

$$2^{k_1} + 2^{k_2} - 1 \leq 2^{max(k_1,k_2)+1} - 1$$

4) Montrez par induction structurelle que $\forall a \in AB$:

$$h(a) \leq n(a) \leq 2^{h(a)} - 1$$

1.3.5 Exercice 5

Soient $\mathcal{F}$ un ensemble de symboles de fonction d'arité quelconque et $\mathcal{X}$ un ensemble de variables. On définit inductivement l'ensemble noté $\mathcal{T}(\mathcal{F}, \mathcal{X})$ par :
- *Base :* $\forall x \in \mathcal{X},\ x \in \mathcal{T}(\mathcal{F}, \mathcal{X})$
- *Induction :* $\forall f \in \mathcal{F}$ d'arité n, $\forall t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X}) : f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$

1) Listez quelques éléments de $\mathcal{T}(\mathcal{F}, \mathcal{X})$ lorsque $\mathcal{F} = \{f, g, h, i\}$ et $\mathcal{X} = \{x, y, z\}$ où f, g, h, i sont d'arité respective $1, 2, 4$ et 0.
2) Soit $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ une fonction définie par : $\begin{cases} \sigma(x) = i \\ \sigma(y) = f(z) \\ \sigma(z) = z \end{cases}$
On définit inductivement la fonction $\sigma^* : \mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ de la façon suivante :
   - *Base :* $\forall x \in \mathcal{X},\ \sigma^*(x) = \sigma(x)$
   - *Induction :* $\forall f \in \mathcal{F}$ d'arité n, $\forall t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$:

$$\sigma^*(f(t_1, \ldots, t_n)) = f(\sigma^*(t_1), \ldots, \sigma^*(t_n))$$

   Évaluez $\sigma^*\big(h(x, i, f(y), z)\big)$.
3) $\forall t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, on note respectivement $\mathcal{V}ar(t)$ et $\mathcal{F}un(t)$ l'ensemble des variables et celui des symboles de fonction intervenant dans t. Donnez une définition inductive des fonctions $\mathcal{V}ar$ et $\mathcal{F}un$.

Chapitre 2

Algèbre de Boole

2.1 Booléens

Définition 1.1 : On appelle **booléen** tout élément de $\mathbb{B} = \{0, 1\}$.

Définition 1.2 : Soit $x \in \mathbb{B}$, on note $\overline{x}$ le booléen défini par :

$$\overline{x} = \begin{cases} 1 & si\ x = 0 \\ 0 & si\ x = 1 \end{cases}$$

Définition 1.3 : On appelle $\boldsymbol{n}-$**uplet de booléens** tout élément de $\mathbb{B}^n$, c'est à dire tout $n-$uplet :

$$(x_1, \ldots, x_n)$$

tel que $\forall i \in [\![1, n]\!],\ x_i \in \mathbb{B}$.

Remarque 1.4 : Il exists 2^n $n-$uplets différents de booléens : $card\,(\mathbb{B}^n) = 2^n$:
- $\mathbb{B}^1 = \mathbb{B} = \{0, 1\}\ :\ card\left(\mathbb{B}^1\right) = 2^1 = 1$
- $\mathbb{B}^2 = \{(0,0), (0,1), (1,0), (1,1)\}\ :\ card\left(\mathbb{B}^2\right) = 2^2 = 4$
- $\mathbb{B}^3 = \{(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\}\ \ldots$

Remarque 1.5 : Soit $x = (x_1, \ldots, x_n) \in \mathbb{B}^n$. On note $\overline{x}$ le $n-$uplet de booléens défini par $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_n)$.

Définition 1.6 : Soient $x \in \mathbb{B}$ et $\varepsilon \in \{0, 1\}$. On note x^ε le booléen défini par :

$$x^\varepsilon = \begin{cases} x & si\ \varepsilon = 1 \\ \overline{x} & si\ \varepsilon = 0 \end{cases}$$

Remarque 1.7 : Soient $x = (x_1, \ldots, x_n) \in \mathbb{B}^n$ et $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n) \in \{0, 1\}^n$, on note x^ε le $n-$uplet de booléens défini par : $(x_1^{\varepsilon_1}, \ldots, x_n^{\varepsilon_n})$.

Définition 1.8 : On définit sur $\mathbb{B}^n$ la relation d'ordre partiel « **supérieur ou égal** », notée $\geq$, telle que $\forall (x_1, \ldots, x_n), (y_1, \ldots, y_n) \in \mathbb{B}^n$:

$$(x_1, \ldots, x_n) \geq (y_1, \ldots, y_n) \ \Leftrightarrow\ \forall i \in [\![1, n]\!],\ x_i \geq y_i$$

Remarque 1.9 : On définit de la même façon la relation d'ordre partiel « **inférieur ou égal** » et notée $\leq$.

Remarque 1.10 : Deux éléments x et y de $\mathbb{B}^n$ tels que $x \not\leq y$ et $x \not\geq y$ sont dits **incomparables**. On note alors $x \bowtie y$.

Exemple 1.11 : Soient cinq éléments de $\mathbb{B}^4$: $x_1 = (0,0,1,1)$, $x_2 = (1,1,0,1)$, $x_3 = (1,1,1,1)$, $x_4 = (0,1,1,1)$ et $x_5 = (0,0,1,0)$. On a :

$\nearrow$	x_1	x_2	x_3	x_4	x_5
x_1	$=$	$\bowtie$	$\leq$	$\leq$	$\geq$
x_2	$\bowtie$	$=$	$\leq$	$\bowtie$	$\bowtie$
x_3	$\geq$	$\geq$	$=$	$\geq$	$\geq$
x_4	$\geq$	$\bowtie$	$\leq$	$=$	$\geq$
x_5	$\leq$	$\bowtie$	$\leq$	$\leq$	$=$

2.2 Fonctions booléennes

2.2.1 Définition

Définition 2.1 : Une **fonction booléenne** à n variables ($\Leftrightarrow$ d'arité n) est une application

$$f : \mathbb{B}^n \longrightarrow \mathbb{B}$$

qui à un $n-$uplet de booléens associe un booléen.

Remarque 2.2 : On note $\mathbb{F}_n$ l'ensemble des fonctions booléennes d'arité n et

$$\mathbb{F} = \bigcup_{n=0}^{\infty} \mathbb{F}_n$$

l'ensemble de toutes les fonctions booléennes.

2.2.2 Fonctions usuelles

2.2.2.1 Fonctions de $\mathbb{F}_0$

Il existe 2 fonctions dans $\mathbb{F}_0$:

$$\mathbf{0} : \begin{array}{l} \longrightarrow \mathbb{B} \\ \longmapsto 0 \end{array} \quad \text{et} \quad \mathbf{1} : \begin{array}{l} \longrightarrow \mathbb{B} \\ \longmapsto 1 \end{array}$$

Ces deux fonctions sont bien évidemment des constantes[1].

[1] Informatiquement, on peut considérer qu'il s'agit de constructeurs.

2.2.2.2 Fonctions de $\mathbb{F}_1$

F_1 contient 4 éléments : les deux fonctions constantes

$$\begin{array}{rccc} \mathbf{0} : & \mathbb{B} & \longrightarrow & \mathbb{B} \\ & x & \longmapsto & 0 \end{array} \quad \text{et} \quad \begin{array}{rccc} \mathbf{1} : & \mathbb{B} & \longrightarrow & \mathbb{B} \\ & x & \longmapsto & 1 \end{array}$$

ainsi que deux fonctions de projections (positive et négative) :

$$\begin{array}{rccc} \textbf{identité} : & \mathbb{B} & \longrightarrow & \mathbb{B} \\ & x & \longmapsto & x \end{array} \quad \text{et} \quad \begin{array}{rccc} \textbf{négation} : & \mathbb{B} & \longrightarrow & \mathbb{B} \\ & x & \longmapsto & \overline{x} = \left\{ \begin{array}{ll} 0 & si \quad x = 1 \\ 1 & si \quad x = 0 \end{array} \right. \end{array}$$

2.2.2.3 Fonctions de $\mathbb{F}_2$

F_2 contient 16 éléments : les deux fonctions constantes

$$\begin{array}{rccc} \mathbf{0} : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & 0 \end{array} \quad \text{et} \quad \begin{array}{rccc} \mathbf{1} : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & 1 \end{array}$$

les deux fonctions de projections

$$\begin{array}{rccc} x_1 : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & x_1 \end{array} \quad \text{et} \quad \begin{array}{rccc} x_2 : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & x_2 \end{array}$$

les deux fonctions de négations des projections :

$$\begin{array}{rccc} \textbf{négation de } x_1 : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & \overline{x_1} \end{array}$$

et

$$\begin{array}{rccc} \textbf{négation de } x_2 : & \mathbb{B}^2 & \longrightarrow & \mathcal{B} \\ & (x_1, x_2) & \longmapsto & \overline{x_2} \end{array}$$

les fonctions « somme » (ou) et « produit » (et) :

$$\begin{array}{rccc} + : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & x_1 + x_2 = \left\{ \begin{array}{ll} 1 & si \quad x_1 = 1 \; ou \; x_2 = 1 \\ 0 & sinon \end{array} \right. \end{array}$$

et

$$\begin{array}{rccc} \times : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & x_1 \times x_2 = \left\{ \begin{array}{ll} 1 & si \quad x_1 = 1 \; et \; x_2 = 1 \\ 0 & sinon \end{array} \right. \\ & & & (\textit{également noté } x_1.x_2) \end{array}$$

les fonctions « implique » et « est impliqué par » :

$$\begin{array}{rccc} \Rightarrow : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & \overline{x_1} + x_2 \end{array} \quad \text{et} \quad \begin{array}{rccc} \Leftarrow : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & x_1 + \overline{x_2} \end{array}$$

les fonctions $\downarrow$ (nor) et $\uparrow$ (nand) :

$$\begin{array}{rccc} \downarrow : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & \overline{x_1 + x_2} \end{array} \quad \text{et} \quad \begin{array}{rccc} \uparrow : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & \overline{x_1.x_2} \end{array}$$

les fonctions « n'implique pas » et « n'est pas impliqué par » :

$$\begin{array}{rccc} \not\Rightarrow : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & x_1.\overline{x_2} \end{array} \quad \text{et} \quad \begin{array}{rccc} \not\Leftarrow : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & \overline{x_1}.x_2 \end{array}$$

enfin, les fonctions « équivalent » et « ou exclusif » :

$$\begin{array}{rccc} \Leftrightarrow : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & x_1.x_2 + \overline{x_1}.\overline{x_2} \end{array} \quad \text{et} \quad \begin{array}{rccc} \oplus : & \mathbb{B}^2 & \longrightarrow & \mathbb{B} \\ & (x_1, x_2) & \longmapsto & x_1.\overline{x_2} + \overline{x_1}.x_2 \end{array}$$

2.2.3 Table de vérité

Définition 2.3 : Pour représenter de façon simple et claire l'évaluation d'une fonction booléenne, on utilise une table de vérité. La table de vérité d'une fonction booléenne $f \in \mathbb{F}_n$ est un tableau comportant $n+1$ colonnes dont les n premières colonnes représentent les valeurs prises par les arguments $x_1,\ x_2,\ \ldots,\ x_n$ et la dernière colonne représente l'évaluation de la fonction $f(x_1, \ldots, x_n)$.

Exemple 2.4 : Les tables de vérité des fonctions $\times$, $+$ et $\Rightarrow \in \mathbb{F}_2$ sont les suivantes :

x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	$x_1 \times x_2$
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	$x_1 \Rightarrow x_2$
0	0	1
0	1	1
1	0	0
1	1	1

Exemple 2.5 : Complétez les tables de vérité suivantes :

x_1	x_2	$x_1 \oplus x_2$
0	0	
0	1	
1	0	
1	1	

x_1	x_2	$x_1 \Leftrightarrow x_2$
0	0	
0	1	
1	0	
1	1	

x_1	x_2	x_3		$x_1 + x_2.x_3$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Il est recommandé de mettre à profit la colonne supplémentaire permettant d'effectuer un calcul intermédiaire.

2.2.4 Propriétés

Les fonctions que nous venons de présenter possèdent un certain nombre de propriétés dont les suivantes :

$\overline{0} = 1$	$\overline{1} = 0$	$\overline{\overline{x}} = x$	$0 + x = x + 0 = x$
$1 + x = x + 1 = 1$	$0.x = x.0 = 0$	$1.x = x.1 = x$	$x + x = x$
$x.x = x$	$x + y = y + x$	$x.y = y.x$	$x + x.y = x$
$x + \overline{x} = 1$	$x.\overline{x} = 0$	$x \oplus 0 = 0 \oplus x = x$	$1 \oplus x = x \oplus 1 = \overline{x}$
$x.(y + z) = x.y + x.z$	$x + y.z = (x + y).(x + z)$	$x \oplus x = 0$	$x.(y \oplus z) = x.y \oplus x.z$
$x + \overline{x}.y = x + y$	lois de De Morgan :	$\overline{x + y} = \overline{x}.\overline{y}$	$\overline{x.y} = \overline{x} + \overline{y}$

Remarque 2.6 : Les opérateurs $+$, $\times$ et $\oplus$ sont associatifs et l'ordre de priorité des opérateurs est le suivant :

$$\textit{négation} \ > \ \times \ > \ + \ > \ \oplus \ = \Rightarrow = \Leftrightarrow$$

où « $>$ » signifie « est strictement plus prioritaire que » et « $=$ » signifie « est de même priorité que ».

Définition 2.7 : Soit $f \in \mathbb{F}_n$, on dit que f est **complète** ssi $\nexists i \in [\![1, n]\!]$, $\forall (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \in \mathbb{B}^{n-1}$:

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

autrement dit ssi tous les arguments de f « jouent un rôle » dans l'évaluation de f.

Exemple 2.8 : Les fonctions $f : (x_1, x_2, x_3) \mapsto x_1 + x_3$ et $g : (x_1, x_2, x_3) \mapsto x_1.(x_3 + x_2) + \overline{(x_2 + x_3) \Rightarrow x_1}$ ne sont pas complètes : $\forall x_1, x_2, x_3 \in \mathbb{B}^n$

$$\begin{aligned} f(x_1, 0, x_3) &= x_1 + x_3 \\ &= f(x_1, 1, x_3) \end{aligned}$$

et

$$\begin{aligned} f(x_1, x_2, x_3) &= x_1.(x_3 + x_2) + \overline{(x_2 + x_3) \Rightarrow x_1} \\ &= x_1.(x_3 + x_2) + (x_2 + x_3).\overline{x}_1 \end{aligned}$$

donc

$$\begin{aligned} f(0, x_2, x_3) &= x_2 + x_3 \\ &= f(1, x_2, x_3) \end{aligned}$$

2.2.5 Comparaisons

Définition 2.9 : Deux fonctions booléennes f et g de même arité n sont égales si et seulement si :

$$\forall x \in \mathbb{B}^n, \ f(x) = g(x)$$

Remarque 2.10 : Deux principales méthodes sont utilisées pour montrer que deux fonctions booléennes sont égales :

- (i) dresser la table de vérité des deux fonctions et constater qu'elles sont identiques
- (ii) utiliser les propriétés connues pour aboutir à l'égalité

Exemple 2.11 : On souhaite montrer que les fonctions $f : (x_1, x_2) \mapsto 1 \oplus x_1 \oplus x_2$ et $g : (x_1, x_2) \mapsto x_1 \Leftrightarrow x_2$ sont égales. Pour cela, on dresse la table de vérité de f puis celle de g. Pour établir la table de vérité de f, on se permet d'ajouter une colonne pour représenter le 1 intervenant dans la définition de la fonction et de procéder en 2 étapes :

1	x_1	x_2	$1 \oplus x_1$	$1 \oplus x_1 \oplus x_2$
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

donc

x_1	x_2	$1 \oplus x_1 \oplus x_2$
0	0	1
0	1	0
1	0	0
1	1	1

Or, la table de vérité de g étant la suivante :

x_1	x_2	$x_1 \Leftrightarrow x_2$
0	0	1
0	1	0
1	0	0
1	1	1

on peut conclure que les fonctions f et g sont égales. On aurait également pu procéder en utilisant les propriétés énoncées précédemment :

$$\begin{aligned} 1 \oplus x_1 \oplus x_2 &= (1 \oplus x_1) \oplus x_2 \\ &= \overline{x_1} \oplus x_2 \\ &= \overline{\overline{x_1}}.x_2 + \overline{x_1}.\overline{x_2} \\ &= x_1.x_2 + \overline{x_1}.\overline{x_2} \\ &= x_1 \Leftrightarrow x_2 \end{aligned}$$

Exemple 2.12 : Montrez de deux façons différentes que les fonctions $f : (x_1, x_2) \mapsto x_1 \Rightarrow (x_1 \oplus x_2)$ et $g : (x_1, x_2) \mapsto \overline{x_1.x_2}$ sont égales.

Définition 2.13 : On définit la relation d'ordre partiel sur $\mathbb{F}_n$ « **supérieur ou égal** », notée $\geq$, telle que $\forall f, g \in \mathbb{F}_n$:

$$f \geq g \;\Leftrightarrow\; \forall x \in \mathbb{B}^n,\; f(x) \geq g(x)$$

De la même façon, on définit la relation d'ordre partiel « **inférieur ou égal** », notée $\leq$.

Remarque 2.14 : Soient f et $g \in \mathbb{F}_n$ deux fonctions booléennes. Les propositions suivantes sont équivalentes :

(i) $f \leq g$

(*ii*) $f + g = g$
(*iii*) $f.g = f$

2.2.6 Support d'une fonction booléenne

Définition 2.15 : Le support $S_n(f)$ d'une fonction $f \in \mathbb{F}_n$ est l'ensemble des $n-$uplets $(x_1, ..., x_n) \in \mathbb{B}^n$ tels que $f(x_1, ..., x_n) = 1$.

Remarque 2.16 : $\forall x \notin S_n(f),\ f(x) = 0$.

Remarque 2.17 : Pour déterminer le support d'une fonction booléenne, il suffit de dresser la table de vérité de cette dernière.

Exemple 2.18 : Soit $f : (x_1, x_2) \mapsto 1 \oplus x_1 \oplus x_2$. Dans un exemple précédent, on a déterminé la table de vérité de f :

x_1	x_2	$1 \oplus x_1 \oplus x_2$
0	0	1
0	1	0
1	0	0
1	1	1

On a donc $S_2(f) = \{(0, 0), (1, 1)\}$.

Exemple 2.19 : Déterminez le support des fonctions suivantes : $f_1 : (x_1, \ldots, x_n) \mapsto 0$, $f_2 : (x_1, \ldots, x_n) \mapsto 1$ et $f : (x_1, x_2, x_3) \mapsto x_1 \times x_2 + \overline{x}_1 \times x_3$.

Remarque 2.20 : Il faut bien faire attention de prendre en considération les arguments d'une fonction qui n'apparaîssent pas dans le corps de la définition de cette dernière.

Exemple 2.21 : Soit $f : (x_1, x_2, x_3) \mapsto 1 \oplus x_1 \oplus x_2 \ \in \mathbb{F}_3$, le support de f est $S_3 = \{(0, 0, 0), (0, 0, 1), (1, 1, 0), (1, 1, 1)\}$.

Remarque 2.22 : On a les trois résultats suivants :

$$S_n\left(\sum_{i=1}^{p} f_i\right) = \bigcup_{i=1}^{p} S_n(f_i)$$

$$S_n\left(\prod_{i=1}^{p} f_i\right) = \bigcap_{i=1}^{p} S_n(f_i)$$

$$S_n(\overline{f}) = \overline{S_n(f)}$$

Remarque 2.23 : Soient f et $g \in \mathbb{F}_n$. On a les propriétés suivantes :
(*i*) $f = g$ si et seulement si $S_n(f) = S_n(g)$
(*ii*) $f \leq g$ si et seulement si $S_n(f) \subseteq S_n(g)$

2.2.7 Monômes

Définition 2.24 : On appelle **monôme conjonctif** de $\mathbb{F}_n$ tout produit de projections et de négations de projections, autrement dit tout élément pouvant s'écrire :

$$m = \prod_{i \in I} x_i^{\varepsilon_i}$$

où $I \subseteq [\![1, n]\!]$ et $\forall i \in I,\ \varepsilon_i \in \{0, 1\}$.

Remarque 2.25 : Si $I = \varnothing$, $m = 1$.

Exemple 2.26 : Donnez quelques exemples de monômes conjonctifs de $\mathbb{F}_4$:

Définition 2.27 : On appelle **monôme disjonctif** de $\mathbb{F}_n$ toute somme de projections et de négations de projections, autrement dit tout élément pouvant s'écrire :

$$m = \sum_{i \in I} x_i^{\varepsilon_i}$$

où $I \subseteq [\![1, n]\!]$ et $\forall i \in I,\ \varepsilon_i \in \{0, 1\}$.

Remarque 2.28 : Si $I = \varnothing$, $m = 0$.

Définition 2.29 : Un monôme de $\mathbb{F}_n$ est dit **canonique** si chacune des n variables intervient exactement une fois.

Exemple 2.30 : Donnez quelques exemples de monômes disjonctifs canoniques de $\mathbb{F}_5$:

Définition 2.31 : On appelle **ordre** d'un monôme le nombre de variables intervenant dans le monôme.

Remarque 2.32 : Un monôme de $\mathbb{F}_n$ est canonique ssi il est d'ordre n.

2.2.8 Formes normales de Lagrange

Théorème 2.33 : Toute fonction booléenne de $\mathbb{F}_n$ peut s'écrire comme somme de monômes conjonctifs canoniques de $\mathbb{F}_n$. Cette écriture, appelée **première forme normale de Lagrange** est déterminée à partir du support de la fonction :

$$f(x_1, ..., x_n) = \sum_{(\varepsilon_1, ..., \varepsilon_n) \in S_n(f)} x_1^{\varepsilon_1}.x_2^{\varepsilon_2}.\ldots.x_n^{\varepsilon_n}$$

Exemple 2.34 : Soit $f : (x_1, x_2, x_3) \mapsto 1 \oplus x_1 \oplus x_2 \ \in \mathbb{F}_3$. On a vu précédemment que le support de f est $S_3 = \{(0,0,0),(0,0,1),(1,1,0),(1,1,1)\}$. f peut donc s'écrire sous la forme suivante :

$$f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + x_1x_2x_3$$

Exemple 2.35 : Soit $f : (x_1, x_2, x_3) \mapsto x_3 \Leftrightarrow (x_1 \oplus x_2) \ \in \mathbb{F}_3$. Déterminez la première forme normale de Lagrange de f.

Théorème 2.36 : Toute fonction booléenne de $\mathbb{F}_n$ peut s'écrire comme produit de monômes disjonctifs canoniques de $\mathbb{F}_n$.

2.3 Opérations sur les fonctions booléennes

2.3.1 Composition de fonctions booléennes

2.3.1.1 Fonctions booléennes obtenues par composition

Définition 3.1 : Soient $f \in \mathbb{F}_n$ et $g_1, ..., g_n \in \mathbb{F}_m$. On note $f(g_1, \ldots, g_n)$ la fonction booléenne d'arité m définie par :

$$\begin{array}{rccc} f(g_1, \ldots, g_n) : & \mathbb{B}^m & \to & \mathbb{B} \\ & (x_1, \ldots, x_m) & \mapsto & f(g_1(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m)) \end{array}$$

Exemple 3.2 : Soient $f : (x_1, x_2) \mapsto x_1 \oplus x_2$, $g_1 : (x_1, x_2, x_3) \mapsto x_1x_2x_3$ et $g_2 : (x_1, x_2, x_3) \mapsto x_2 \Rightarrow x_1$. La fonction $h = f(g_1, h_2) \in \mathbb{F}_3$ est définie par :

$$h : (x_1, x_2, x_3) \mapsto x_1x_2x_3 \oplus (x_2 \Rightarrow x_1)$$

Définition 3.3 : Soit $E \subset \mathbb{F}$ un ensemble de fonctions booléennes. On définit **l'ensemble des fonctions obtenues par composition à partir de l'ensemble** E, noté $comp(E) \subset \mathbb{F}$ comme étant l'ensemble défini inductivement par :

i) **base** : tout élément de $E \cup \{(x_1, \ldots, x_n) \mapsto x_i,\ i = 1, \ldots, n\}$ appartient à $comp(E)$

ii) **induction** : pour toute opération $\phi \in E$ d'arité m et $f_1, \ldots, f_m$ appartenant à $comp(E)$ (et de même arité) : $\phi(f_1, \ldots, f_m) \in comp(E)$

Exemple 3.4 : Soit $E = \{ou, non\} \subset \mathbb{F}$ avec : $\begin{cases} ou : (x_1, x_2) \in \mathbb{B}^2 \mapsto x_1 + x_2 \\ non : x \in \mathbb{B} \mapsto \overline{x} \end{cases}$.
On note $proj_i$ la $i^{\text{ième}}$ fonction projection, *i.e.* $proj_i : (x_1, x_2) \in \mathbb{B}^2 \mapsto x_i$.

Montrons que $impl : (x_1, x_2) \in \mathbb{B}^2 \mapsto x_1 \Rightarrow x_2$ appartient à $comp(E)$.
- **base** : $proj_1$ et $proj_2$ appartiennent à $comp(E)$.
- **étape inductive 1** : $non(proj_1)$ appartient à $comp(E)$ car non appartient à E et $proj_1$ appartient à $comp(E)$
- **étape inductive 2** : $ou(non(proj_1), proj_2) \in comp(E)$ car ou appartient à E et $non(proj_1)$ et $proj_2$ appartiennent à $comp(E)$

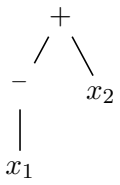
Or, $impl = ou(non(proj_1), proj_2)$, en effet, $\forall (x_1, x_2) \in \mathbb{B}^2$:

$$\begin{aligned} impl(x_1, x_2) &= x_1 \Rightarrow x_2 \\ &= \overline{x_1} + x_2 \\ &= ou(non(proj_1(x_1, x_2)), proj_2(x_1, x_2)) \end{aligned}$$

Donc $impl$ appartient à $comp(E)$.

Remarque 3.5 : On rappelle qu'à tout élément d'un ensemble construit inductivement on peut associer un arbre de construction dont :
- les feuilles sont des éléments de la base
- les noeuds représentent les étapes inductives

Exemple 3.6 : Pour l'exemple précédent, l'arbre de construction est le suivant :



2.3.1.2 Parties génératrices

Définition 3.7 : E est une partie génératrice ssi $comp(E) = \mathbb{F}$

Définition 3.8 : E est une partie génératrice minimale ssi
- E est une partie génératrice
- il n'existe pas $E' \subsetneq E$ tel que E' soit une partie génératrice

2.3.1.3 Propriétés stables par composition

Définition 3.9 : Toute fonction booléenne $f \in \mathbb{F}_n$ possède une fonction **duale** notée f^* telle que :

$$f^* : (x_1, ...x_i, ..., x_n) \in \mathbb{B}^n \mapsto \overline{f}(\overline{x_1}, ..., \overline{x_i}, ..., \overline{x_n})$$

Remarque 3.10 : La dualité est involutive : $f^{**} = f$

Exemple 3.11 : Soit $f : (x_1, x_2) \mapsto x_1 + x_2 \in \mathbb{F}_2$. $\forall (x_1, x_2) \in \mathbb{B}^2$:

$$\begin{aligned} f^*(x_1, x_2) &= \overline{f}(\overline{x}_1, \overline{x}_2) \\ &= \overline{\overline{x}_1 + \overline{x}_2} \\ &= \overline{\overline{x}}_1 \times \overline{\overline{x}}_2 \\ &= x_1 \times x_2 \end{aligned}$$

Conclusion : l'addition et la multiplication sont duales.

Définition 3.12 : Une fonction $f \in \mathbb{F}_n$ est dite **croissante** si et seulement si :

$$\forall x, y \in \mathbb{B}^n,\ x \geq y \Rightarrow f(x) \geq f(y)$$

Remarque 3.13 : Une fonction $f \in \mathbb{F}_n$ est croissante ssi

$$\forall x \in S_n(f),\ y \notin S_n(f),\ x \geq y$$

Exemple 3.14 : Soit $f : (x_1, x_2, x_3) \mapsto x_1 + (\overline{x}_2 \Rightarrow x_3)$. $S_3(f) = \mathbb{B}^3 \setminus \{(0,0,0)\}$. $\forall x \in S_3(f), (0,0,0) \leq x$, donc f est croissante.

Définition 3.15 : Soit $f \in F_n$, on dit que f est **linéaire** si $f \in Comp(\{0, 1, \oplus\})$.

Remarque 3.16 : Le terme linéaire vient du fait que toute fonction linéaire s'écrit sous la forme d'une somme exclusive ($\oplus$) de monômes d'ordre 1 : $0,\ 1,\ x,\ 1 \oplus x,\ x_1 \oplus x_2,\ 1 \oplus x_1 \oplus x_2,\ x_1 \oplus x_2 \oplus x_3 \oplus x_4 \ldots$

Théorème 3.17 : Soit $f \in \mathbb{F}_n$ une fonction complète. Alors f est linéaire ssi elle vérifie l'une des conditions suivantes :

- (i) f est la fonction constante 0
- (ii) f est la fonction constante 1
- (iii) $S_n(f) = \{(\varepsilon_1, \ldots, \varepsilon_n) \in \mathbb{B}^n \mid card(\{i \in [\![1, n]\!] \mid \varepsilon_i = 1\})\ \text{impair}\}$
- (iv) $S_n(f) = \{(\varepsilon_1, \ldots, \varepsilon_n) \in \mathbb{B}^n \mid card(\{i \in [\![1, n]\!] \mid \varepsilon_i = 1\})\ \text{pair}\}$

Exemple 3.18 : Soit $f : (x_1, x_2, x_3) \mapsto x_1 x_2 \overline{x}_3 + x_1 \overline{x}_2 x_3 + \overline{x}_1 x_2 x_3 + \overline{x}_1 \overline{x}_2 \overline{x}_3$. f est complète et $S_3(f) = \{(\varepsilon_1, \varepsilon_2, \varepsilon_3) \mid card(\{i \mid \varepsilon_i = 1\})\ \text{pair}\}$, elle est donc linéaire. Elle peut donc s'exprimer comme une somme exclusive de monômes d'ordre 1 : $f : (x_1, x_2, x_3) \mapsto 1 \oplus x_1 \oplus x_2 \oplus x_3$.

Théorème 3.19 : Soient les 5 propriétés suivantes :

$$\begin{cases} P_1(f) & \Leftrightarrow & f(0,\ldots,0)=0 \\ P_2(f) & \Leftrightarrow & f(1,\ldots,1)=1 \\ P_3(f) & \Leftrightarrow & f=f^* \\ P_4(f) & \Leftrightarrow & f \text{ croissante} \\ P_5(f) & \Leftrightarrow & f \in comp(\{0,1,\oplus\}) \end{cases}$$

Ces cinq propriétés sont **stables par composition**, autrement dit, pour tout $k = 1,2\ldots,5$:

$$P_k(f_1),\ldots,P_k(f_n) \quad \Rightarrow \quad \forall f \in Comp(\{f_1,\ldots,f_n\}),\ P_k(f)$$

Théorème 3.20 : On considère les cinq propriétés $P_1,\ldots,P_5$ du théorème 3.19. Une partie $E=\{f_1,\ldots,f_n\}$ est génératrice ssi pour chaque propriété P, il existe au moins un élément de E qui ne vérifie pas P :

$$comp(E)=\mathbb{F} \quad \Leftrightarrow \quad \forall k \in \{1,2,3,4,5\}, \exists f \in E, \neg P_k(f)$$

où $\neg P_k(f)$ signifie « f ne vérifie pas la propriété P_k ».

Exemple 3.21 : Soit $E=\{f_1 : x \mapsto 0,\ f_2 : x \mapsto \overline{x},\ f_3 : (x,y) \mapsto x+y\}$ une partie de $\mathbb{F}$. On s'autorise à écrire E sous la forme $\{0, \bar{}, +\}$. Pour montrer que E est une partie génératrice, nous devons trouver pour chaque propriété P_k un élément de E qui ne vérifie pas P_k :
- P_1 : $f_2(0)=1 \neq 0$ donc $\neg P_1(f_2)$
- P_2 : $f_1(1)=0 \neq 1$ donc $\neg P_2(f_1)$
- P_3 : $f_3^*=(x,y) \mapsto x \times y \ \neq \ f_3$ donc $\neg P_3(f_3)$
- P_4 : $f_2(0)=1$ et $f_2(1)=0 \not\geq 1$ donc $\neg P_4(f_2)$
- P_5 : f_3 est une complète et $S_2(f_3)=\{(0,1),(1,0),(1,1)\}$ donc $\neg P_5(f_3)$

Conclusion : E est une partie génératrice.

2.3.2 Simplification de fonctions booléennes

2.3.2.1 Définitions

Remarque 3.22 : Soient m et m' deux monômes conjonctifs de $\mathbb{F}_n$. m est plus grand que m' ($m \geq m'$) ssi $\exists m''$ monôme conjonctif tel que $m=m'.m''$, autrement dit si tous les facteurs de m apparaissent dans m'.

Exemple 3.23 : Soient $m_1=x_1\overline{x}_2x_4$ et $m_2=x_1\overline{x}_2\overline{x}_3x_4$. Tous les facteurs de m_1 (x_1, $\overline{x}_2$ et x_4) apparaissent dans m_2, donc $m_1 \geq m_2$.

Définition 3.24 : Un monôme m est **maximal** pour une fonction $f \in \mathbb{F}_n$ ssi il vérifie les trois conditions suivantes :

(i) m est un monôme conjonctif

(ii) $m \leq f$

(iii) $\nexists m' \neq m$ monôme conjonctif tel que $m \leq m' \leq f$

On note $\mathcal{M}(f)$ l'ensemble des monômes maximaux de f.

Définition 3.25 : Un monôme maximal m est **central** pour $f \in \mathbb{F}_n$ ssi

$$\nexists\ m_1, ..., m_n \in \mathcal{M}(f) \setminus \{m\} \text{ tels que } m \leq \sum_{i=1}^{n} m_i$$

On note $\mathcal{C}(f)$ l'ensemble des monômes centraux de f.

Définition 3.26 : Soit $f \in \mathbb{F}_n$ une fonction et $\tilde{f}$ une écriture de f. On dit que $\tilde{f}$ est une forme **simplifiée** de f ssi les deux conditions suivantes sont vérifiées :

(i) $\tilde{f}$ est une somme de monômes maximaux de f :

$$\tilde{f} \equiv \sum_{i=1}^{n} m_i,\ \forall i \in [\![1, n]\!],\ m_i \in \mathcal{M}(f)$$

(ii) si l'on retire un terme à $\tilde{f}$, alors on ne retrouve pas une écriture de f :

$$\nexists j \in [\![1, n]\!],\ f = \sum_{i=1, i\neq j}^{n} m_i$$

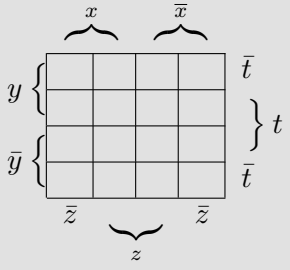
où $\equiv$ représente l'égalité syntaxique.

2.3.2.2 Méthode de Karnaugh

La méthode de Karnaugh est une méthode qui permet de déterminer simplement l'ensemble des formes simplifiées d'une fonction en en déterminant graphiquement les monômes maximaux. Cette méthode est principalement utilisée pour les fonctions d'arité inférieure ou égale à 4.

Méthode : Soit $f \in \mathbb{F}_4$.

(i) on dresse un tableau à $card(\mathbb{B}^4) = 2^4 = 16$:



chaque case représentant un élément de $\mathbb{B}^4$

(*ii*) pour chaque $(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4) \in \mathbb{B}^4$, on remplit la case $x^{\varepsilon_1} y^{\varepsilon_2} z^{\varepsilon_3} t^{\varepsilon_4}$ avec $f(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4)$

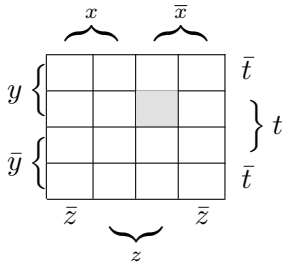
(*iii*) on détermine l'ensemble des monômes maximaux de f : on effectue les plus grands regroupements rectangulaires non discontinus possibles de 1, 2, 4, 8 ou 16 cases adjacentes.

(*iv*) on détermine l'ensemble des monômes centraux : on repère parmi les monômes maximaux ceux qui contiennent au moins une case « 1 » qui n'est pas recouverte par un autre monôme maximal.

(*v*) on détermine l'ensemble de toutes les combinaisons de monômes maximaux recouvrant toutes les cases « 1 ».

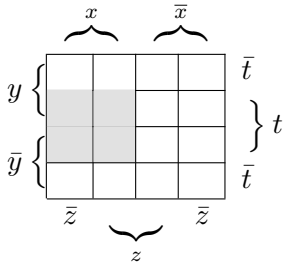
Remarque 3.28 :

(*i*) Chaque case du tableau représente un élément de $\mathbb{B}^4$, ou, de façon équivalente, un monôme d'ordre 4. Par exemple, la case grisée du tableau suivant représente le quadruplet $(0, 1, 1, 1)$ ou encore le monôme $x^0 y^1 z^1 t^1 = \overline{x}yzt$:

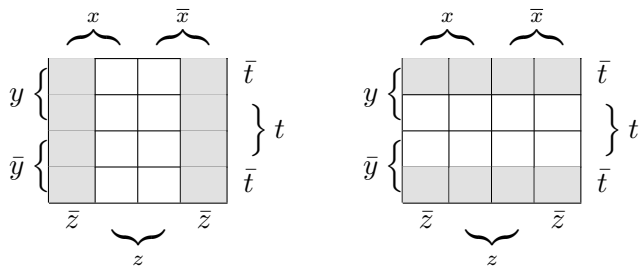


(*ii*) Le tableau de Karnaugh est une table de vérité présentée d'une façon particulière.

(*iii*) Regrouper des cases équivaut à additionner les monômes qu'elles représentent. Ainsi, regrouper 2 monômes d'ordre n qui ne diffèrent que par une variable (donc adjacent) permet d'obtenir un monôme d'ordre $n-1$. Ainsi, un monôme d'ordre 3 est représenté par un regroupement de $2 \times 1 = 2$ cases, un monôme d'ordre 2 par $2 \times 2 = 4$ cases , un monôme d'ordre 1 par $2 \times 4 = 8$ cases. Le regroupement des 16 cases correspond au seul monôme conjonctif d'ordre 0 : 1. Par exemple, le regroupement de cases suivant représente le monôme xt :



(*iv*) A noter que la première et la dernière colonne (resp. ligne) sont considérées comme étant adjacentes :



(*v*) Il faut bien comprendre que l'union de plusieurs cases représente la somme des monômes correspondants et que si un groupement de cases recouvre entièrement un autre groupement, c'est que le monôme correspondant au premier groupement est supérieur au second. Par définition, un monôme est central ssi il est maximal et s'il n'est pas inférieur à la somme (= l'union) des autres monômes maximaux. Il contient donc nécessairement au moins une case qui n'est recouverte par aucun autre monôme maximal.

(*vi*) On cherche l'ensemble de toutes les formes simplifiées de f, autrement dit, tous les sous-ensembles M de $\mathcal{M}(f)$ tels que

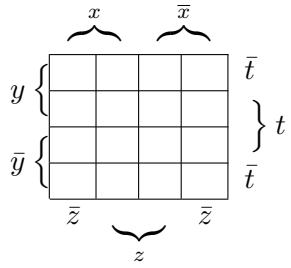
$$f = \sum_{m \in M} m$$

Or, si l'on note $c_1, \ldots, c_n$ l'ensemble des cases (monômes d'ordre 1) contenant « 1 », $f = c_1 + \ldots + c_n$ (il s'agit de la première forme normale de Lagrange) et $\forall m \in M,\ \exists I \subseteq [\![1, n]\!],\ m = \sum_{i \in I} c_i$. Donc $\sum_{m \in M} m = f \ \Leftrightarrow\ \sum_{m \in M} m = \sum_{i=1}^{n} c_i$, il faut donc que l'union de tous les m recouvre toutes (et uniquement) les cases « 1 ». Il est recommandé de représenter l'ensemble des combinaisons possibles par un arbre de possibilités. A noter que, par définition, les monômes centraux apparaissent dans toutes les solutions.

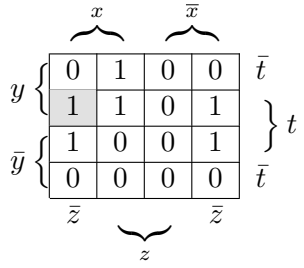
Exemple 3.29 : Soit $f \in \mathbb{F}_4$ telle que

$$S_4(f) = \{(1,1,1,0), (1,1,0,1), (1,1,1,1), (1,0,0,1), (0,1,0,1), (0,0,0,1)\}$$

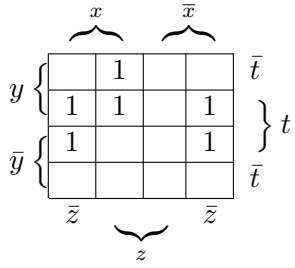
(*i*)



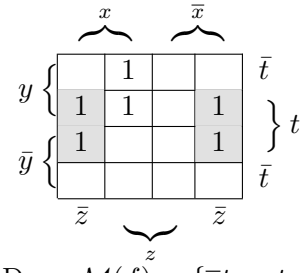
(*ii*)

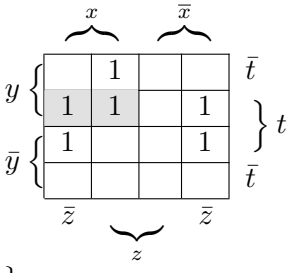


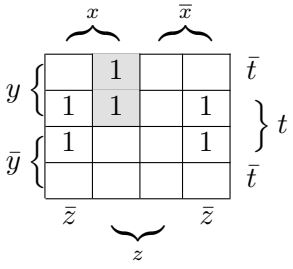
pour plus de lisibilité, on peut omettre les 0 :



(*iii*)







Donc $\mathcal{M}(f) = \{\overline{z}t, xyt, xyz\}$.

(*iv*) Seul $\overline{z}t$ recouvre au moins une case qui n'est recouverte par aucun autre monôme maximal, donc $\mathcal{C}(f) = \{\overline{z}t\}$.

(*v*) Il y a deux façons différentes de recouvrir toutes les cases « 1 » :

$$\overline{z}t + \left\{ \begin{array}{l} xyt \\ xyz \end{array} \right.$$

f possède donc deux formes simplifiées : $\overline{z}t + xyt$ et $\overline{z}t + xyz$.

Remarque 3.30 : Dans le cas où l'on ne dispose pas du support ou d'une définition de f qui nous permette de remplir le tableau de Karnaugh, on dispose d'un ensemble de propriétés qui nous permet de déterminer la valeur de f pour chaque élément de $\mathbb{B}^4$. Ces propriétés sont généralement exprimées d'une des deux façons suivantes :

(*i*) $expr \Rightarrow f(x, y, z, t)$
où $expr$ est une expression booléenne dont on cherche à déterminer le support. En effet, $expr \Rightarrow f(x, y, z, t)$ donc $S_4(expr) \subseteq S_4(f)$. On met alors « 1 » dans le tableau pour tous les éléments de $S_4(expr)$. A noter qu'il est équivalent (et plus simple) d'exprimer $expr$ sous la forme d'une somme de monômes conjonctifs : $expr = m_1 + \ldots + m_n$ et de mettre « 1 » dans l'ensemble des cases représentant $m_1,\ m_2, \ldots, m_n$.

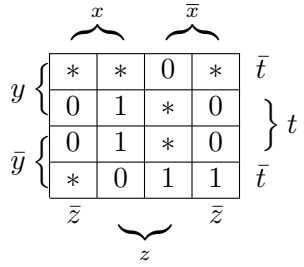
(*ii*) $f(x, y, z, t) \Rightarrow expr$
Sous cette forme, on ne peut pas déduire la valeur de f en fonction de celle de $expr$. On utilise la contraposée de cette propriété : $\overline{expr} \Rightarrow \overline{f}(x, y, z, t)$, donc $S_4(\overline{expr}) \subseteq S_4(\overline{f})$, autrement dit, lorsque $\overline{expr}$ vaut 1, $f(x, y, z, t)$ vaut 0. Comme dans le cas précédent, on exprime $\overline{expr}$ sous la forme d'une somme de monôme conjonctifs $m_1 + \ldots + m_n$ et l'on met « 0 » dans l'ensemble des cases représentant $m_1, \ldots, m_n$.

Une fois le tableau rempli, on poursuit selon la méthode exposée précédemment.

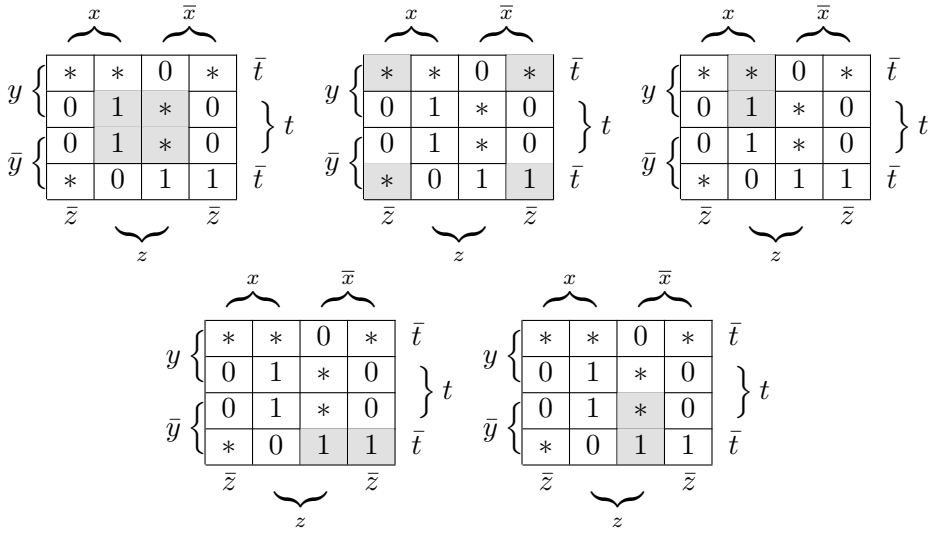
Remarque 3.31 : Il peut arriver que l'ensemble des propriétés décrivant f ne spécifie que partiellement cette dernière : certaines cases du tableau sont alors vides. On les remplit alors par des « $*$ ». L'ensemble de la méthode reste la même sauf la troisième étape :

(*iii*) on effectue des regroupements rectangulaires non discontinus de 1, 2, 4, 8 ou 16 cases de « 1 » et de « $*$ » et l'on note $\mathcal{M}(f_{max})$ (au lieu de $\mathcal{M}(f)$) l'ensemble des monômes maximaux obtenus. On retire de $\mathcal{M}(f_{max})$ l'ensemble des monômes constitués uniquement de « $*$ » et on note $\mathcal{M}(f)$ l'ensemble obtenu.

Exemple 3.32 : On considère le tableau de Karnaugh suivant



Les monômes maximaux ne contenant pas uniquement des $*$ sont les suivants :



Soit $\mathcal{M}(f) = \{zt, \overline{z}\overline{t}, xyz, \overline{x}\overline{y}\overline{t}, \overline{x}\overline{y}z\}$.

Exemple 3.33 : On considère une fonction booléenne f d'arité 4. On suppose que cette fonction vérifie les conditions suivantes :

(*a*) $xy(z \Leftrightarrow t) \Rightarrow f(x,y,z,t)$

(*b*) $x\overline{z}\overline{t} \Rightarrow f(x,y,z,t)$

(*c*) $zt(x \oplus y) \Rightarrow f(x,y,z,t)$

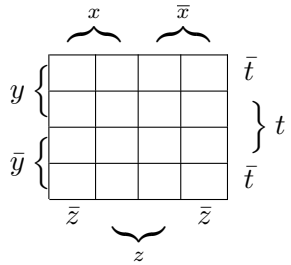
(*d*) $f(x,y,z,t) \Rightarrow x + \overline{y} + zt$

(*e*) $f(x,y,z,t) \Rightarrow ((y \Rightarrow t) + x\overline{z})$

(*f*) $f(x,y,z,t) \Rightarrow y + z + \overline{t}$

On cherche l'ensemble des formes simplifiées de f.

(*i*) On dresse le tableau de Karnaugh vide :



(*ii*) On cherche à remplir le tableau grâce aux propriétés :

(a) $xy(z \Leftrightarrow t) = xy(\overline{z}\overline{t} + zt) = xy\overline{z}\overline{t} + xyzt$. On remplit donc les cases correspondant aux monômes $xy\overline{z}\overline{t}$ et $xyzt$ par « 1 », ce que l'on notera $xy\overline{z}\overline{t} \leftarrow 1$ et $xyzt \leftarrow 1$.

(b) $x\overline{z}\overline{t} \leftarrow 1$

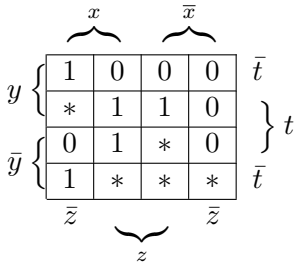
(c) $zt(x \oplus y) = zt(x\overline{y} + \overline{x}y) = x\overline{y}zt + \overline{x}yzt$. Donc $\begin{cases} x\overline{y}zt \leftarrow 1 \\ \overline{x}yzt \leftarrow 1 \end{cases}$

(d) $\overline{x + \overline{y} + zt} = \overline{x}y\overline{z} + \overline{x}y\overline{t}$. Donc $\begin{cases} \overline{x}y\overline{z} \leftarrow 0 \\ \overline{x}y\overline{t} \leftarrow 0 \end{cases}$

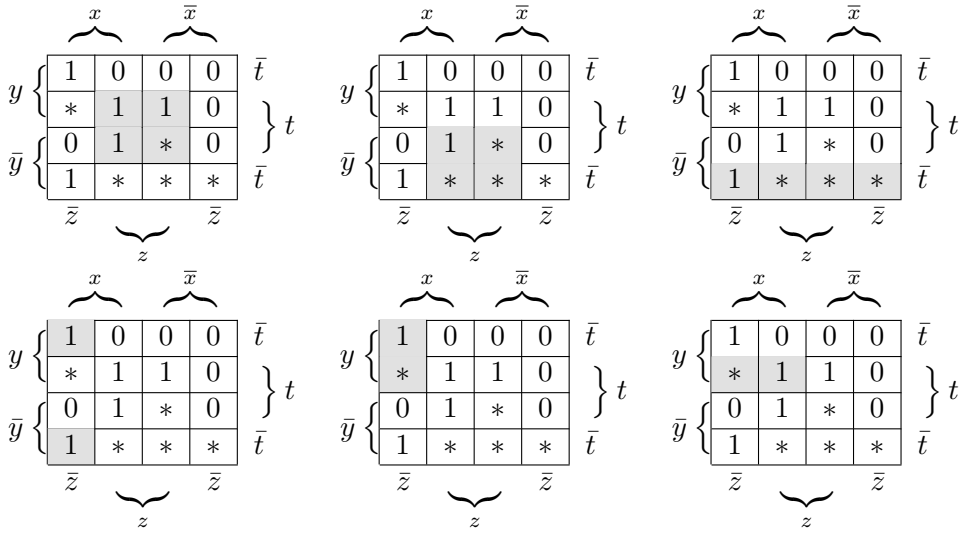
(e) $\overline{(y \Rightarrow t) + x\overline{z}} = \overline{(y \Rightarrow t)} \times \overline{x\overline{z}} = y\overline{t}(\overline{x} + z) = \overline{x}y\overline{t} + yz\overline{t}$ Donc $\begin{cases} \overline{x}y\overline{t} \leftarrow 0 \\ yz\overline{t} \leftarrow 0 \end{cases}$

(f) $\overline{y}\overline{z}t \leftarrow 0$

On obtient alors le tableau suivant :



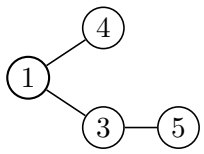
(*iii*)



On a donc : $\mathcal{M}(f) = \{zt, \overline{y}z, \overline{y}\overline{t}, x\overline{z}\overline{t}, xy\overline{z}, xyt\}$.
On notera $\mathcal{M}(f) = \{①, \ldots, ⑥\}$ dans la suite pour plus de lisibilité.

(*iv*) Seul un monôme recouvre au moins un « 1 » qui n'est recouvert par aucun autre monôme : $\mathcal{C}(f) = \{zt\}$.

(*v*) L'ensemble des formes simplifiées est le suivant :



Soit : $zt + x\overline{z}\overline{t}$ et $zt + \overline{y}\overline{t} + xy\overline{z}$. On remarque que deux monômes maximaux ne sont jamais utilisés : cela vient du fait que les cases « 1 » qui les composent sont recouvertes par des monômes centraux.

2.4 Travaux Dirigés

2.4.1 Exercice 1

Démontrez les propriétés suivantes :

(*i*) $\{f \in \mathbb{F}_n \mid f^* = f\} = \{f \in \mathbb{F}_n \mid \forall x \in \mathbb{B}^n,\ x \in S_n(f) \Leftrightarrow \overline{x} \notin S_n(f)\}$

(*ii*) $\forall (x_1, x_2, \ldots, x_n) \in \mathbb{B}^n : 1 \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_n = 1 \Leftrightarrow card(\{i \in [\![1, n]\!] \mid x_i = 1\})$ est pair

(*iii*) $\forall (x, y) \in \mathbb{B}^2,\ x \oplus y = \overline{x} \oplus \overline{y}$.

2.4.2 Exercice 2

Soit $\Rightarrow$ la relation sur $\mathbb{F}_n \times \mathbb{F}_n$ définie par :

$$\{(f,g) \in \mathbb{F}_n^2 \mid \forall x \in \mathbb{B}^n, \text{ si } f(x) = 1 \text{ alors } g(x) = 1\}$$

Montrez que $\Rightarrow$ et $\leq$ définissent la même relation sur $\mathbb{F}_n \times \mathbb{F}_n$.

2.4.3 Exercice 3

Soient $f_1(x,y,z) = xy + yz + xz$ et $f_2 : x \mapsto \bar{x}$ deux fonctions booléennes.

1) Montrez que $\{f_1, f_2\}$ n'est pas une partie génératrice.
2) Trouvez l'ensemble des $f \in \mathbb{F}_2$ tels que $\{f_1, f_2, f\}$ soit une partie génératrice (précisez quand $\{f_1, f_2, f\}$ est minimale).

2.4.4 Exercice 4

On considère la fonction booléenne $\phi(x_1, x_2, x_3) = x_1.x_3 + x_2.\overline{x}_3$

1) Quel principe réalise cette fonction ?
2) Déterminez $S_3(\phi)$.
3) Déterminez la première forme normale de Lagrange.
4) Montrez que pour $\forall f \in \mathbb{F}_n$, :

$$f(x_1, \ldots, x_n) = \phi\big(f(x_1, \ldots, x_{n-1}, 1)\,,\, f(x_1 \ldots, x_{n-1}, 0)\,,\, x_n\big)$$

5) Après avoir rappelé les propriétés stables par composition, montrez que $\{\phi, 0, 1\}$ est une partie génératrice minimale.

2.4.5 Exercice 5

On note $\mathbb{F}_n^C$ l'ensemble des fonctions complètes d'arité n. Trouvez et listez les fonctions de $\mathbb{F}_0^C$, $\mathbb{F}_1^C$ puis $\mathbb{F}_2^C$.

2.4.6 Exercice 6

Soit une fonction booléenne $f(x_1, ..., x_n)$ d'arité n. On appelle résidu de f par rapport à x_i (noté f_{x_i}) la fonction d'arité $n-1$ définie par :

$$f_{x_i} : (x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n) \mapsto f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n)$$

De la même façon, on appelle résidu de f par rapport à $\overline{x}_i$ (noté $f_{\overline{x}_i}$) la fonction d'arité $n-1$ définie par :

$$f_{\overline{x}_i} : (x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n) \mapsto f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n)$$

1) Montrez que $\forall f \in \mathbb{F}_n,\ f = x_i.f_{x_i} + \overline{x}_i.f_{\overline{x}_i}$
2) On définit la dérivée booléenne par rapport à x_i d'une fonction booléenne f par

$$\frac{\partial f}{\partial x_i} = f_{x_i} \oplus f_{\overline{x}_i}$$

De plus, on dit qu'une fonction booléenne est indépendante de x_i si pour tout $x = (x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n) \in \mathcal{B}^n$, la valeur de $f(x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n)$ ne change pas lorsque la valeur de x_i varie.
Montrez que f est indépendante de x_i si et seulement si

$$\frac{\partial f}{\partial x_i} = 0$$

3) Montrez que

$$\frac{\partial f}{\partial x_i} = \frac{\partial \overline{f}}{\partial x_i}$$

4) On admet la relation suivante

$$\frac{\partial (f.g)}{\partial x_i} = \left(f.\frac{\partial g}{\partial x_i}\right) \oplus \left(g.\frac{\partial f}{\partial x_i}\right) \oplus \left(\frac{\partial f}{\partial x_i}.\frac{\partial g}{\partial x_i}\right)$$

Montrez que :

$$\frac{\partial (f+g)}{\partial x_i} = \left(\overline{f}.\frac{\partial g}{\partial x_i}\right) \oplus \left(\overline{g}.\frac{\partial f}{\partial x_i}\right) \oplus \left(\frac{\partial f}{\partial x_i}.\frac{\partial g}{\partial x_i}\right)$$

2.4.7 Exercice 7

Soit $f \in \mathbb{F}_4$ une fonction booléenne vérifiant les conditions suivantes :

(i) $x\overline{y}(t \oplus z) \Rightarrow f(x,y,z,t)$

(ii) $\big((z \Rightarrow (x+t)) \Rightarrow xyt\big) \Rightarrow f(x,y,z,t)$

(iii) $f(x,y,z,t) \Rightarrow (xy \Rightarrow t)$

(iv) $f(x,y,z,t) \Rightarrow (t \Rightarrow x)(\overline{t} \Rightarrow z)$

Dressez le tableau de Karnaugh associé à cette fonction, déterminez les monômes maximaux puis centraux et en déduire les formes simplifiées de f

2.4.8 Exercice 8

Soient $n \geq 2$ et $f \in \mathbb{F}_n$. On dit que f définit implicitement sa $n^{\text{ième}}$ variable si :

$$\forall (x_1, \ldots, x_{n-1}) \in \mathbb{B}^{n-1},\ \exists\,!\, x_n \in \mathbb{B},\ f(x_1, \ldots, x_n) = 0$$

1) Soit $f_1 \in \mathbb{F}_2$ définie par la table de vérité suivante :

x_1	x_2	$f_1(x_1, x_2)$
0	0	1
0	1	0
1	0	0
1	1	1

Montrez que f_1 définit implicitement sa seconde variable. Donnez l'ensemble de toutes les fonctions de $\mathbb{F}_2$ qui définissent implicitement leur seconde variable.

2) Soit $f \in \mathbb{F}_n$. Montrez que f définit implicitement sa $n^{\text{ième}}$ variable si et seulement si :

$$\forall (x_1, \ldots, x_n) \in \mathbb{B}^n,\ f(x_1, \ldots, x_{n-1}, \overline{x}_n) = \overline{f(x_1, \ldots, x_n)}$$

3) Soit $f \in \mathbb{F}_n$ la fonction définie par :

$$f : (x_1, \ldots, x_n) \mapsto x_1.x_2 \ldots x_n + \sum_{i=1}^{n-1} \overline{x}_i.\overline{x}_n$$

Démontrez que f définit implicitement sa $n^{\text{ième}}$ variable, autrement dit qu'il existe une fonction booléenne $g \in \mathbb{F}_{n-1}$ telle que :

$$\forall (x_1, \ldots, x_n) \in \mathbb{B}^n,\ \big(f(x_1, \ldots, x_n) = 0 \quad \Leftrightarrow \quad x_n = g(x_1, \ldots, x_{n-1})\big)$$

que vous expliciterez.

Chapitre 3

Théorie des langages

3.1 Introduction à la théorie des langages

3.1.1 Définitions de base

Définition 1.1 : On appelle **alphabet** tout ensemble fini.

Remarque 1.2 : Les ensembles suivants sont des alphabets :
- $A = \{0, 1\}$: alphabet binaire
- $A = \{a, b, \dots, z, A, B, \dots Z\}$: alphabet latin
- $A = \{a, b, \dots, z, A, B, \dots Z, 0, 1, \dots, 9\}$: alphabet ASCII

Remarque 1.3 : Soit A un alphabet. Comme A est fini, on peut alors indicer ses éléments : $A = \{a_1, \dots, a_n\}$

Définition 1.4 : Les éléments d'un alphabet A sont appelés des **lettres**.

Remarque 1.5 : L'alphabet binaire est composé des lettres 0 et 1.

Définition 1.6 : On appelle **mot** sur un alphabet A toute suite finie de lettre de A. L'ensemble des mots sur A est noté A^*.

On donne également la définition suivante (équivalente à la première) d'un mot sur un alphabet A :

Définition 1.7 : On appelle **mot** sur un alphabet A toute application

$$\alpha : [\![1, n]\!] \to A$$

où $n \in \mathbb{N}$ est la **longueur** du mot, notée $|\,\alpha\,|$ et $\forall i \in [\![1, n]\!]\,,\ \alpha(i)$ (aussi notée α_i) est la $i^{\text{ième}}$ lettre du mot α.

Remarque 1.8 : On note un mot par la simple juxtaposition de ses lettres. Par exemple, sur l'alphabet binaire, le mot $(0, 1, 0, 1, 1, 1, 0)$, qui s'écrit également :

$$\begin{array}{rccc} \alpha : & 1 & \mapsto & 0 \\ & 2 & \mapsto & 1 \\ & 3 & \mapsto & 0 \\ & 4 & \mapsto & 1 \\ & 5 & \mapsto & 1 \\ & 6 & \mapsto & 1 \\ & 7 & \mapsto & 0 \end{array}$$

est simplement noté 0101110.

Remarque 1.9 : Pour tout alphabet A, il existe un unique mot de A^* de longueur nulle. Ce mot est noté ε (ou Λ selon les ouvrages) et est appelé **mot vide**.

Remarque 1.10 : Soit A un alphabet. On peut confondre toute lettre a de A avec le mot a de A^* de longueur 1.

Proposition 1.11 : Soient $\alpha = \alpha_1\,\alpha_2 \ldots \alpha_p$ et $\beta = \beta_1\,\beta_2 \ldots \beta_q$ deux mots sur l'alphabet A. On a l'équivalence suivante :

$$\alpha = \beta \quad \Leftrightarrow \quad \begin{cases} |\,\alpha\,| = |\,\beta\,| = n \\ \forall i \in [1, n],\ \alpha_i = \beta_i \end{cases}$$

Remarque 1.12 : Soient A un alphabet, $a \in A$ et $\alpha \in A^*$. On note $|\,\alpha\,|_a$ le nombre d'occurences de la lettre a dans le mot α. Si l'on note $\alpha = \alpha_1\,\alpha_2 \ldots \alpha_n$:

$$|\,\alpha\,|_a = \mathrm{card}\,\{i \in [1, n],\ \alpha_i = a\}$$

3.1.2 Loi produit et factorisation

Définition 1.13 : Soit A un alphabet. On appelle **produit** (ou **concaténation**) la loi de composition interne définie par :

$$\begin{array}{rccc} . \ : & A^* \times A^* & \to & A^* \\ & (\alpha, \beta) & \mapsto & \gamma = \alpha.\beta \end{array}$$

où $\alpha = \alpha_1\,\alpha_2 \ldots \alpha_p$, $\beta = \beta_1\,\beta_2 \ldots \beta_q$ et $\alpha.\beta = \alpha_1\,\alpha_2 \ldots \alpha_p\beta_1\,\beta_2 \ldots \beta_q$

Proposition 1.14 : La loi produit (ou concaténation) possède les propriétés suivantes :

- (i) $\forall \alpha, \beta \in A^*,\ |\,\alpha.\beta\,| = |\,\alpha\,| + |\,\beta\,|$
- (ii) $\forall \alpha, \beta, \gamma \in A^*,\ (\alpha.\beta).\gamma = \alpha.(\beta.\gamma)$ (la loi est associative)
- (iii) $\forall \alpha \in A^*,\ \alpha.\varepsilon = \varepsilon.\alpha = \alpha$ (le mot vide ε est l'élément neutre du produit)
- (iv) $\forall \alpha \in A^*,\ \alpha.\alpha = \alpha \Leftrightarrow \alpha = \varepsilon$ (le mot vide ε est le seul mot idempotent)

Remarque 1.15 : La définition et les propriétés (*ii*) et (*iii*) permettent de dire que $(A^*, .)$ est un monoïde[1].

Définition 1.16 : Soit A un alphabet. Soient $\alpha, \beta \in A^*$. On dit que α est **facteur** de β si

$$\exists \gamma, \delta \in A^*,\ \beta = \gamma.\alpha.\delta$$

Si $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **facteur propre** de β.

Définition 1.17 : Soit A un alphabet. Soient $\alpha, \beta \in A^*$. On dit que α est **préfixe** de β et l'on note $\alpha \sqsubseteq \beta$ si

$$\exists \delta \in A^*,\ \beta = \alpha.\delta$$

Si $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **préfixe propre** de β et l'on note alors $\alpha \sqsubset \beta$.

Remarque 1.18 : La relation $\sqsubseteq$ est une relation d'ordre :

(*i*) $\forall \alpha \in A^*,\ \alpha \sqsubseteq \alpha$ (refléxivité)

(*ii*) $\forall \alpha, \beta, \gamma \in A^*$, si $\alpha \sqsubseteq \beta$ et $\beta \sqsubseteq \gamma$ alors $\alpha \sqsubseteq \gamma$ (transitivité)

(*iii*) $\forall \alpha, \beta \in A^*$, si $\alpha \sqsubseteq \beta$ et $\beta \sqsubseteq \alpha$ alors $\alpha = \beta$ (antisymétrie)

Définition 1.19 : Soit A un alphabet. Soient $\alpha, \beta \in A^*$. On dit que α est **suffixe** de β si

$$\exists \gamma \in A^*,\ \beta = \gamma.\alpha$$

Si $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **suffixe propre** de β.

3.1.3 Définition et opérations sur les langages

3.1.3.1 Définition

Définition 1.20 : Soit A un alphabet. On appelle langage sur A toute partie (sous-ensemble) L de A^*. L'ensemble des langages sur A est donc :

$$\mathcal{P}(A^*) = \{L,\ L \subset A^*\}$$

Un langage sur un alphabet est donc un ensemble de mots sur cet alphabet.

Remarque 1.21 : A^* est le plus grand langage sur A au sens de l'inclusion.

Remarque 1.22 : Voici quelques exemples de langage :

- $L = \{aa, ab, ba, bb\}$ est le langage sur l'alphabet $A = \{a, b\}$ composé des mots de longueur 2.
- $L = \{\alpha \in \{a, b\}^*,\ |\alpha|_a = |\alpha|_b\}$ est le langage sur l'alphabet $A = \{a, b\}$ composé des mots contenant autant de a que de b.
- $L = \{\alpha \in \{0, 1\}^*,\ \exists \beta \in \{0, 1\}^*,\ \alpha = \beta.1\}$ est le langage sur l'alphabet $A = \{0, 1\}$ composé des nombres impairs représentés en base 2.
- $L = \varnothing$ est le langage vide. Il ne contient aucun mot.
- $L = \{\varepsilon\}$ est le langage contenant uniquement le mot vide.

[1] On rappelle qu'un **monoïde** est un ensemble muni d'une loi interne associative et d'un élément neutre.

3.1.3.2 Opérations ensemblistes

Soit A un alphabet. On définit les opérations ensemblistes (ou booléennes) sur les langages sur A.

Définition 1.23 : Soient L_1 et L_2 deux langages sur A. On appelle **union** de L_1 et de L_2 et l'on note $L_1 \cup L_2$ (ou $L_1 + L_2$) le langage défini par :

$$L_1 \cup L_2 = L_1 + L_2 = \{\alpha \in A^*,\ \alpha \in L_1 \vee \alpha \in L_2\}$$

Remarque 1.24 : On utilisera indifféremment la notation additive ($+$) ou ensembliste ($\cup$) pour représenter l'opérateur d'union.

Définition 1.25 : Soient L_1 et L_2 deux langages sur A. On appelle **intersection** de L_1 et de L_2 et l'on note $L_1 \cap L_2$ le langage défini par :

$$L_1 \cap L_2 = \{\alpha \in A^*,\ \alpha \in L_1 \wedge \alpha \in L_2\}$$

Définition 1.26 : Soient L_1 et L_2 deux langages sur A. On appelle **différence symétrique** (ou **réunion disjointe**) entre L_1 et L_2 et l'on note $L_1 \Delta L_2$ le langage défini par :

$$L_1 \Delta L_2 = \{\alpha \in A^*,\ \alpha \in L_1 \oplus \alpha \in L_2\} = L_1 \cup L_2 \setminus L_1 \cap L_2$$

Définition 1.27 : Soit L un langage sur A. On appelle **complémentaire** de L et l'on note $\overline{L}$ le langage défini par :

$$\overline{L} = A^* \setminus L = \{\alpha \in A^*,\ \alpha \notin L\}$$

On s'autorise également l'utilisation des autres opérations ensemblistes (produit cartésien, différence, ...) même si l'on n'en rappelle pas les définitions.

3.1.3.3 Opérations non ensemblistes

Soit A un alphabet. On définit de nouvelles opérations sur les langages sur A.

Définition 1.28 : Soient L_1 et L_2 deux langages sur A. On appelle **produit** (ou **concaténation**) de L_1 et de L_2 et l'on note $L_1.L_2$ le langage défini par :

$$L_1.L_2 = \{\gamma \in A^*,\ \exists(\alpha, \beta) \in L_1 \times L_2,\ \gamma = \alpha.\beta\}$$

Remarque 1.29 : Soient $L_1 = \{a^n,\ n \geq 0\}$ et $L_2 = \{b^n,\ n \geq 0\}$. $L_1.L_2 = \{a^p b^q,\ p, q \geq 0\}$

Définition 1.30 : Soient L un langage sur A. On appelle $\mathbf{n^{ième}}$ **puissance** de L et l'on note L^n le langage défini par :

$$\begin{cases} \{\varepsilon\} & \text{si } n = 0 \\ L & \text{si } n = 1 \\ L.L^{n-1} & \text{sinon} \end{cases}$$

Définition 1.31 : Soient L un langage sur A. On appelle **fermeture de Kleene** ou **étoile** (ou **itéré**) de L et l'on note L^* le langage défini par :

$$L^* = \bigcup_{n \geq 0} L^n = \sum_{n \geq 0} L^n$$

Il s'agit du plus petit langage sur A contenant L et le mot vide et stable par l'opération produit.

Remarque 1.32 : Cette définition peut également se formuler de façon inductive :

Définition 1.33 : Soient A^* l'ensemble des mots sur un alphabet quelconque A et $L \subset A^*$ un langage sur A. On définit inductivement la fermeture de Kleene de L, notée L^* par :

i) **base** : $B = L \cup \{\varepsilon\}$

ii) **induction** : $\mathcal{O}p = \{\text{produit}\}$

Autrement dit, L^* est le plus petit sous-ensemble de A^* vérifiant :

i) **base** : tout élément de $B = L \cup \{\varepsilon\}$ appartient à L^*

ii) **induction** : si α_1 et α_2 appartiennent à L^*, alors $\alpha_1.\alpha_2$ appartient à L^*

où $\alpha_1.\alpha_2$ est le produit de α_1 et de α_2.

Remarque 1.34 : Soient $A = \{a, b\}$ un alphabet, $L_1 = \{a\},\ L_2 = \{ab\}$ et $L_3 = A$ trois langages sur A. On a :
– $L_1^* = \{a^n,\ n \geq 0\}$
– $L_2^* = \{(ab)^n,\ n \geq 0\}$
– $L_3^* = A^*$

Définition 1.35 : Soient L un langage sur A. On appelle **étoile stricte** (ou **itéré strict**) de L et l'on note L^+ le langage défini par :

$$L^+ = \bigcup_{n > 0} L^n = \sum_{n > 0} L^n$$

Il s'agit du plus petit langage sur A contenant L et stable par l'opération produit.

Remarque 1.36 : La définition inductive de l'itéré strict d'un langage L est obtenu de la même façon que celle de le fermeture de Kleene mais en prenant $B = L$ pour base.

Remarque 1.37 : On a naturellement : $L^* = L^+ \cup \{\varepsilon\}$

Remarque 1.38 : $L^* = L^+ \Leftrightarrow \varepsilon \in L$

Définition 1.39 : Soit L un langage sur A. On appelle **réfléchi** de L et l'on note $\tilde{L}$ le langage défini par :

$$\tilde{L} = \{\phi(\alpha),\ \alpha \in L\}$$

où $\phi : A^* \to A^*$ est l'application définie par :

$$\begin{cases} \phi(\varepsilon) = \varepsilon \\ \forall \alpha \in A^*, a \in A, \phi(a.\alpha) = \phi(\alpha).a \end{cases}$$

Remarque 1.40 : ϕ est tout simplement la fonction qui « inverse » un mot :

$$\phi(\alpha_1\,\alpha_2\ldots\alpha_n) = \alpha_n\ldots\alpha_2\,\alpha_1$$

Définition 1.41 : Soient L_1 et L_2 deux langages sur A. On appelle **quotient gauche** de L_2 par L_1 et l'on note $L_1^{-1}.L_2$ le langage défini par :

$$L_1^{-1}.L_2 = \{\gamma \in A^*,\ \exists\alpha \in L_1,\ \alpha.\gamma \in L_2\}$$

De la même façon, on appelle **quotient droit** de L_1 par L_2 et l'on note $L_1.L_2^{-1}$ le langage défini par :

$$L_1.L_2^{-1} = \{\gamma \in A^*,\ \exists\alpha \in L_2,\ \gamma.\alpha \in L_1\}$$

Remarque 1.42 : L'opérateur quotient gauche est un opérateur particulièrement important en théorie des codes.

3.1.3.4 Propriétés

Soit A un alphabet. On énonce dans cette partie les propriétés remarquables des opérations précédemment définies.

Proposition 1.43 : L'union est associative, commutative, d'élément neutre $\varnothing$ et d'élément absorbant A^*. Autrement dit, $\forall L, L_1, L_2, L_3 \in \mathcal{P}(A^*)$ (langages sur A) :

- $(L_1 + L_2) + L_3 = L_1 + (L_2 + L_3)$
- $L_1 + L_2 = L_2 + L_1$
- $\varnothing + L = L + \varnothing = L$
- $L + A^* = A^* + L = A^*$

Proposition 1.44 : Le produit est associatif, d'élément neutre $\{\varepsilon\}$, d'élément absorbant $\varnothing$ et distributif par rapport à l'union (même infinie). Autrement dit, $\forall L, L_1, L_2, L_3 \in \mathcal{P}(A^*)$ (langages sur A) :

- $(L_1.L_2).L_3 = L_1.(L_2.L_3)$
- $\{\varepsilon\}.L = L$
- $\varnothing.L = L.\varnothing = \varnothing$
- $L_1.(L_2 + L_3) = L_1.L_2 + L_1.L_3$

Remarque 1.45 : Le produit n'est pas commutatif en général (sauf si $\mathrm{card}(A) \leq 1$). En effet, si $A = \{a, b\}$, $L_1 = \{a^n,\ n \geq 0\}$ et $L_2 = \{b^n,\ n \geq 0\}$:

$$L_1.L_2 = \{a^p b^q,\ p, q \geq 0\} \neq L_2.L_1 = \{b^p a^q,\ p, q \geq 0\}$$

Proposition 1.46 : La fermeture de Kleene (ainsi que l'étoile stricte) est idempotente, autrement dit, $\forall L \in \mathcal{P}(A^*)$:

- $(L^*)^* = L^*$
- $(L^+)^+ = L^+$

et possède la propriété suivante : $\forall L_1, L_2 \in \mathcal{P}(A^*)$,

$$(L_1^* + L_2^*)^* = (L_1 + L_2)^* = (L_1^*.L_2^*)^*$$

3.2 Langages rationnels

3.2.1 Définitions

Nous allons désormais parler d'une catégorie de langages bien connue : les langages dits **rationnels** ou encore **réguliers**. Pour toute cette partie, on considère un alphabet A et l'ensemble $\mathcal{P}(A^*)$ de tous les langages sur A.

Définition 2.1 : On appelle ensemble des langages **rationnels** (ou **réguliers**) et l'on note $Rat(A^*)$ l'ensemble défini inductivement par :

i) **base** : $B = \{\text{langages finis sur } A\}$

ii) **induction** : $\mathcal{O}p = \{\text{union, produit, itéré}\}$

Autrement dit, $Rat(A^*)$ est le plus petit sous-ensemble de $\mathcal{P}(A^*)$ vérifiant :

i) **base** : tout élément de $B = \{\text{langages finis sur } A\}$ appartient à $Rat(A^*)$

ii) **induction** : si L, L_1 et L_2 appartiennent à $Rat(A^*)$, alors $L_1 \cup L_2$, $L_1.L_2$ et L^* appartiennent à $Rat(A^*)$

où les symboles « $\cup$ », « . » et « * » représentent respectivement les opérations union, produit et itéré.

3.2.2 Expressions rationnelles

Tout langage rationnel peut s'exprimer au moyen de ce que l'on appelle **expression rationnelle** (ou **régulière**).

Définition 2.2 : On appelle ensemble des **expressions rationnelles** (ou **régulières**) sur A l'ensemble noté $\mathcal{R}_A$ et défini inductivement par :

i) **base** : $\begin{cases} \varnothing \in \mathcal{R}_A \\ \forall a \in A \cup \{\varepsilon\},\ a \in \mathcal{R}_A \end{cases}$

ii) **induction** : $\forall e_1, e_2 \in \mathcal{R}_A, \begin{cases} (e_1 + e_2) \in \mathcal{R}_A \\ (e_1\, e_2) \in \mathcal{R}_A \\ (e)^* \in \mathcal{R}_A \\ (e)^+ \in \mathcal{R}_A \end{cases}$

Remarque 2.3 : Les expressions rationnelles sont tout simplement un moyen d'écrire les langages rationnels.

Définition 2.4 : On définit alors une application qui à toute expression rationnelle associe le langage rationnel décrit par l'expression :

$$\begin{array}{rccl} \mathcal{L}_{\mathcal{R}} : & \mathcal{R}_A & \rightarrow & Rat(A^*) \\ & \varnothing & \mapsto & \varnothing \\ & \varepsilon & \mapsto & \{\varepsilon\} \\ & \alpha \in A & \mapsto & \{\alpha\} \\ & (e) & \mapsto & \mathcal{L}_{\mathcal{R}}(e) \\ & e_1 + e_2 & \mapsto & \mathcal{L}_{\mathcal{R}}(e_1) \cup \mathcal{L}_{\mathcal{R}}(e_2) \\ & e_1\, e_2 & \mapsto & \mathcal{L}_{\mathcal{R}}(e_1).\mathcal{L}_{\mathcal{R}}(e_2) \\ & e^* & \mapsto & \mathcal{L}_{\mathcal{R}}(e)^* \\ & e^+ & \mapsto & \mathcal{L}_{\mathcal{R}}(e)^+ \end{array}$$

Remarque 2.5 : Soit $e \in \mathcal{R}_A$ et $L \in Rat(A^*)$. Si $L = \mathcal{L}_\mathcal{R}(e)$, on dit que e **dénote** L.

Théorème 2.6 : Un langage est rationnel si et seulement si il existe une expression rationnelle qui le dénote.

Remarque 2.7 : L'application $\mathcal{L}_\mathcal{R}$ est surjective. Autrement dit, tout langage rationnel peut être décrit (dénoté) par au moins une expression rationnelle. La surjectivité de $\mathcal{L}_\mathcal{R}$ peut se démontrer simplement par induction structurelle. En revanche, elle n'est pas injective : si l'on considère le langage $L = \{a^n, n > 0\}$, on peut exhiber 3 expressions e_1, e_2 et $e_3 \in \mathcal{R}_A$ simples dénotant L, *i.e.* telles que $\mathcal{L}_\mathcal{R}(e_1) = \mathcal{L}_\mathcal{R}(e_2) = \mathcal{L}_\mathcal{R}(e_3) = L$:

- $e_1 = (a.a^*)$
- $e_2 = (a^*.a)$
- $e_3 = a^+$

Remarque 2.8 : Usuellement, on se permet de confondre un langage rationnel avec les expressions rationnelles qui le dénotent. Par exemple, on dira :

« Soit L le langage rationnel défini par $L = (ab)^*$ »

alors que l'on devrait dire

« Soit L le langage rationnel défini par $L = \mathcal{L}_\mathcal{R}(e)$ où $e = (ab)^*$ »

Remarque 2.9 : Cet abus d'écriture nous amènera à écrire des égalités du style :

$$(a^*b^*)^* = (a + b)^*$$

alors que les deux membres de cette égalité appartiennent à $\mathcal{R}_A$ et sont distincts ! Il faut alors comprendre que les langages dénotés par ces deux expressions sont égaux :

$$\mathcal{L}_\mathcal{R}\left((a^*b^*)^*\right) = \mathcal{L}_\mathcal{R}\left((a + b)^*\right)$$

3.3 Introduction à la théorie des codes

3.3.1 Définitions

Soient un alphabet A et $\mathcal{P}(A^*)$ l'ensemble de tous les langages sur A.

Définition 3.1 : On dit qu'un langage non vide $C \subset A^+$ est un code si tout mot de C^* se décompose de manière unique comme produit de mots de C.

Cette première définition peut également s'énoncer de la façon suivante :

Définition 3.2 : On dit qu'un langage non vide $C \subset A^+$ est un code si $\forall \alpha = \alpha_1\,\alpha_2 \ldots \alpha_p,\ \beta = \beta_1\,\beta_2 \ldots \beta_q \in C^*$:

$$\alpha = \beta \Leftrightarrow \left\{ \begin{array}{l} p = q \\ \forall i \in [1, p],\ \alpha_i = \beta_i \end{array} \right.$$

3.3.2 Propriétés et algorithme

La proposition suivante est une conséquence directe de la définition d'un code.

Proposition 3.3 : Soit C un sous-ensemble fini de A^+. Si C est un code, alors tout sous-ensemble de C est un code.

Proposition 3.4 : Soit C un sous-ensemble fini de A^* :

- (i) Si C est un code, $\varepsilon \notin C$
- (ii) Si C est un code, C et $\bigcup_{n\geq 2} C^n$ sont deux ensembles disjoints
- (iii) Si tous les mots de C sont de même longueur non nulle, alors C est un code. On parle alors de code **uniforme**.
- (iv) Si aucun mot de C n'est préfixe (resp. suffixe) d'un autre mot de C, C est un code. Dans ce cas, le code C est qualifié de code **préfixe** (resp. suffixe).

Pour terminer, on présente l'algorithme de Sardinas et Patterson qui permet de déterminer si un langage est un code ou non.

Algorithme 3.5 : Soit $L \in \mathcal{P}(A^*)$ un langage sur A. On cherche à déterminer si L est un code. L'**algorithme de Sardinas et Patterson** consiste en la construction d'une suite d'ensembles :

- *Initialisation* : $U_0 = L^{-1}.L \setminus \{\varepsilon\}$
- *Itération* : $U_{n+1} = U_n^{-1}.L \cup L^{-1}.U_n$
- *Condition d'arrêt* : $\begin{cases} \varepsilon \in U_n & \Rightarrow L \text{ n'est pas un code} \\ U_n = U_{n-1} & \Rightarrow L \text{ est un code} \end{cases}$

Remarque 3.6 : La condition d'arrêt $\varepsilon \in U_n$ est équivalente à $U_n \cap L \neq \varnothing$ (*i.e.* il existe un élément de U_n qui est un mot de L).

Remarque 3.7 : En remarquant que $U_n = \varnothing \Rightarrow U_{n+1} = \varnothing$, on peut arrêter l'algorithme dès que $U_n = \varnothing$ (et alors conclure que L est un code).

3.4 Travaux Dirigés

3.4.1 Exercice 1

Soient A un alphabet et $L,\ L_1,\ L_2,\ L_3$ et L_4 cinq langages sur A. Montrer que :

- (a) si $L_1 \subset L_2$ et $L_3 \subset L_4$ alors $L_1.L_3 \subset L_2.L_4$
- (b) si $L_1 \subset L_2$ alors $L_1^* \subset L_2^*$
- (c) $(L^*)^* = L^*$
- (d) $L^+ = L.L^* = L^*.L$
- (e) $L.L^* + \varepsilon = L^*$

3.4.2 Exercice 2

Soient A un alphabet et L, L_1, L_2, L_3 quatre langages sur A. Parmis les propriétés suivantes, lesquelles sont vraies ?

(a) $L + \emptyset = \emptyset + L = L$

(b) $L_1 + L_2 = L_2 + L_1$

(c) $(L_1 + L_2) + L_3 = L_1 + (L_2 + L_3)$

(d) $L_1.L_2 = L_2.L_1$

(e) $L + A^* = A^* + L = A^*$

(f) $(L_1.L_2).L_3 = L_1.(L_2.L_3)$

(g) $\{\varepsilon\}.L = L.\{\varepsilon\} = L$

(h) $\emptyset.L = L.\emptyset = \emptyset$

(i) $L_1.(L_2 + L_3) = L_1.L_2 + L_1.L_3$

3.4.3 Exercice 3

Soit $A = \{a, b\}$ un alphabet. Ecrire les expressions rationnelles (régulières) qui dénotent les langages suivants :

- L_1 = l'ensemble des mots de A^* se terminant par a
- L_2 = l'ensemble des mots de A^* contenant au moins un a
- L_3 = l'ensemble des mots de A^* contenant au plus un b
- L_4 = l'ensemble des mots de A^* contenant un nombre pair de a
- L_5 = l'ensemble des mots de A^* contenant autant de a que de b

Avez-vous rencontré une difficulté pour décrire l'un de ces langages ? Qu'en conclure ?

3.4.4 Exercice 4

Soit $A = \{a, b\}$ un alphabet. Décrire les langages dénotés par les expression rationnelles suivantes :

- $e_1 = (a + b)^*$
- $e_2 = (a^*b^*)^*$
- $e_3 = a^*\, b\, a^*\, b\, a^*\, b\, a^*$

3.4.5 Exercice 5

Soit $L = \{(ab^nc), n \geq 0\}$ un langage sur $A = \{a, b, c\}$. Parmi les langages suivants, lequel représente L^* ? Donnez un exemple de mot appartenant à L et un appartenant à L^*.

- $L_1 = \{(ab^nc)^p,\ n \geq 0, p \geq 0\}$
- $L_2 = \{(ab^nc)^p,\ n \geq 0, p > 0\}$
- $L_3 = \left\{ \prod_{i=1}^{p} (ab^{n_i}c),\ \forall i \in [1, p], n_i \geq 0, p \geq 0 \right\}$
- $L_4 = \left\{ \prod_{i=1}^{p} (ab^{n_i}c),\ \forall i \in [1, p], n_i \geq 0, p > 0 \right\}$
- $L_5 = \{(ab^nc), n \geq 0\}$

L peut-il être l'itéré d'un autre langage ? Si oui, donnez le langage M tel que $L = M^*$, sinon, expliquez pourquoi.

3.4.6 Exercice 6

On rappelle qu'un langage non vide $C \subset A^+$ est un code si tout mot de C^* se décompose de manière unique comme produit de mots de C ainsi que les propriétés suivantes :

1. Tout sous-ensemble d'un code est un code.
2. Si C est un code, $\varepsilon \notin C$
3. Si C est un code, C et $\bigcup_{n\geq 2} C^n$ sont deux ensembles disjoints
4. Si tous les mots de C sont de même longueur non nulle, alors C est un code. On parle alors de code uniforme.
5. Si aucun mot de C n'est préfixe (resp. suffixe) d'un autre mot de C, C est un code. Dans ce cas, le code C est qualifié de code préfixe (resp. suffixe).

Lorsque ces propriétés ne sont pas suffisantes pour monter qu'un langage est ou n'est pas un code, on a recours à l'algorithme de Sardinas et Patterson qui consiste en la construction d'une suite d'ensembles :

- *Initialisation* : $U_0 = L^{-1}.L \setminus \{\varepsilon\}$
- *Itération* : $U_{n+1} = U_n^{-1}.L \cup L^{-1}.U_n$
- *Condition d'arrêt* : $\begin{cases} \varepsilon \in U_n & \Rightarrow L \text{ n'est pas un code} \\ U_n = U_{n-1} & \Rightarrow L \text{ est un code} \end{cases}$

Parmi les langages suivants, déterminer ceux qui sont des codes[2]

- $L_1 = \{a, b\}$
- $L_2 = ab^*$
- $L_3 = \{a, ab, ba\}$
- $L_4 = \{a, ba, bb\}$
- $L_5 = \{aa, baa, ba\}$
- $L_6 = \{a, abbba, babab, bb\}$
- $L_7 = a^+b^+$
- $L_8 = a^+b^*$
- $L_9 = \{a^nb^n, n \in \mathbb{N}^*\}$
- $L_{10} = (ab)^*$

[2] On demande une démonstration pour chaque langage. Il est recommandé d'utiliser prioritairement les propriétés.

Chapitre 4

Théorie des automates

4.1 Introduction à la théorie des automates

4.1.1 Définition d'un automate fini

Définition 1.1 : On appelle automate fini (AF) tout quintuplet

$$\mathcal{A} = (A, Q, I, E, T)$$

où
- A est un **alphabet**
- Q est un ensemble fini, l'ensemble des **états**
- $I \subset Q$ est l'ensemble des **états initiaux**
- $T \subset Q$ est l'ensemble des **états finaux (terminaux)**
- $E \subset Q \times A \cup \{\varepsilon\} \times Q$ est une relation appelée **relation de transition**

Remarque 1.2 : Vous pourrez trouver dans la littérature la définition suivante équivalente : « On appelle automate fini sur un alphabet A tout quadruplet $\mathcal{A} = (Q, I, E, T)$ ... ».

Définition 1.3 : Soit $\mathcal{A} = (A, Q, I, E, T)$ un automate fini. Pour toute transition (p, a, q) de E, on dit que :
- a (qui appartient à A) est l'**étiquette** de la transition
- p (qui appartient à Q) est l'**origine** de la transition
- q (qui appartient à Q) est l'**extrémité** de la transition

On note également $p \xrightarrow{a} q$ la transition (p, a, q).

Définition 1.4 : On appelle **transition vide** ou **ε-transition** toute transition d'étiquette ε.

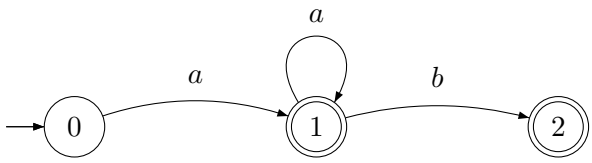
Remarque 1.5 : Cette transition joue un rôle particulier. En effet, si l'on essaie de comprendre le fonctionnement d'un automate, on peut associer à toute transition $q \xrightarrow{a} q'$ l'assertion suivante « si l'on est en q et que l'on lit un a, on arrive en q' ». Une ε-transition $q \xrightarrow{\varepsilon} q'$ est donc une transition qui permet de passer de l'état q à l'état q' sans avoir besoin de lire une lettre.

4.1.2 Représentation sagittale (par un graphe étiqueté)

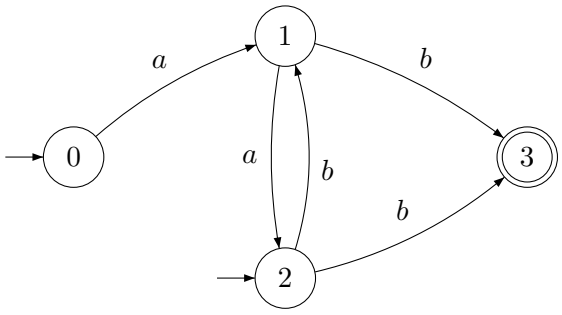
Tout automate se représente sous forme sagittale (*i.e.* par un graphe étiqueté). Soit $\mathcal{A} = (A, Q, I, E, T)$ un AF et $\mathcal{G}$ sa représentation graphique, alors :

- les sommets de $\mathcal{G}$ sont les états de $\mathcal{A}$ (donc les éléments de Q)
- pour toute transition $p \xrightarrow{a} q \in E$, on trace un arc orienté d'origine p et de destination q étiqueté par la lettre a
- pour tout état initial p de I, on ajoute une flèche sans origine et pointant sur p
- pour tout état terminal q de T, soit on ajoute une flèche d'origine q pointant vers l'extérieur du graphe, soit on ajoute un cercle inscrit dans le premier cercle représentant le sommet

Exemple 1.6 : Soit $\mathcal{A} = (A, Q, I, E, T)$ un automate défini par :
$A = \{a, b\}$, $Q = \{0, 1, 2\}$, $I = \{0\}$, $T = \{1, 2\}$ et $E = \big\{(0, a, 1), (1, a, 1), (1, b, 2)\big\}$.
La représentation sagittale de $\mathcal{A}$ est alors la suivante :



Soit $\mathcal{A}' = (A', Q', I', E', T')$ un autre automate fini défini par :
$A' = \{a, b\}$, $Q' = \{0, 1, 2, 3\}$, $I' = \{0, 2\}$, $T' = \{3\}$ et $E' = \big\{(0, a, 1), (1, b, 3), (1, a, 2), (2, b, 1), (2, b, 3)\big\}$. La représentation sagittale de $\mathcal{A}'$ est alors la suivante :



Définition 1.7 : Soit $\mathcal{A} = (A, Q, I, E, T)$ un automate fini. On appelle **calcul** dans $\mathcal{A}$ toute suite de transitions $(q_i \xrightarrow{a_i} q_{i+1})_{i \in [1, n-1]}$ tel que l'extrémité d'une transition est l'origine de la suivante. On note un tel calcul de la façon suivante :

$$c = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \xrightarrow{a_3} \dots \xrightarrow{a_{n-1}} q_n$$

On dira que :

- $a_1 a_2 a_3 \dots a_{n-1} \in A^*$ est l'étiquette du calcul c
- $q_1 \in Q$ est son origine
- $q_n \in Q$ est son extrémité

Exemple 1.8 : Le calcul défini par :

$$c = \big((1,a,1),(1,b,2),(2,b,3),(3,a,2)\big) = 1 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{b} 3 \xrightarrow{a} 2$$

a pour origine l'état 1, pour extrémité l'état 2 et pour étiquette le mot *abba*.

Remarque 1.9 : On s'autorise désormais à écrire $q \xrightarrow{\alpha} q'$ pour désigner un calcul d'origine q, d'étiquette $\alpha \in A^*$ et d'extrémité q' sans expliciter les états intermédiaires.

Définition 1.10 : On dit qu'un calcul dans un automate $\mathcal{A} = (A, Q, I, E, T)$ est **réussi** lorsque son origine appartient à I (*i.e.* est un état initial) et son extrémité à T (*i.e.* est un état final) :

$$c \text{ est un calcul réussi} \quad \Leftrightarrow \quad c = p \xrightarrow{\alpha} q,\ p \in I, q \in T,\ \alpha \in A^*$$

Définition 1.11 : Soit $\mathcal{A} = (A, Q, I, E, T)$ un automate fini et $q \in Q$. On dit que q est **accessible** si et seulement si il existe un mot $\alpha \in A^*$ et un état initial $q_0 \in I$ tel que $q_0 \xrightarrow{\alpha} q$, autrement dit, s'il existe un calcul dont l'origine est un état intial et dont il est l'extrémité. L'ensemble des états accessibles de $\mathcal{A}$ est donc :

$$\big\{q \in Q,\ \exists \alpha \in A^*,\ \exists q_0 \in I,\ q_0 \xrightarrow{\alpha} q\big\}$$

Un état qui n'est pas accessible est dit **inaccessible**.

4.1.3 Déterminisme et indéterminisme

La définition des automates finis qui a été donnée est générale. On distingue deux types d'automates finis : les automates finis déterministes et les automates finis indéterministes. Intuitivement, un automate fini sera déterministe si on ne peut pas trouver deux moyens différents (deux calculs) pour faire la même chose (lire une lettre ou un mot à partir d'un état). Plus formellement, on considèrera que la définition d'un automate fini indéterministe est celle qui a été donnée en début de chapitre et celle d'un automate fini déterministe est la suivante :

Définition 1.12 : On appelle **automate fini déterministe** tout quintuplet

$$\mathcal{A} = (A, Q, q_0, \delta, T)$$

tel que :
- A est un alphabet
- Q est un ensemble fini, l'ensemble des états
- $q_0 \in Q$ est l'unique état initial
- $\delta : Q \times A \to Q$ est la fonction transition
- $T \subset Q$ est l'ensemble des états terminaux

Remarque 1.13 : On remarque que la définition d'un automate fini déterministe est une restriction de celle d'un automate fini indéterministe avec les contraintes suivantes :
- I est un singleton
- E est une fonction : *i.e.* pour tout état $q \in Q$ et toute lettre $a \in A$ il existe au plus un état $q' \in Q$ tel que $q \xrightarrow{a} q'$

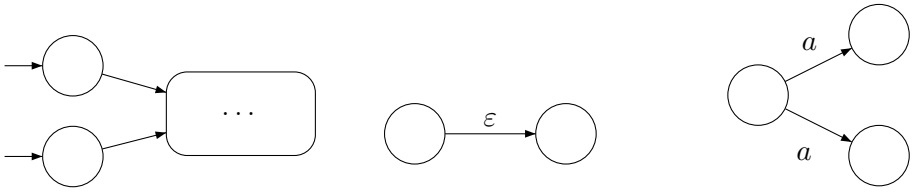
Remarque 1.14 : Soit $\mathcal{A} = (A, Q, I, E, T)$ un automate. S'il comporte au moins un des éléments suivants :

i) plusieurs états initiaux

ii) une transition vide (entre deux états distincts)

iii) un état qui est l'origine de plusieurs transitions différentes de même étiquette

alors $\mathcal{A}$ est indéterministe, sinon, il est déterministe.

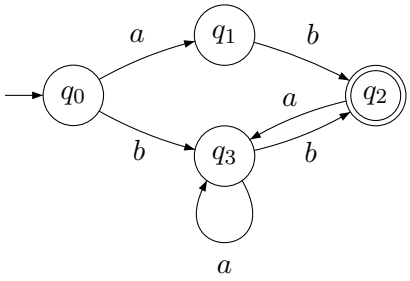


cas *i*) cas *ii*) cas *iii*)

Remarque 1.15 : Comme l'indique la définition, dans le cas de l'automate fini déterministe, la relation transition $E : Q \times A \times Q$ devient une fonction $\delta : Q \times A \to Q$, càd une relation qui à chaque couple de $Q \times A$ associe **au plus** un élément de Q. Ceci étant, le même vocabulaire est utilisé, *i.e.*, pour $\delta(p, a) = q$:

- a est l'étiquette
- p est l'origine
- q est l'extrémité

On exprime alors souvent la fonction transition sous forme d'un tableau où l'on met en tête de chaque colonne les lettres de l'alphabet A et en tête de chaque ligne les états de Q. Par exemple, si l'on considère l'automate représenté par le graphe suivant :



La fonction transition est donnée par le tableau suivant :

δ	a	b
q_0	q_1	q_3
q_1	$-$	q_2
q_2	q_3	$-$
q_3	q_3	q_2

$\leftarrow A$

$\uparrow$

Q

Remarque 1.16 : De la même façon que la relation transition devient une fonction transition, l'ensemble des calculs dans un automate fini déterministe peut se voir comme le prolongement de la fonction transition.

Définition 1.17 : Soit $\mathcal{A} = (A, Q, q_0, \delta, T)$ un automate fini déterministe. La fonction $\delta : Q \times A \to Q$ se prolonge en une fonction $\delta^* : Q \times A^* \to Q$ définie par :
- $\forall q \in Q,\ \delta^*(q, \varepsilon) = q$
- $\forall q \in Q,\ \forall \alpha \in A^*,\ \forall a \in A,\ \delta^*(q, a.\alpha) = \delta^*(\delta(q, a), \alpha)$

Exemple 1.18 : Si l'on reprend l'automate défini plus haut, on peut calculer l'image de $(q_0, abab)$ par δ^* :

$$\begin{array}{rclcrclcrcl} \delta^*(q_0, abab) & = & \delta^*\left(\delta(q_0, a), bab\right) & = & \delta^*\left(q_1, bab\right) & = & \delta^*\left(\delta(q_1, b), ab\right) \\ & = & \delta^*\left(q_2, ab\right) & = & \delta^*\left(\delta(q_2, a), b\right) & = & \delta^*\left(q_3, b\right) \\ & = & \delta^*\left(\delta(q_3, b), \varepsilon\right) & = & \delta^*\left(q_2, \varepsilon\right) & = & q_2 \end{array}$$

4.1.4 Langage reconnaissable par un automate fini

Définition 1.19 : Soit $\mathcal{A} = (A, Q, I, E, T)$ un automate fini. On dit qu'un mot $\alpha \in A^*$ est **reconnu** (ou **accepté**) par $\mathcal{A}$ s'il existe un calcul réussi d'étiquette α :

$$\exists\, (q_0, \ldots, q_n) \in Q^{n+1},\ q_0 \in I,\ q_n \in T,\ q_0 \xrightarrow{\alpha_1} q_1 \xrightarrow{\alpha_2} \ldots q_{n-1} \xrightarrow{\alpha_n} q_n$$

où $\alpha = \alpha_1\alpha_2 \ldots \alpha_n$ est de longueur n. Ce qui s'écrit, sous forme plus condensée :

$$\exists q_0 \in I,\ q_n \in T,\ q_0 \xrightarrow{\alpha} q_n$$

Définition 1.20 : Soit $\mathcal{A} = (A, Q, I, E, T)$. On appelle **langage reconnu** par $\mathcal{A}$ l'ensemble des mots reconnu par cet automate :

$$\mathcal{L}(\mathcal{A}) = \left\{\alpha \in A^*,\ \exists q \in I,\ \exists q' \in T,\ q \xrightarrow{\alpha} q'\right\}$$

Définition 1.21 : On dit qu'un langage est **reconnaissable par un AF** s'il existe un automate fini qui le reconnaît. On note $Rec(A^*)$ l'ensemble des langages sur l'aphabet A reconnaissables par un automate fini.

4.2 Équivalence entre automates finis et langages rationnels

4.2.1 Théorème de Kleene

Théorème 2.1 : Soit A un alphabet, $Rat(A^*)$ l'ensemble des langages rationnels (réguliers) sur A et $Rec(A^*)$ l'ensemble des langages sur A reconnaissables par un automate fini. Le **théorème de Kleene** indique que :

$$Rat(A^*) = Rec(A^*)$$

Autrement dit, les langages rationnels sont exactement ceux reconnus par les automates finis déterministes.

Remarque 2.2 : Ce théorème nous permet de dire que pour toute expression régulière dénotant un langage $\mathcal{L}$, on peut construire un automate reconnaissant $\mathcal{L}$ et, inversement, à tout automate $\mathcal{A}$ on peut associer une expression régulière dénotant $\mathcal{L}(\mathcal{A})$.

4.2.2 Des expressions rationnelles aux automates finis

On rappelle la définition des expressions rationnelles sur A (définition 2.2) : $\mathcal{R}_A$ est le plus petit ensemble tel que :

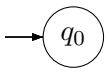
i) $\varnothing \in \mathcal{R}_A$

ii) $\forall a \in A \cup \{\varepsilon\},\ a \in \mathcal{R}_A$

iii) $\forall e_1, e_2 \in \mathcal{R}_A, \begin{cases} (e_1 + e_2) \in \mathcal{R}_A \\ (e_1\, e_2) \in \mathcal{R}_A \\ (e)^* \in \mathcal{R}_A \\ (e)^+ \in \mathcal{R}_A \end{cases}$

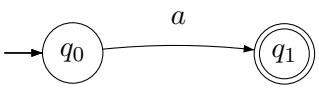
Comme toute expression rationnelle dénote un langage reconnaissable par un automate fini, on définit une procédure de construction de cet automate sur la base de la définition des expressions rationnelles. Ainsi :

i) à l'expression rationnelle vide ($\varnothing$) on associe l'automate $\mathcal{A} = (A, Q, I, E, T)$ à un état intial $I = \{q_0\}$ et zéro état final $T = \varnothing$:



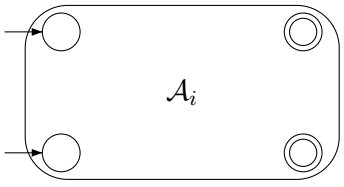
Cet automate ne reconnaît aucun mot, donc $\mathcal{L}(\mathcal{A}) = \varnothing = \mathcal{L}_{\mathcal{R}}(\varnothing)$

ii) à tout $a \in A \cup \{\varepsilon\}$ on associe l'automate $\mathcal{A} = \big(Q = \{q_0, q_1\}, A, E = \{q_0 \xrightarrow{a} q_1\}, I = \{q_0\}, T = \{q_1\}\big)$:



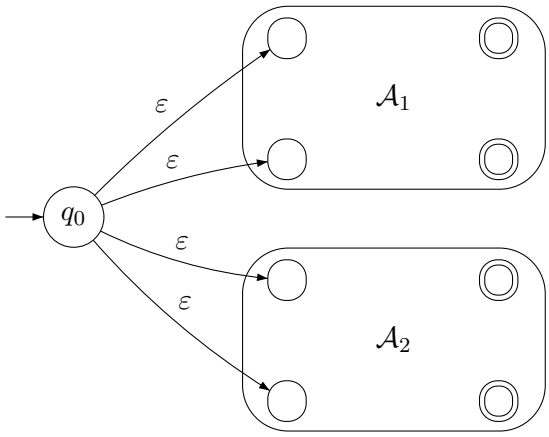
Cet automate reconnaît le mot a, donc $\mathcal{L}(\mathcal{A}) = \{a\} = \mathcal{L}_{\mathcal{R}}(a)$

iii) Soient $\mathcal{A}_1 = (A_1, Q_1, I_1, E_1, T_1)$ et $\mathcal{A}_2 = (A_2, Q_2, I_2, E_2, T_2)$ deux automates reconnaissant respectivement les langages dénotés par les expressions e_1 et e_2. On représentera l'automate $\mathcal{A}_i$ par le schéma :



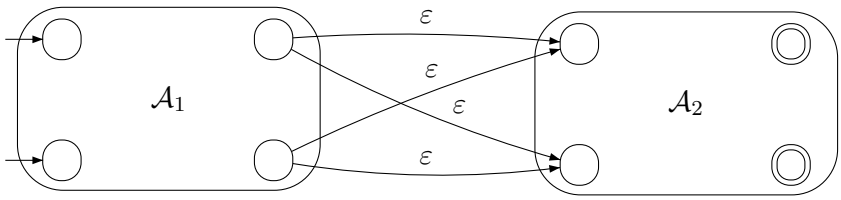
où les cercles de gauche représentes les états initiaux et les cercles à droite représentent les états finaux.

- L'automate $\mathcal{A} = (A, Q, I, E, T)$ reconnaissant $\mathcal{L}_{\mathcal{R}}\big((e_1 + e_2)\big)$ est tel que :



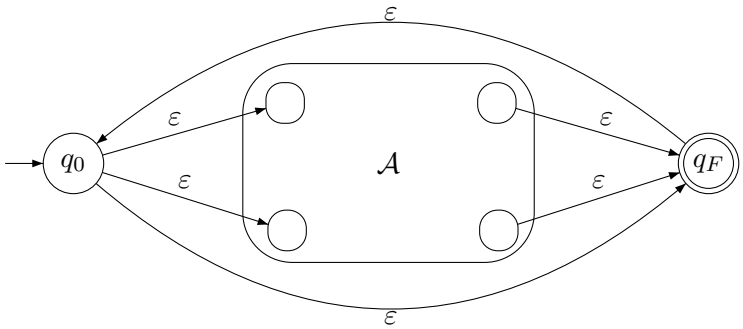
En somme, soit on prend $I = I_1 \cup I_2$, soit on crée un nouvel état initial q_0 tel que, $\forall q \in I_1 \cup I_2,\ q_0 \xrightarrow{\varepsilon} q$. Par ailleurs, $Q = Q_1 \cup Q_2 \cup \{q_0\}$ et $T = T_1 \cup T_2$.

- L'automate $\mathcal{A} = (A, Q, I, E, T)$ reconnaissant $\mathcal{L}_{\mathcal{R}}\big((e_1 e_2)\big)$ est tel que :



$Q = Q_1 \cup Q_2$, $I = I_1$ et $T = T_2$ (les états initiaux de $\mathcal{A}_2$ ne sont plus initiaux et les états finaux de $\mathcal{A}_1$ ne sont plus finaux). il suffit alors de joindre chaque état final de $\mathcal{A}_1$ à chaque état initial de $\mathcal{A}_2$ par une ε-transition.

- Soit $\mathcal{A}$ l'automate reconnaissant le langage dénoté par l'expression rationnelle e. L'automate $\mathcal{A}' = (A', Q', I', E', T')$ reconnaissant $\mathcal{L}_{\mathcal{R}}\big((e)^*\big)$ est tel que :



$Q' = Q \cup \{q_0, q_F\},\ I' = \{q_0\},\ T' = \{q_F\}$ et $E' = E \cup \{q_0 \xrightarrow{\varepsilon} q, q \in I\} \cup \{q \xrightarrow{\varepsilon} q_F, q \in F\} \cup \{q_0 \xrightarrow{\varepsilon} q_F, q_F \xrightarrow{\varepsilon} q_0\}$.

L'automate reconnaissant $\mathcal{L}_{\mathcal{R}}\big((e)^+\big)$ s'obtient, quant à lui, de la même façon que le précédent en retirant la transition $q_0 \xrightarrow{\varepsilon} q_F$ (dont le but est uniquement de faire reconnaître le mot vide par l'automate).

4.2.3 Des automates finis aux expressions rationnelles

4.2.3.1 Langage associé à un état

Définition 2.3 : Soit $\mathcal{L}$ un langage reconnu par un automate fini déterministe $\mathcal{A} = (A, Q, q_0, \delta, T)$. Pour tout couple d'état $(p, q) \in Q^2$, on définit l'ensemble :

$$E_{p,q} = \left\{\alpha \in A^*,\ \delta^*(p, \alpha) = q\right\}$$

ainsi que l'ensemble L_p, défini pour tout état $p \in Q$:

$$L_p = \left\{\alpha \in A^*,\ \exists q \in T,\ \delta^*(p, \alpha) = q\right\}$$

Remarque 2.4 : L_p décrit le langage reconnu par le sous-automate $\mathcal{A}_p = (A, Q, \{p\}, E, T)$.

Remarque 2.5 : Pour tout état $p \in Q$, on note $\gamma_{p,F} = \varepsilon$ si $p \in T$ et $\varnothing$ sinon, on a

$$L_p = \sum_{q \in Q} E_{p,q}.L_q + \gamma_{p,T}$$

4.2.3.2 Lemme de Arden et systèmes d'équations sur les langages

Théorème 2.6 : (Lemme de Arden) Soit A un alphabet et L_1 et L_2 deux langages sur A, on considère l'équation

$$X = L_1.X + L_2$$

où l'inconnue X est un langage sur A.
- Si $\varepsilon \notin L_1$ l'équation admet une solution unique $X = L_1^*.L_2$.
- Si $\varepsilon \in L_1$, l'ensemble des solutions de l'équation est $\{L_1^*.L\ ,\ L_2 \subset L\}$

Théorème 2.7 : Soit $n \in \mathbb{N}^*$, soient $E_{i,j},\ (i, j) \in [1, n]^2$ des langages ne contenant pas ε et soient $F_i,\ i \in [1, n]$ des langages. Le système :

$$\left\{ L_i = \sum_{j=1}^{n} E_{i,j}.L_j + F_i \right\}_{i \in [1,n]}$$

d'inconnues L_i admet une unique solution (et si les langages $E_{i,j}$ et F_i sont réguliers alors les langages L_i le sont aussi).

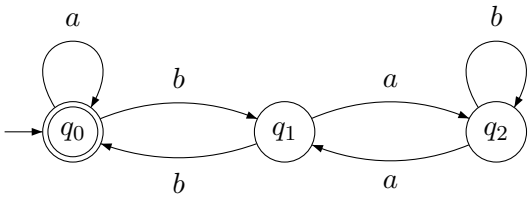
4.2.3.3 Retour sur les automates

Soit $\mathcal{A} = (A, Q, q_0, \delta, T)$ un automate fini déterministe. Nous avons vu précédemment que l'on peut associer à chaque état $p \in Q$ un langage solution de l'équation

$$L_p = \sum_{p \in Q} E_{p,q}.L_q + \gamma_{p,T}$$

Le lemme d'Arden nous assure l'unicité de la solution dans le cas où $\varepsilon \notin E_{p,q}$. Or, comme l'automate $\mathcal{A}$ est déterministe, il n'existe pas de ε-transition, donc $E_{p,q}$ ne peut pas contenir ε, ce qui fait notre fortune. Le langage reconnu par l'automate est alors L_{q_0}.

Exemple 2.8 : Soit $\mathcal{A} = (A, Q, q_0, \delta, T)$ un automate fini déterministe représenté par le graphe suivant :



Pour des raisons de clarté, on note L_i à la place de L_{q_i}. Le système d'équation s'écrit simplement :

$$\begin{cases} L_0 = a.L_0 + b.L_1 + \varepsilon \\ L_1 = b.L_0 + a.L_2 \\ L_2 = a.L_1 + b.L_2 \end{cases}$$

En appliquant une première fois le lemme d'Arden sur la troisième équation, on obtient :

$$L_2 = b^* a L_1$$

On substitue alors L_2 par cette expression dans la seconde équation, ce qui nous permet d'écrire :

$$L_1 = ab^* a L_1 + bL_0$$

qui devient

$$L_1 = (ab^* a)^* bL_0$$

Enfin, en remplaçant L_1 par son expression dans la première équation, on obtient :

$$L = L_0 = (a + b(ab^* a)^* b)^*$$

Remarque 2.9 : On a montré que tout langage rationnel était reconnaissable par un automate fini, donc l'ensemble des langages rationnels est inclus dans l'ensemble des langages reconnaissables par un automate fini. Il a ensuite été montré qu'à tout automate fini déterminste on peut associer une expression rationnelle, montrant ainsi que l'ensemble des langages reconnaissables par un automate fini déterministe est inclus dans l'ensemble des langages rationnels. Autrement dit, si l'on note $Rec_D(A^*)$ l'ensemble des langages sur A reconnaissables par un automate fini déterministe, on a établi la relation suivante :

$$Rec_D(A^*) \subset Rat(A^*) \subset Rec(A^*)$$

Pour avoir l'égalité, il reste à établir que $Rec(A^*) \subset Rec_D(A^*)$, autrement dit, qu'à tout automate fini on peut associer un automate fini déterministe équivalent.

4.3 Déterminisation

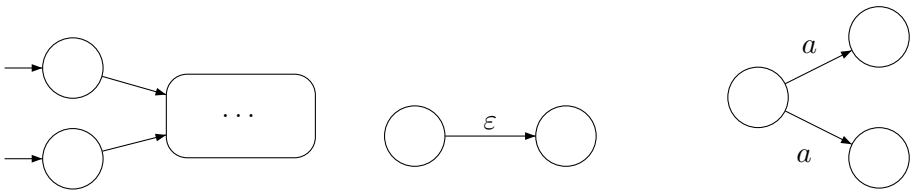
4.3.1 Idées générales

Soit $\mathcal{A} = (A, Q, I, E, T)$ un automate. L'indéterminisme de $\mathcal{A}$ peut se manifester sous les trois formes suivantes :

i) plusieurs états initiaux

ii) une transition vide (entre deux états distincts)

iii) un état qui est l'origine de plusieurs transitions différentes de même étiquette



cas *i*) cas *ii*) cas *iii*)

L'idée est donc d'éliminer l'indéterminisme en :
- regroupant les états initiaux en un seul état initial,
- éliminant les ε-transitions,
- regroupant les états qui peuvent être l'extrêmité de calculs de même origine et même étiquette.

Remarque 3.1 : De cette façon, si S est l'ensemble des états de l'automates fini indéterministe $\mathcal{A}$, l'ensemble des états de l'automate déterministe correspondant $\mathcal{A}'$ sera un sous-ensemble de $\mathcal{P}(S)$. (Chaque état de $\mathcal{A}'$ sera un ensemble d'états de $\mathcal{A}$).

Remarque 3.2 : On peut également envisager le problème sous un autre angle. Nous avons vu précédemment que la différence, formelle, entre un automate fini déterministe et indéterministe résidait dans le fait que I est un singleton et la relation E est une fonction. La déterminisation consiste donc à effectuer cette transformation.

4.3.2 Passage d'une relation à une fonction

Nous allons montrer comment à toute relation on peut associer, de manière unique, une fonction. Pour cela, on rappelle la définition d'une fonction.

Définition 3.3 : Soit $f \subset E \times F$ une relation de E dans F. f est une **fonction** de E dans F si et seulement si tout élément de E possède au plus une (soit zéro, soit une) image dans F par f :

$$\forall x \in E,\ card\big(f(\{x\})\big) \leq 1$$

Proposition 3.4 : Soit $\mathcal{R} \subset E \times F$ une relation de E dans F. La relation définie par :

$$\begin{array}{rccl} \phi: & E & \rightarrow & \mathcal{P}(F) \\ & x & \mapsto & \{y \in F \mid (x,y) \in \mathcal{R}\} \end{array}$$

est une fonction (c'est même une application) et l'application

$$\begin{array}{rccl} \mathcal{T}: & \mathcal{P}(E \times F) & \rightarrow & E \rightarrow \mathcal{P}(F) \\ & \mathcal{R} & \mapsto & \phi : x \mapsto \{y \in F \mid (x,y) \in \mathcal{R}\} \end{array}$$

est bijective.

▷ **Démonstration :** *ϕ est clairement une application puisqu'à tout élément x de E on associe exactement un élément de $\mathcal{P}(F)$. En effet, trois cas sont possibles :*

- *il n'existe aucun y tel que $(x, y) \in \mathcal{R}$: dans ce cas, $\phi(x) = \varnothing \in \mathcal{P}(F)$*
- *il existe exactement un y tel que $(x, y) \in \mathcal{R}$: dans ce cas, $\phi(x) = \{y\} \in \mathcal{P}(F)$*
- *il existe plusieurs images de x, $y_1, \ldots, y_n$: $\forall i \in [1, n]$, $(x, y_i) \in \mathcal{R}$: dans ce cas, $\phi(x) = \{y_1, \ldots, y_n\} \in \mathcal{P}(F)$*

Ensuite, pour montrer que $\mathcal{T}$ est bijective, il suffit de remarquer que sa réciproque est l'application $\mathcal{T}^{-1}$ qui à toute fonction ϕ de E dans $\mathcal{P}(F)$ associe la relation $\mathcal{R}$ telle que $(x, y) \in \mathcal{R}$ ssi $y \in \phi(x)$. ■

Exemple 3.5 : Soient $A = \{a, b, c\}$, $B = \{1, 2\}$ et $F = \{\alpha, \beta, \delta, \gamma, \lambda\}$ et $\mathcal{R} \subset A \times B \times F$ une relation définie par :

$$\mathcal{R} = \{(a, 1, \beta), (a, 1, \delta), (b, 2, \alpha), (b, 2, \beta), (b, 2, \lambda), (c, 1, \gamma), (c, 2, \alpha)\}$$

$\phi = \mathcal{T}(\mathcal{R})$ est une application de $A \times B$ dans $\mathcal{P}(F)$ définie par :

$$\begin{array}{rccl} \phi : & A \times B & \rightarrow & \mathcal{P}(F) \\ & (a, 1) & \mapsto & \{\beta, \delta\} \\ & (a, 2) & \mapsto & \varnothing \\ & (b, 1) & \mapsto & \varnothing \\ & (b, 2) & \mapsto & \{\alpha, \beta, \lambda\} \\ & (c, 1) & \mapsto & \{\gamma\} \\ & (c, 2) & \mapsto & \{\alpha\} \end{array}$$

Remarque 3.6 : On peut également se contenter de définir $\phi = \mathcal{T}(\mathcal{R})$ comme une fonction en ne définissant pas ϕ pour les éléments de E dont l'image est $\varnothing$.

4.3.3 Automate fini déterministe équivalent

Théorème 3.7 : Soit $\mathcal{A}_{nd} = (A, Q, I, E, T)$ un automate fini indéterministe. Si l'on définit l'automate $\mathcal{A} = (A, Q', q_0, \delta, T')$ par :

- $Q' = \mathcal{P}(Q)$
- $q_0 = \bigcup_{i \in I} \phi(i, \varepsilon)$
- $\delta : \left\{ \begin{array}{rcl} Q' \times A & \rightarrow & Q' \\ (s, a) & \mapsto & \bigcup_{q \in s} \phi^*(q, a) \end{array} \right.$
- $T' = \{s \in Q' \mid s \cap T \neq \varnothing\}$

où

$$\begin{array}{rccl} \phi = \mathcal{T}(E) : & Q \times A \cup \{\varepsilon\} & \rightarrow & \mathcal{P}(Q) \\ & (q, a) & \mapsto & \{q' \in Q \mid (q, a, q') \in E\} \end{array}$$

et $\forall, (q, a) \in Q \times A \cup \{\varepsilon\}$, $\phi^*(q, a)$ est l'ensemble défini inductivement par :

- *base* : $\left\{ \begin{array}{l} q \in \phi^*(q, \varepsilon) \\ \forall s \in \phi(q, a),\ q \in \phi^*(q, a) \end{array} \right.$
- *induction* : $\forall s' \in \phi^*(q, a),\ \forall s \in \phi(s', \varepsilon),\ s \in \phi^*(q, a)$

Alors $\mathcal{A}$ est déterministe et équivalent à $\mathcal{A}_{nd}$ (il reconnaît le même langage que $\mathcal{A}_{nd}$).

Remarque 3.8 : On remarque dans ce théorème que l'ensemble des états de l'automate fini déterministe équivalent à $\mathcal{A}_{nd} = (A, Q, I, E, T)$ est $\mathcal{P}(Q)$, donc l'automate possède $2^{card(Q)}$ états ... En pratique, la plupart des états de $\mathcal{P}(Q)$ sont inaccessibles. En ce sens, on utilise un algorithme qui permet, à partir d'un automate fini indéterministe, de construire un AFD équivalent sur la base du théorème précédent mais ne faisant apparaître que les états accessibles. Cet algorithme part de l'état initial q_0 et construit de nouveaux états en calculant $\delta(q, a)\ \forall a \in A$ pour q déjà construit. L'algorithme s'arrête lorsqu'aucun nouvel état n'est généré.

Exemple 3.9 :
Soit l'automate fini indéterministe $\mathcal{A}_{nd} = (A, Q, I, E, T)$ défini par :

- $A = \{a, b\}$
- $Q = \{0, 1, 2, 3, 4\}$
- $I = \{0\}$
- $T = \{4\}$
- $E = \left\{ \begin{array}{l} (0, \varepsilon, 1), (0, \varepsilon, 3), (1, a, 2), (1, a, 3), (2, b, 1), (3, a, 1), \\ (3, a, 3), (3, b, 4), (4, a, 1), (4, a, 2), (4, b, 3), (4, \varepsilon, 3) \end{array} \right\}$

(i) On commence par dresser la table de ϕ :

ϕ	ε	a	b
0	1, 3		
1		2, 3	
2			1
3		1, 3	4
4	3	1, 2	3

$\Rightarrow$

ϕ^*	ε	a	b
0	1, 3		
1		2, 3	
2			1
3		1, 3	3, 4
4	3	1, 2	3

(ii) On construit δ pas à pas. L'état initial est l'ensemble des états directement accessibles à partir de l'ancien état initial, soit $q_0 = \{0, 1, 3\}$. On établit ensuite l'ensemble de tous les états accessibles à partir de 0, 1 et 3 en lisant a puis b : il s'agit simplement de faire l'union du contenu des lignes 0, 1 et 3 du tableau de ϕ^*.

δ	a	b
0, 1, 3	1, 2, 3	3, 4

Deux nouveaux états sont créés : $\{1, 2, 3\}$ et $\{3, 4\}$.

δ	a	b
0, 1, 3	1, 2, 3	3, 4
1, 2, 3		
3, 4		

On poursuit ainsi jusqu'à ce qu'il n'y ait plus de nouvel état créé :

δ	a	b
0, 1, 3	1, 2, 3	3, 4
1, 2, 3	1, 2, 3	1, 3, 4
3, 4	1, 2, 3	3, 4
1, 3, 4	1, 2, 3	3, 4

(iii) Les états finaux sont tous les états qui contiennent un élément de T (ici 4), donc $T' = \big\{\{3, 4\}, \{1, 3, 4\}\big\}$

L'automate déterminite équivalent à $\mathcal{A}_{nd}$ est ainsi entièrement construit (on connaît $A,\ Q',\ q_0,\ \delta$ et T').

4.3.4 Conclusion

Remarque 3.10 : Cette dernière partie relative à la déterminisation des automates finis indéterministes nous permet de conclure avec la relation suivante :

$$Rec(A^*) = Rec_D(A^*) = Rat(A^*)$$

pour tout alphabet A.

Remarque 3.11 : On a désormais trois caractérisations différentes des langages rationnels sur un alphabet A :
- la définition, *i.e.* un langage est rationnel s'il est fini ou l'union, l'itéré ou le produit de langages rationnels
- un langage est rationnel ssi il est dénoté par une expression rationnelle
- un langage est rationnel ssi il est reconnu par un automate fini (déterministe ou indéterministe)

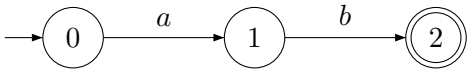
4.4 Minimalisation

4.4.1 Relation d'équivalence sur l'ensemble des états

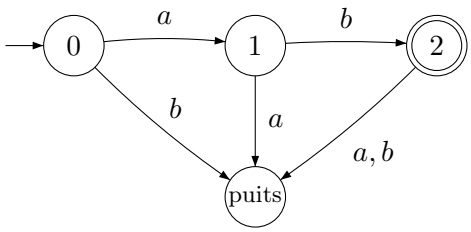
On chercher à déterminer un algorithme qui, à partir d'un automate déterministe, construit l'automate minimum (*i.e* comportant un nombre minimal d'états) équivalent (*i.e.* reconnaissant le même langage). Pour établir un tel algorithme, nous allons définir et étudier une relation d'équivalence entre les états.

Définition 4.1 : Soit $\mathcal{A} = (A, Q, q_0, \delta, T)$ un automate fini déterministe. On dit que A est complet ssi la fonction δ est une application, autrement dit, si $\delta(q, a)$ est définie pour tous les couples (q, a) de $Q \times A$.

Remarque 4.2 : Pour trouver un AFD complet équivalent à un AFD incomplet, il suffit d'ajouter un état puits (état de refus) qui sera l'extrêmité de toutes les transitions non définies. Soit par exemple l'AFD non complet représenté par le graphe suivant.



δ n'est pas définie pour tous les couples : $\delta(0, b)$ n'est pas défini. Un automate fini déterministe complet équivalent est donné par le graphe suivant :



Définition 4.3 : Pour chaque état $q \in Q$, on note :

$$L_q = \{\alpha \in A^* \mid \delta^*(q, \alpha) \in T\}$$

L_q s'interprète comme étant le langage reconnu par l'automate lorsque q est l'état initial.

Remarque 4.4 : On remarquera que $\varepsilon \in L_q \Leftrightarrow q \in T$

Définition 4.5 : On considère un AFD complet $\mathcal{A} = (A, Q, q_0, \delta, T)$. On définit sur Q la relation d'équivalence $\sim$ par :

$$\begin{aligned} p \sim q &\Leftrightarrow L_p = L_q \\ &\Leftrightarrow \forall \alpha \in A^*,\ (\delta^*(p, \alpha) \in T \Leftrightarrow \delta^*(q, \alpha) \in T) \end{aligned}$$

Définition 4.6 : On note appelle classe d'équivalence d'un état q et on note $\langle\ q\ \rangle$ l'ensemble :

$$\langle\ q\ \rangle = \{p \in Q \mid p \sim q\}$$

On note $\widetilde{Q}$ l'ensemble des classes d'équivalence : $\widetilde{Q} = \{\langle\ q\ \rangle \mid q \in Q\}$.

Proposition 4.7 : $\forall (p, q) \in Q^2,\ \forall \alpha \in A^*$:

$$p \sim q \quad \Leftrightarrow \quad \delta^*(p, \alpha) \sim \delta^*(q, \alpha)$$

▷ **Démonstration :** *Soient p et $q \in Q$.*

$$\begin{aligned} \forall \alpha \in A^*,\ \delta^*(p, \alpha) \sim \delta^*(q, \alpha) &\Leftrightarrow \forall \alpha \in A^*,\ L_{\delta^*(p,\alpha)} = L_{\delta^*(q,\alpha)} \\ &\Leftrightarrow \forall \alpha \in A^*,\ \forall \beta \in A^*,\ \big(\delta^*(\delta^*(p, \alpha), \beta) \in T \ \Leftrightarrow\ \delta^*(\delta^*(q, \alpha), \beta) \in T\big) \\ &\Leftrightarrow \forall \alpha, \beta \in A^*,\ \big(\delta^*(p, \alpha\beta) \in T \ \Leftrightarrow\ \delta^*(q, \alpha\beta) \in T\big) \\ &\Leftrightarrow \forall \gamma \in A^*,\ \big(\delta^*(p, \gamma) \in T \ \Leftrightarrow\ \delta^*(q, \gamma) \in T\big) \\ &\Leftrightarrow p \sim q \end{aligned}$$

■

Remarque 4.8 : On définit ainsi une application $\widetilde{\delta}$ de $\widetilde{Q} \times A$ dans $\widetilde{Q}$ en posant :

$$\widetilde{\delta}(\langle\ q\ \rangle, a) = \langle\ \delta(q, a)\ \rangle$$

En effet, l'énoncé ci-dessus est équivalent à : $\forall p \in \langle\ q\ \rangle,\ \delta(p, a) \in \langle\ \delta(q, a)\ \rangle$, autrement dit :

$$p \sim q \quad \Rightarrow \quad \delta(p, a) \sim \delta(q, a)$$

ce qui est l'objet de la démonstration précédente.

Remarque 4.9 : On montre par induction structurelle que l'extension $\widetilde{\delta}^*$ de $\widetilde{\delta}$ à $\widetilde{Q} \times A^*$ vérifie : $\forall (q, \alpha) \in \widetilde{Q} \times A^*$

$$\widetilde{\delta}^*(\langle\ q\ \rangle, \alpha) = \langle\ \delta^*(q, \alpha)\ \rangle$$

▷ **Démonstration :** *On précise que l'extension de $\widetilde{\delta}$, notée $\widetilde{\delta}^*$ est définie par, $\forall \langle\ q\ \rangle \in \widetilde{Q}$:*

$$\begin{cases} \widetilde{\delta}^*(\langle\ q\ \rangle, \varepsilon) = \langle\ q\ \rangle \\ \forall (a,\beta) \in A \times A^*,\ \widetilde{\delta}^*(\langle\ q\ \rangle, a\beta) = \widetilde{\delta}^*(\widetilde{\delta}(\langle\ q\ \rangle, a), \beta) \end{cases}$$

On procède par induction structurelle sur A^* :

- *Base : $\alpha = \varepsilon$*
  $\widetilde{\delta}^*(\langle\ q\ \rangle, \varepsilon) = \langle\ q\ \rangle = \langle\ \delta^*(q, \varepsilon)\ \rangle$
- *Induction : On suppose que $\forall q \in \widetilde{Q},\ \widetilde{\delta}^*(\langle\ q\ \rangle, \beta) = \langle\ \delta^*(q,\beta)\ \rangle$ et on cherche à montrer que cette égalité est vraie pour $\alpha = a.\beta,\ \forall a \in A$*

$$\begin{aligned} \widetilde{\delta}^*(\langle\ q\ \rangle, \alpha) &= \widetilde{\delta}^*(\langle\ q\ \rangle, a\beta) \\ &= \widetilde{\delta}^*\left(\widetilde{\delta}(\langle\ q\ \rangle, a), \beta\right) \\ &\overset{2^e}{=} \widetilde{\delta}^*(\langle\ \delta(q,a)\ \rangle, \beta) \\ &\overset{HI}{=} \langle\ \delta^*(\delta(q,a), \beta)\ \rangle \\ &= \langle\ \delta^*(q, a\beta)\ \rangle \\ &= \langle\ \delta^*(q, \alpha)\ \rangle \end{aligned}$$

■

Remarque 4.10 : On montre également que si $t \in T$ et $t \sim q$ alors $q \in T$. En effet, si $t, q \in Q$ et $t \sim q$, càd $L_t = L_q$, alors :

$$t \in T \Leftrightarrow \varepsilon \in L_t \Leftrightarrow \varepsilon \in L_q \Leftrightarrow q \in T$$

Proposition 4.11 : On pose $\widetilde{q_0} = \langle\ q_0\ \rangle$ et $\widetilde{T} = \{\langle\ t\ \rangle \mid t \in T\}$. L'automate $\widetilde{\mathcal{A}} = (A, \widetilde{Q}, \widetilde{q_0}, \widetilde{\delta}, \widetilde{T})$ est un AFD complet reconnaissant le même langage que A.

▷ **Démonstration :** *$\widetilde{\delta}$ étant une application de $\widetilde{Q} \times A$ dans $\widetilde{Q}$, $\widetilde{\mathcal{A}}$ est un AFD complet. Soit $\alpha \in A^*$:*

$$\begin{aligned} \alpha \in \mathcal{L}(\widetilde{\mathcal{A}}) &\Leftrightarrow \widetilde{\delta}^*(\widetilde{q_0}, \alpha) \in \widetilde{T} \\ &\Leftrightarrow \widetilde{\delta}^*(\langle\ q_0\ \rangle, \alpha) \in \widetilde{T} \\ &\Leftrightarrow \langle\ \delta^*(q_0, \alpha)\ \rangle \in \widetilde{T} \\ &\Leftrightarrow \exists t \in T \mid \delta^*(q_0, \alpha) \in \langle\ T\ \rangle \\ &\Leftrightarrow \exists t \in T \mid \delta^*(q_0, \alpha) \sim t \\ &\Leftrightarrow \delta^*(q_0, \alpha) \in T \quad \textit{d'après 4)} \\ &\Leftrightarrow \alpha \in \mathcal{L}(\mathcal{A}). \end{aligned}$$

■

Remarque 4.12 : Si A est un AFD complet et monogène (*i.e.* si tous ses états sont accessibles), alors $\widetilde{\mathcal{A}}$ est l'automate minimal reconnaissant le même langage que A.

4.4.2 Algorithme

Nous allons présenter un algorithme itératif qui permet de déterminer à partir d'un automate fini déterministe monogène les différentes classes d'équilence sur l'ensemble des états et ainsi définir l'automate fini déterministe équivalent à ce dernier.

Algorithme 4.13 : Soit $\mathcal{A} = (A, Q, q_0, \delta, T)$un automate fini déterministe monogène.

- **Initialisation**
  Définir $\sim_0$ de la façon suivante : $\forall q, q' \in Q,\ q \sim_0 q'$ ssi
  $$q \in T \quad \Leftrightarrow \quad q' \in T$$
- **Jusqu'à ce que** $\sim_{i+1} = \sim_i$
  Définir $\sim_{i+1}$ à partir de $\sim_i$ de la façon suivante : $\forall q, q' \in Q,\ q \sim_{i+1} q'$ ssi
  $$q \sim_i q' \ \text{ et } \ \forall a \in A,\ \delta(q,a) \sim_i \delta(q',a)$$

Remarque 4.14 : En pratique, on représente la relation d'équivalence par un tableau dans lequel chaque colonne représente une classe d'équivalence. A l'initialisation, le tableau comporte deux colonnes, une pour la classe d'équivalente T et l'autre pour $Q \setminus T$. D'une itération à l'autre, deux éléments appartenant précédemment à deux colonnes différentes ($q \nsim_i q'$) ne peuvent pas se retrouver dans une même colonne ($q \sim_{i+1} q'$) : chaque colonne est donc scindée de nouveau par la règle : $\forall a \in A,\ \delta(q,a) \sim_i$. On arrête l'algorithme dès que deux tableaux successifs sont identiques.

Exemple 4.15 : Soit $\mathcal{A} = (A, Q, q_0, \delta, T)$ l'automate fini déterministe monogène complet défini par :

- $A = \{a, b\}$
- $Q = \{1, \ldots, 8\}$
- $q_0 = 0$
- δ définie par la table suivante :

δ	0	1	2	3	4	5	6	7	8
a	1	6	8	7	5	2	8	2	2
b	6	5	4	8	2	7	3	5	0

- $T = \{2, 6, 8\}$

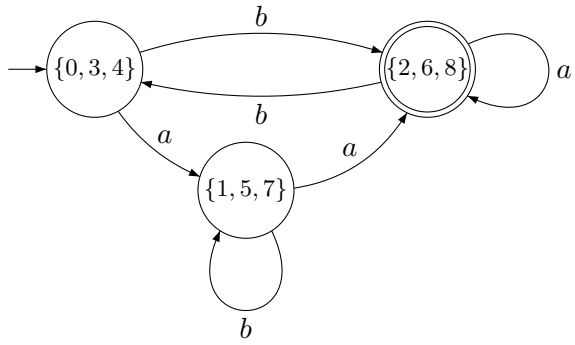
Le déroulement de l'algorithme de minimalisation est le suivant :

$$\sim_0 = \begin{array}{|cccccc|}\hline 0 & 1 & 3 & 4 & 5 & 7 \\ \hline 2 & 6 & 8 & & & \\ \hline \end{array}$$

$$\sim_1 = \begin{array}{|ccc|}\hline 0 & 3 & 4 \\ \hline 1 & 5 & 7 \\ \hline 2 & 6 & 8 \\ \hline \end{array}$$

$$\sim_2 = \begin{array}{|ccc|}\hline 0 & 3 & 4 \\ \hline 1 & 5 & 7 \\ \hline 2 & 6 & 8 \\ \hline \end{array}$$

Les relations d'équivalence $\sim_1$ et $\sim_2$ sont égales, l'algorithme s'arrête et les trois classes d'équivalence obtenues sont les états de l'automate minimal :



4.5 Travaux Dirigés

4.5.1 Exercice 1

Construire des automates finis indéterministes reconnaissant les langages dénotés par les expressions suivantes :

- $e_1 = (a + b)^*$
- $e_2 = (a^*b)^*$
- $e_3 = (a^*b + bb^*a)^* + (baa)^+(ab)^*$

4.5.2 Exercice 2

Le mot *aabba* (resp. *baab*) appartient-il à $\mathcal{L}_{\mathcal{R}}(e_3)$? Si oui, donnez un calcul réussi dont il est l'étiquette.

4.5.3 Exercice 3

Soit $A = \{a, b\}$ un alphabet et L un langage sur A dénoté par l'expression $e = A^*abaA^*$.

a) Décrire le langage dénoté par e.

b) Construire un automate fini A indéterministe reconnaissant L (précisez en quoi il est indéterministe).

c) Construire un automate fini déterministe équivalent à A.

4.5.4 Exercice 4

Soit $\mathcal{A} = (A, Q, q_0, \delta, T)$ un automate fini déterministe sur $A = \{a, b\}$ défini par :
$Q = \{0, 1, \ldots, 10\}$, $q_0 = 0$, $T = \{2, 6, 8\}$ et $\delta : Q \times A \to Q$ décrite par la table suivante :

δ	0	1	2	3	4	5	6	7	8	9	10
a	1	6	8	7	5	2	8	2	2	1	7
b	6	5	4	8	2	7	10	5	0	2	8

a) Après avoir vérifié que $\mathcal{A}$ est déterministe, supprimer les états inaccessibles.

b) Minimaliser cet automate et dessiner le graphe sagittal de l'automate minimal obtenu.

c) Donnez une expression rationnelle (régulière) dénotant $\mathcal{L}(\mathcal{A})$.

4.5.5 Exercice 5

Le but de ce problème est l'étude des propriétés des opérations de dérivation à gauche $m^{-1}.\mathcal{A}$ et à droite $\mathcal{A}.m^{-1}$ d'un automate fini $\mathcal{A}$ selon un mot m. Pour simplifier les preuves, nous nous limiterons au cas des automates finis semi-indéterministes, c'est-à-dire les automates finis non déterministes qui ne contiennent pas de transitions instantanées (ou ε-transitions). Les résultats étudiés s'étendent au cadre des automates finis quelconques.

Définition 5.6 : Un automate fini **semi-indéterministe** sur un alphabet X est un quintuplet $\mathcal{A} = (X, Q, I, \gamma, T)$ composé de :
- Un ensemble fini d'états : Q ;
- Un ensemble d'états initiaux : $I \subseteq Q$;
- Un ensemble d'états terminaux : $T \subseteq Q$;
- Une relation de transition confondue avec son graphe : $\gamma \subseteq Q \times X \times Q$.

(*i.e.* un automate fini indéterministe ne comportant pas de ε-transition.

Remarquons que γ est le graphe d'une application de transition $\delta : Q \times X \to \mathcal{P}(Q)$ dont les valeurs sont définies par

$$\forall\, o \in Q,\ \forall\, e \in X,\ \delta(o, e) = \{d \in Q \mid (o, e, d) \in \gamma\}$$

La notation γ est plus adaptée que δ à la formalisation et la construction des preuves dans le cadre des automates indéterministes.

a) Soit $\mathcal{E} = (X, Q, I, \gamma, T)$ un automate fini semi-indéterministe tel que :

$$Q = \{A, B, C, D, E\},\quad X = \{a, b\},\quad I = \{A, B\},\quad T = \{D, E\}$$
$$\gamma = \{(A, a, C), (A, b, D), (B, a, D), (B, b, B), (C, b, E), (D, a, C), (E, a, C)\}$$

Donnez une représentation sagittale de $\mathcal{E}$.

b) En quoi $\mathcal{E}$ est semi-indéterministe et pas déterministe ?

c) On note γ^* le prolongement de γ à $Q \times X^* \times Q$ défini inductivement par :
- $\forall\, q \in Q,\ (q, \varepsilon, q) \in \gamma^*$
- $\forall\, e \in X,\ \forall\, m \in X^*,\ \forall\, o \in Q,\ \forall\, d \in Q$:

$$\big((o, e.m, d) \in \gamma^*\big) \Leftrightarrow \big(\exists q \in Q,\ ((o, e, q) \in \gamma) \wedge ((q, m, d) \in \gamma^*)\big)$$

On rappelle que le langage sur X^* reconnu par l'automate $\mathcal{A}$ est :

$$\mathcal{L}(\mathcal{A}) = \{m \in X^* \mid \exists\, o \in I,\ \exists\, d \in T, (o, m, d) \in \gamma^*\}$$

Déterminez une expression rationnelle (régulière) ou ensembliste dénotant $\mathcal{L}(\mathcal{E})$.

d) On considère désormais les opérations internes sur les automates finis déterministes définies par :

Définition 5.7 : (Dérivées selon un mot) Soient $\mathcal{A} = (X, Q, I, \gamma, T)$ un automate fini semi-indéterministe et $m \in X^*$, les automates $m^{-1}.\mathcal{A}$ (dérivation à gauche selon m) et $\mathcal{A}.m^{-1}$ (dérivation à droite selon m) sont définis par :

$$\begin{aligned} m^{-1}.\mathcal{A} &= (X, Q, \{q \in Q \mid \exists\, i \in I, (i, m, q) \in \gamma^*\}, \gamma, T) \\ \mathcal{A}.m^{-1} &= (X, Q, I, \gamma, \{q \in Q \mid \exists\, t \in T, (q, m, t) \in \gamma^*\}) \end{aligned}$$

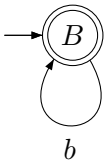
Construire les automates $a^{-1}.\mathcal{E}$, $b^{-1}.\mathcal{E}$, $\mathcal{E}.a^{-1}$ et $\mathcal{E}.b^{-1}$ (seuls les états et les transitions utiles, c'est-à-dire accessibles depuis les états initiaux, devront être construits).

Définition 5.8 : Soit $\mathcal{A} = (Q, X, I, T, \gamma)$ un automate fini semi-indéterministe. On appelle **état de refus** tout état $q \in Q$ tel qu'il n'existe pas de calcul d'origine q et d'extrémité $e \in T$:

$$\nexists (m, e) \in X^* \times T,\ (q, m, e) \in \gamma^*$$

(Il n'est pas utile de représenter les états de refus lorsque l'on effectue la représentation sagittale d'un automate.)

e) Montrez que l'automate $\mathcal{E}.a^{-1}$ est équivalent à l'automate suivant :



f) En déduire les expressions rationnelles dénotant $\mathcal{L}(\mathcal{E}.a^{-1})$, $\mathcal{L}(a^{-1}.\mathcal{E})$, $\mathcal{L}(b^{-1}.\mathcal{E})$ et $\mathcal{L}(\mathcal{E}.b^{-1})$.

g) Soit $\mathcal{A}$ un automate fini semi-indéterministe. On rappelle que, par définition de γ^*, on a, $\forall\, m, n \in X^*,\ \forall\, o, d \in Q$:

$$\Big(\exists\, q \in Q,\ (o, m, q) \in \gamma^* \wedge (q, n, d) \in \gamma^*\Big) \ \Leftrightarrow\ (o, m.n, d) \in \gamma^*.$$

Montrez que :

- $\forall\, m \in X^*,\ \forall\, n \in X^*,\ n \in \mathcal{L}(m^{-1}.\mathcal{A}) \Leftrightarrow m.n \in \mathcal{L}(\mathcal{A})$
- $\forall\, m \in X^*,\ \forall\, n \in X^*,\ n \in \mathcal{L}(\mathcal{A}.m^{-1}) \Leftrightarrow n.m \in \mathcal{L}(\mathcal{A})$

Chapitre 5

Langages algébriques

5.1 Grammaires algébriques

Not yet implemented

5.2 Langages algébriques et rationnels

5.2.1 Rappels

Définition 2.1 : On appelle **grammaire algébrique** tout quadruplet

$$G = (N, T, \rightarrow, X)$$

où
- N est un ensemble fini appelé ensemble des **non terminaux**
- T est un alphabet appelé ensemble des **terminaux** tel que $N \cap T = \varnothing$
- $\rightarrow$ est une relation entre N et $(N \cup T)^*$ appelée **relation de production** telle que $\forall A \in N$:
$$\{\alpha \in (N \cup T)^* \mid A \rightarrow \alpha\} \text{ est fini}$$
- $X \in N$ est l'**axiome** de la grammaire

Remarque 2.2 : L'objectif d'une grammaire algébrique est de décrire un langage. On dit qu'une grammaire G **engendre** un langage et l'on note ce langage $\mathcal{L}(G)$.

Exemple 2.3 : Soit $G_1 = (N, T, \rightarrow, X)$ avec
- $N = \{①, ②\}$
- $T = \{a, b\}$
- $\rightarrow$ définie par :
$$\left\{ \begin{array}{lcl} ① & \rightarrow & a\,① \\ ① & \rightarrow & b\,② \\ ② & \rightarrow & b\,② \\ ② & \rightarrow & \varepsilon \end{array} \right\}$$
- $X = ①$

$\mathcal{L}(G_1)$ est le langage rationnel dénoté par l'expression a^*bb^*.

Exemple 2.4 : Soit $G_2 = (N, T, \rightarrow, X)$ avec
- $N = \{①\}$
- $T = \{a, b\}$
- $\rightarrow$ définie par :
$$\left\{ \begin{array}{lcl} ① & \rightarrow & a\,①\,b \\ ① & \rightarrow & \varepsilon \end{array} \right\}$$
- $X = ①$

$\mathcal{L}(G_2)$ est le langage $\{a^n b^n \mid n \geq 0\}$.

5.2.2 Les grammaires algébriques vues comme des automates

On rappelle la définition d'un automate fini, dont le but est également de décrire un langage :

Définition 2.5 : On appelle automate fini (AF) tout quintuplet

$$\mathcal{A} = (A, Q, I, E, T)$$

où

- A est un **alphabet**
- Q est un ensemble fini, l'ensemble des **états**
- $I \subset Q$ est l'ensemble des **états initiaux**
- $T \subset Q$ est l'ensemble des **états finaux (terminaux)**
- E est une relation de $Q \times A \cup \{\varepsilon\}$ dans Q appelée **relation transition**

On s'aperçoit que le fonctionnement d'une grammaire algébrique est « similaire » à celui d'un automate.

Exemple 2.6 : On reprend la grammaire $G_1 = (N, T, \rightarrow, X)$:
- $N = \{①,\ ②\}$
- $T = \{a, b\}$
- $\rightarrow$ définie par :

$$\left\{ \begin{array}{lcl} ① & \rightarrow & a\,① \\ ① & \rightarrow & b\,② \\ ② & \rightarrow & b\,② \\ ② & \rightarrow & \varepsilon \end{array} \right\}$$

- $X = ①$

et on cherche à savoir si le mot $abbb$ appartient à $\mathcal{L}(G_1)$:
- on part de l'axiome ①
- on utilise la première règle de production : on lit le caractère a et l'on est renvoyé au non-terminal ①, *i.e* on s'autorise à utiliser une règle de production dont le membre gauche est ①.
- on utilise la seconde règle de production : on lit le caractère b et l'on est renvoyé au non-terminal ②
- on utilise la troisième règle de production : on lit le caractère b et l'on est renvoyé au non-terminal ②
- on utilise de nouveau la troisième règle de production : on lit le caractère b et l'on est renvoyé au non-terminal ②
- on utilise la quatrième règle de production : on lit le caractère vide et on s'arrête.

Le fonctionnement de la grammaire semble similaire à celui d'un automate fini ! En effet, on peut faire l'analogie suivante :

Automate		**Grammaire**
ensemble des états Q	$\longleftrightarrow$	ensemble des non-terminaux N
alphabet A	$\longleftrightarrow$	ensemble des terminaux T
relation transition	$\longleftrightarrow$	relation de production
états initiaux I	$\longleftrightarrow$	axiome X
états finaux F	$\longleftrightarrow$	non-terminaux produisant un mot de T^*

Remarque 2.7 : On s'aperçoit qu'un **calcul réussi** dans un automate correspond à une **dérivation** pour une grammaire. En effet,

$$X = ① \rightarrowtail a① \rightarrowtail ab② \rightarrowtail abb② \rightarrowtail abbb② \rightarrowtail abbb\varepsilon = abbb$$

Cette dérivation peut être vue comme un calcul réussi d'étiquette $abbb$.

Remarque 2.8 : Continuons notre comparaison avec les automates finis. On rappelle qu'un automate fini est indéterministe dès lors que pour un même mot α, on peut trouver plusieurs calculs réussis différents d'étiquette α et qu'une grammaire est dite ambigüe dès

que pour un mot α on peut trouver plusieurs dérivations générant α. On a donc correspondance entre l'**indéterminisme** des automates et **ambigüité** des grammaires.

Remarque 2.9 : Pour cette partie nous avons utilisé une notation peu standard pour les non-terminaux afin de mettre en évidence la correspondance entre ces derniers et les états d'un automate. A partir de maintenant, nous représenterons les non-terminaux par des majuscules et nous regrouperons les règles

$$\left\{ \begin{array}{lcl} A & \rightarrow & \alpha \\ A & \rightarrow & \beta \end{array} \right.$$

en

$$A \rightarrow \alpha \mid \beta$$

5.2.3 Transformations de grammaires

De la même façon que nous avions défini les états inaccessibles et les états puits sur les automates, nous allons définir les non-terminaux inaccessibles et improductifs.

5.2.3.1 Non-terminaux improductifs

On rappelle que dans un automate, on appelle **état puits** (ou état de refus) tout état qui ne peut pas être l'origine d'un calcul menant à un état final, autrement dit un état qui ne permet plus d'« accéder » à un état final. En ce souvenant que nous avons défini une correspondance entre les états finaux et non-terminaux produisant un mot de T^*, on obtient la définition suivante :

Définition 2.10 : Soit $G = (N, T, \rightarrow, X)$ une grammaire. Un non-terminal $A \in N$ est dit **improductif** s'il n'existe pas de mot $\alpha \in T^*$ tel que

$$A \overset{*}{\rightarrowtail} \alpha$$

Dans le cas contraire, A est dit **productif**.

Exemple 2.11 : Soit $G = (N = \{A, B, C\}, T = \{a, b\}, \rightarrow, A)$ avec $\rightarrow$ définie par :

$$\left\{ \begin{array}{lcl} A & \rightarrow & b\,B \mid a\,C \\ B & \rightarrow & b \\ C & \rightarrow & a\,C \end{array} \right\}$$

On s'aperçoit que C est improductif. En effet, le seul non-terminal « final » est B et une fois que l'on a le non-terminal C, il est impossible d'« accéder » à B.

Remarque 2.12 : On cherche donc un algorithme permettant de supprimer tous les éléments improductifs de la grammaire. On procède comme suit :

- on calcule les non-terminaux productifs
- on en déduit (par complémentaire) les non-terminaux improductifs
- on supprime toutes les règles de production contenant un non-terminal improductif en partie gauche ou droite

Algorithme 1 Calcul des non-terminaux productifs

Entrée : Grammaire algébrique : $G = (N, T, \rightarrow, X)$
Sortie : Ensemble des non-terminaux de G productifs : $Prod$
Initialisation : $Prod \leftarrow \varnothing$
  pour tout $A \rightarrow \alpha$ où $\alpha \in T^*$ **faire**
    $Prod \leftarrow Prod \cup \{A\}$
  fin pour
  tantque $New \neq \varnothing$ **faire**
    $New \leftarrow \varnothing$
    pour tout $A \rightarrow \alpha$ tel que $A \notin Prod$ **faire**
      si $\alpha \in (T \cup Prod)^*$ **alors**
        $New \leftarrow New \cup \{A\}$
      fin si
    fin pour
    $Prod \leftarrow Prod \cup New$
  fin tantque

La recherche des non-terminaux productifs se fait par l'algorithme suivant :

Exemple 2.13 : En reprenant la grammaire de l'exemple précédent, on a :
- Initialisation : $Prod \leftarrow \{B\}$ car $B \rightarrow b$ et $b \in T^*$
- Itération 1 : $Prod \leftarrow Prod \cup \{A\}$ car $A \rightarrow b\,B$ et $b \in T^*,\ B \in Prod$
- Itération 2 : $Prod \leftarrow Prod \cup \varnothing$
- fin

Donc $Prod = \{A, B\}$ donc le non-terminal C est improductif. On peut donc réduire la relation de production de G à l'ensemble suivant :

$$\left\{ \begin{array}{lcl} A & \rightarrow & b\,B \\ B & \rightarrow & b \end{array} \right\}$$

5.2.3.2 Non-terminaux inaccessibles

Définition 2.14 : Soit $G = (N, T, \rightarrow, X)$ une grammaire. Un non-terminal $A \in N$ est dit **inaccessible** s'il n'existe pas $\alpha, \beta \in (N \cup T)^*$ tel que

$$X \overset{*}{\rightarrowtail} \alpha A \beta$$

Sinon, A est dit **accessible**.

Remarque 2.15 : Comme pour les éléments improductifs, on cherche donc un algorithme permettant de supprimer tous les éléments inaccessibles de la grammaire. On procède comme suit :
- on calcule les non-terminaux accessibles
- on en déduit (par complémentaire) les non-terminaux inaccessibles
- on supprime toutes les règles de production contenant un non-terminal inaccessible en partie gauche ou droite

Algorithme 2 Calcul des non-terminaux accessibles

Entrée : Grammaire algébrique : $G = (N, T, \rightarrow, X)$
Sortie : Ensemble des non-terminaux de G accessibles : Acc
Initialisation : $Acc \leftarrow \{X\}$
  tantque $New \neq \varnothing$ **faire**
    $New \leftarrow \varnothing$
    pour tout $A \rightarrow \alpha_1 B_1 \alpha_2 \ldots \alpha_n B_n \alpha_{n+1}$ tel que $A \in Acc$ et $\alpha_i \in T^*,\ B_i \in N$ **faire**
      pour tout $B_i \notin Acc$ **faire**
        $New \leftarrow New \cup B_i$
      fin pour
    fin pour
    $Acc \leftarrow Acc \cup New$
  fin tantque

La recherche des non-terminaux accessibles se fait par l'algorithme suivant :

Exemple 2.16 : Soit $G = (N = \{X, Y, A, B, C, D\}, T = \{a, b, c, d\}, \rightarrow, X)$ avec $\rightarrow$ définie par :

$$\left\{ \begin{array}{lcl} X & \rightarrow & Y \\ Y & \rightarrow & Y\,D \mid Y\,a \mid b \\ A & \rightarrow & B \\ B & \rightarrow & B\,d \mid d \\ C & \rightarrow & c \\ D & \rightarrow & D\,C \end{array} \right\}$$

On a :
- Initialisation : $Acc \leftarrow \{X\}$
- Itération 1 : $Acc \leftarrow \{Y\}$
- Itération 2 : $Acc \leftarrow \{D\}$
- Itération 3 : $Acc \leftarrow \{C\}$
- Itération 4 : $Acc \leftarrow \varnothing$
- fin

On peut donc supprimer les non-terminaux A et B ainsi que toutes les règles les contenant.

Définition 2.17 : Une grammaire est dite **réduite** si tous ses non-terminaux sont productifs et accessibles.

Remarque 2.18 : Réduire une grammaire consiste à supprimer d'abord les non-terminaux improductifs puis les non-terminaux inaccessibles.

5.2.4 Positionnement du problème

On vient d'amorcer la comparaison entre les automates finis et les grammaires algébriques. On se pose donc la question suivante : les grammaires algébriques sont-elles équivalentes aux automates finis ? Autrement dit, les grammaires algébriques décrivent-elles la même classe de langages que les automates finis : les langages rationnels (réguliers) ? Si l'on qualifie d'algébriques les langages engendrés par les grammaires algébriques, la question se formule de la façon suivante :

$$\textbf{algébrique} \stackrel{?}{=} \textbf{rationnel}$$

Pour répondre à cette question, il faut se donner un outil pour vérifier qu'un langage est ou n'est pas rationnel.

5.3 Lemme de l'étoile

5.3.1 Observations

On rappelle la définition de l'ensemble des langages rationnels :

Définition 3.1 : Soient $\mathcal{P}(A^*)$ l'ensemble des langages sur un alphabet quelconque A. L'ensemble des langages rationnels (réguliers) $Rat(A^*)$ est le plus petit sous-ensemble de $\mathcal{P}(A^*)$ vérifiant :

i) **base** : tous les langages finis appartient à $Rat(A^*)$

ii) **induction** : si L, L_1 et L_2 appartiennent à $Rat(A^*)$, alors $L_1 \cup L_2$, $L_1.L_2$ et L^* appartiennent à $Rat(A^*)$

où les symboles « $\cup$ », « . » et « * » représentent respectivement les opérations union, produit et itéré.

On rappelle également le résultat fondamental suivant, appelé théorème de Kleene :

Théorème 3.2 : Un langage est rationnel si et seulement si il est reconnu par un automate fini.

Remarque 3.3 : Les résultats que nous allons établir se fondent sur les quelques observations suivantes :

- Tous les langages finis (comportant un nombre fini de mots) sont rationnels.
- Un langage non rationnel comporte donc nécessairement un nombre infini de mots.
- Pour tout langage comportant un nombre infini de mots, il n'existe pas de borne à la taille des mots du langage[1].
- Tout langage rationnel est reconnu par un automate déterministe comportant un nombre fini d'état.
- Si l'on considère un langage rationnel infini et un automate fini à n états reconnaissant ce langage, pour tout mot α appartenant au langage de longueur supérieure à n, le calcul réussi d'étiquette α passe nécessairement au moins deux fois par le même état (lemme des tiroirs). Autrement dit, si on appelle s cet état, i l'état initial de l'automate et f l'état final qui est l'extrémité du calcul, il existe $x, \beta, y \in A^*,\ \beta \neq \varepsilon$ tel que $\alpha = x\beta y$ et

$$i \xrightarrow{x} s \xrightarrow{\beta} s \xrightarrow{y} f$$

- En conséquence, tous les mots de la forme $x\beta^* y$ sont aussi accepté par l'automate et donc appartiennent au langage.

[1] En effet, soit n un entier naturel. Il existe

$$\sum_{i=0}^{n} card(A)^i = \frac{card(A)^{n+1} - 1}{card(A) - 1}$$

mots de longueur inférieure ou égale à n. Donc on ne peut pas construire de langage infini si l'on borne la taille des mots.

5.3.2 Énoncé

Le théorème suivant, parfois considéré comme un lemme, est appelé **lemme de l'étoile**, théorème du gonflement, lemme d'itération, lemme du facteur itérant ou encore lemme du pompage (*pumping theorem*).

Théorème 3.4 : Soient L un langage rationnel infini et un AFD comportant n états reconnaissant L. $\forall \omega \in L$ de longueur $|\,\omega\,| \geq n$:

$$\exists\, x,\ \beta,\ y \text{ avec } \beta \neq \varepsilon \text{ et } |\,x\beta\,| \leq n \text{ tels que } \omega = x\beta y \text{ et } \forall n \in \mathbb{N},\ x\beta^n y \in L$$

Remarque 3.5 : Pour démontrer ce théorème, il suffit de suivre la suite d'observations de la section précédente.

Remarque 3.6 : Ce théorème ne nous permet pas d'affirmer qu'un langage est rationnel, mais il nous permet d'assurer qu'un langage ne l'est pas !

5.3.3 Exemple et résultat

Exemple 3.7 : On considère le langage $L = \{a^n b^n \mid n \geq 0\}$ dont on sait qu'il est algébrique puisqu'il est engendré par la grammaire algébrique $G = (N = \{X\}, T = \{a, b\}, \rightarrow, X)$ avec $\rightarrow$ définie par :

$$\{\ A \ \rightarrow\ a\,A\,b \ \mid\ \varepsilon\ \}$$

Supposons que L soit rationnel et appelons m le nombre d'états d'un AFD qui reconnaît m. Le mot $\alpha = a^m b^m$ appartient à L et est de longueur supérieure à m. On doit donc pouvoir exprimer α comme la concaténation de trois mots x, β et y avec $\beta \neq \varepsilon$, $|\,x\,\beta\,| \leq m$ tels que $\forall k \in \mathbb{N}$

$$x\,\beta^k\,y \in L$$

Puisque $|\,x\,\beta\,| \leq m$, x et β sont nécessairement composés uniquement de a (puisque les m premières lettres de α sont des a). Le lemme de l'étoile nous assure par exemple que $x\,\beta^2\,y \in L$. Or, en itérant le facteur β, on obtient plus de a que de b, donc le mot n'appartient plus à L. Par conséquent, L ne peut pas être un langage rationnel.

Remarque 3.8 : Nous venons de trouver un langage algébrique qui n'est pas rationnel. On a donc :

$$\textbf{algébrique} \neq \textbf{rationnel}$$

Mais peut-on déterminer une relation plus précise entre l'ensemble des langages rationnels et l'ensemble des langages algébriques ?

5.4 Grammaires linéaires

5.4.1 Définition

Définition 4.1 : Soit $G = (N, T, \rightarrow, X)$ une grammaire algébrique. On dit que G est une **grammaire linéaire** si toutes ses règles sont de la forme :

$$A \rightarrow \alpha\, B \quad \text{ou} \quad A \rightarrow \alpha$$

où A, $B \in N$ et $\alpha \in T^*$.

Remarque 4.2 : Nous allons montrer que tout langage rationnel peut être engendré par une grammaire linéaire puis, réciproquement, que les grammaires linéaires engendre des langages rationnels. Pour ce faire, puisqu'un langage est rationnel ssi il est reconnu par un automate fini (déterministe ou indéterministe), nous allons montrer que l'on peut associer à tout AFD une grammaire linéaire et qu'à toute grammaire linéaire on peut faire correspondre un AF.

5.4.2 Des AFD aux grammaires linéaires

On considère dans toute cette section un langage rationnel L et $\mathcal{A} = (Q, A, \delta, q_0, F)$ un automate fini déterministe reconnaissant L. On rappelle la définition d'un langage associé à un état :

Définition 4.3 : Pour tout état $p \in Q$, on défini le langage associé à p par :

$$L_p = \left\{\alpha \in A^* \mid \exists\, q \in F,\ \delta^*(p, \alpha) = q\right\}$$

i.e l'ensemble des mots qui permettent d'atteindre un état final à partir de p.

Remarque 4.4 : Dans une précédente partie il a été présenté une technique de mise sous forme de système d'équations d'un automate fini déterministe. Pour cela, on a défini, pour tout couple $(p, q) \in Q^2$:

$$E_{p,q} = \left\{\alpha \in A \mid \delta(p, \alpha) = q\right\}$$

qui représente tout simplement l'ensemble des lettres qui étiquettent les transitions allant de p à q. On a alors vu que $\forall p \in Q$:

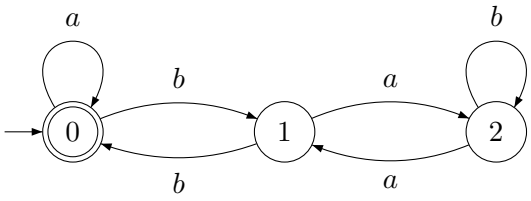
$$\begin{cases} L_p = \displaystyle\sum_{q \in Q} E_{p,q}.L_q & \text{si } p \notin F \\ L_p = \displaystyle\sum_{q \in Q} E_{p,q}.L_q + \varepsilon & \text{si } p \in F \end{cases}$$

Une grammaire linéaire $G = (N, T, \rightarrow, X)$ engendrant L est ainsi définie par :
- $N = Q$ (les non-terminaux de la grammaire sont les états de l'automate)
- $T = A$ (l'ensemble des terminaux de la grammaire est l'alphabet de l'automate)
- $X = q_0$ (l'axiome est l'état initial)
- l'ensemble des règles de production correspond exactement au système d'équation précédemment défini (il suffit de remplacer les symboles « somme » par des symboles « alternative », les deux symboles représentant tous les deux le « ou »)[2] :

$$\left\{ \begin{array}{ll} \textcircled{p} \rightarrow \omega\, \textcircled{q} & \forall (p, \omega, q) \in \delta \\ \textcircled{p} \rightarrow \varepsilon & \forall p \in F \end{array} \right\}$$

[2] Comme δ est une fonction, on note indifférement $(p, \omega, q) \in \delta$ et $\delta(p, \omega) = q$.

Exemple 4.5 : Soit $\mathcal{A} = (Q, A, \delta, q_0, F)$ un automate fini déterministe représenté par le graphe suivant :



Le système d'équation s'écrit simplement :

$$\begin{cases} L_0 = a.L_0 + b.L_1 + \varepsilon \\ L_1 = b.L_0 + a.L_2 \\ L_2 = a.L_1 + b.L_2 \end{cases}$$

Et la grammaire linéaire correspondante est $G = (N, T, \rightarrow, X)$ où :
- $N = \{⓪, ①, ②\}$
- $T = \{a, b\}$
- $X = ⓪$
- $\rightarrow$ définie par

$$\left\{ \begin{array}{lcl} ⓪ & \rightarrow & a\,⓪ \mid b\,① \mid \varepsilon \\ ① & \rightarrow & b\,⓪ \mid a\,② \\ ② & \rightarrow & a\,① \mid b\,② \end{array} \right\}$$

5.4.3 Des grammaires linéaires aux AF

Nous allons désormais montrer qu'il est possible de construire un automate fini à partir d'une grammaire linéaire reconnaissant le même langage. On considère alors une grammaire linéaire $G = (N, T, \rightarrow, X)$ et on définit l'automate fini $\mathcal{A} = (Q, A, E, I, F)$ équivalent par[3] :
- $Q = N \cup \{f\}$ (les états de A sont les non-terminaux de G plus un nouvel état que l'on nomme f)
- $A = T$
- $I = \{X\}$
- $F = \{f\}$ (A ne comportera qu'un seul état final)
- $E = \left\{ \begin{array}{ll} (A, \omega, B) & \text{pour tout } A \rightarrow \omega B \\ (A, \omega, f) & \text{pour tout } A \rightarrow \omega \end{array} \right\}$

5.5 Conclusion et remarques

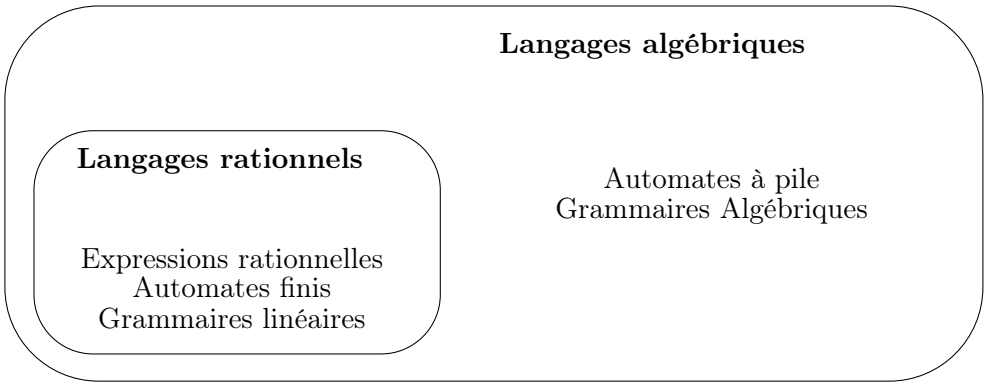
Remarque 5.1 : On a montré dans un premier temps qu'il existe des langages algébriques non réguliers puis que tous les langages réguliers pouvaient être engendrés par des grammaires particulières. On en conlu donc que l'ensemble des langages rationnels est inclus strictement dans l'ensemble des langages algébriques. Autrement dit, si l'on note $Alg(A^*)$ l'ensemble des langages algébriques sur un alphabet A :

$$Rat(A^*) \subsetneq Alg(A^*)$$

[3] On note indifféremment $(A, \omega, B) \in E$ et $A \xrightarrow{\omega} B$

On se pose donc la question suivante :
Question : Puisque l'ensemble des langages rationnels est inclus dans l'ensemble des langages algébriques et que les automates finis définissent exactement les langages rationnels, que manque-t-il à ces derniers pour qu'ils puissent reconnaître tous les langages algébriques ? Peut-on trouver une extension des automates finis pour qu'ils soient équivalents aux grammaires algébriques ?
Réponse : Oui ! Il faut et il suffit de leur ajouter une pile ! Ce type d'automate sera présenté en TD.

Remarque 5.2 : On a donc le schéma suivant :



Remarque 5.3 : A noter qu'il existe des langages plus généraux que les langages algébriques :

$$\text{rationnels} \subsetneq \text{algébriques} \subsetneq \text{contextuels} \subsetneq \text{arbitraires}$$

5.6 Travaux Dirigés

5.6.1 Exercice 1

Dans cet exercice, on notera A un alphabet quelconque, A^* l'ensemble des mots sur A (*i.e.* le plus grand langage sur A au sens de l'inclusion) et $\mathcal{P}(A^*)$ l'ensemble des langages sur A.

Définition 6.1 : Soit ϕ une opération d'arité n. On rappelle qu'un ensemble E est **fermé** (**stable**) par ϕ ssi :

$$L_1,\ L_2,\ \ldots,\ L_n \in E \quad \Rightarrow \quad \phi(L_1, L_2, \ldots, L_n) \in E$$

On sait que, par définition, l'ensemble des langages rationnels $Rat(A^*)$ est stable par concaténation, union et itéré. On cherche alors d'autres propriétés de fermeture de $Rat(A^*)$. Pour cela, on s'appuiera sur le résultat suivant (théorème de Kleene) :

Théorème 6.2 : Un langage est rationnel ssi il est reconnaissable par un automate fini.

On en déduit que l'ensemble $Rat(A^*)$ est stable pour l'opération ϕ ssi :

$$L_1,\ L_2,\ \ldots,\ L_n \text{ sont reconnaissables par des automates finis } \mathcal{A}_1,\ \mathcal{A}_2,\ \ldots,\ \mathcal{A}_n$$
$$\Rightarrow$$
$$\phi(L_1, L_2, \ldots, L_n) \text{ est reconnaissable par un automate fini } \mathcal{A}_\phi$$

Il suffit alors de construire $\mathcal{A}_\phi$ à partir de $\mathcal{A}_1,\ \mathcal{A}_2,\ \ldots,\ \mathcal{A}_n$.

a) Montrer que $Rat(A^*)$ est fermé par complémentaire. En déduire (en utilisant les lois de De Morgan) que $Rat(A^*)$ est fermé par intersection.

b) Montrer que $Rat(A^*)$ est stable par l'opération miroir. On peut montrer de la même façon que $Rat(A^*)$ est fermé par préfixe, suffixe, facteur, quotient gauche et quotient droit.

c) On cherche désormais à déterminer les propriétés de fermeture de l'ensemble des langages algébriques. On s'appuiera sur la propriété suivante :

Théorème 6.3 : Un langage est algébrique ssi il est engendré par une grammaire algébrique.

Pour montrer que $Alg(A^*)$ est stable pour une opération ϕ, il suffit alors de considérer n grammaires algébriques $G_1,\ G_2,\ \ldots,\ G_n$ engendrant respectivement $L_1,\ L_2,\ \ldots,\ L_n$ et de construire une grammaire algébrique engendrant $\phi(L_1, L_2, \ldots, L_n)$ à partir de $G_1,\ G_2,\ \ldots,\ G_n$.
Montrer que $Alg(A^*)$ est fermé par union, concaténation et itération.
($Alg(A^*)$ n'est pas fermé par intersection ni par complémentaire. En revanche, $Alg(A^*)$ est fermé par intersection avec un langage rationnel.)

5.6.2 Exercice 2

On note $nbOcc : A^* \times A \to \mathbb{N}$ l'application qui à un mot α et une lettre a associe le nombre d'occurences de a dans α.

a) Soit $L_1 = \{\alpha \in A^* \mid nbOcc(\alpha, a) = nbOcc(\alpha, b)\}$. Donner une grammaire algébrique G_1 engendrant L_1 et démontrer par induction structurelle que $\mathcal{L}(G_1) = L_1$.

b) Soit la grammaire linéaire $G_2 = \left(\{A, B\}, \{a, b\}, \left\{ \begin{array}{rcl} A & \to & aA \mid B \\ B & \to & bB \mid \varepsilon \end{array} \right\}, A\right)$. Donner un automate équivalent à (*i.e.* reconnaissant le langage engendré par) G_2.

c) Déterminiser l'automate obtenu à la question précédente.

d) Déterminer une expression rationnelle dénotant $L_2 = \mathcal{L}(G_2)$.

e) Montrer, en utilisant le lemme de l'étoile, que le langage $L_3 = \{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas rationnel.

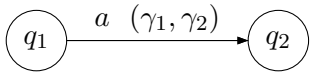
f) Trouver le langage $L = L_1 \cap L_2$. En déduire que L_1 n'est pas rationnel.

5.6.3 Exercice 3

Définition 6.4 : On appelle **automate à pile** tout 6-uplet $\mathcal{A} = (A, Q, q_0, R, F, P)$ où
- A est l'**alphabet d'entrée**
- Q est un ensemble fini, l'ensemble des **états**
- $q_0 \in Q$ est l'**état initial**
- $F \subset Q$ est l'ensemble des **états finaux** (**terminaux**)
- Γ est l'**alphabet de la pile**
- $R \subset Q \times (P \cup \varepsilon) \times (A \cup \varepsilon) \times Q \times (P \cup \varepsilon)$ est une relation, appellée **relation de transition**

Une transition $t \in R$ est donc un quintuplet $(q_1, \gamma_1, a, q_2, \gamma_2)$ où
- $q_1, q_2 \in Q$ sont respectivement l'origine et l'extrêmité de la transition
- a est l'étiquette de la transition
- $\gamma_1 \in P$ est le symbole à dépiler
- $\gamma_2 \in P$ est le symbole à empiler

On peut représenter t de la façon suivante : $(q_1, \gamma_1) \xrightarrow{a} (q_2, \gamma_2)$ ou de façon graphique :



Définition 6.5 : Soit $\mathcal{A} = (A, Q, q_0, R, F, P)$ un automate à pile et $\alpha \in A^*$. On dit que α est reconnu par $\mathcal{A}$ ssi, la pile étant vide avant la lecture de α, il existe une suite non vide de transitions dont l'origine est q_0, d'étiquette α aboutissant soit à un état final, soit à la pile vide.

Construire un automate à pile reconnaissant le langage
- $L_1 = \{\alpha \in A^* \mid nbOcc(\alpha, a) = nbOcc(\alpha, b)\}$
- $L_3 = \{a^n b^n \mid n \in \mathbb{N}\}$.

Chapitre 6

Examens

6.1 Énoncé 2007-2008

École Supérieure d'Informatique et Applications de Lorraine – 1ère année
Rédacteurs : Francis Alexandre, Tony Bourdier, Joseph Rouyer
Date : Mercredi 5 décembre 2007
Durée : 2 heures

Examen – Mathématiques Discrètes 1

Calculatrices, machines électroniques et ordinateurs portables **non autorisés**.
Remarques : La clareté de la rédaction est un élément important de l'évaluation. Le barême est donné à titre indicatif. Cet énoncé est composé de plusieurs parties indépendantes. Il n'est pas demandé de traiter les parties dans l'ordre dans lequel elles sont présentées.

1. Algèbre de Boole */* 6.5 points */*

On rappelle que la fonction appelée « xor » et notée $\oplus$ est définie par :

$$\begin{array}{rccc} \oplus : & \mathbb{B}^2 & \to & \mathbb{B} \\ & (x,y) & \mapsto & x.\overline{y} + \overline{x}.y \end{array}$$

(*a*) *(1 point)* Montrer que pour tout $(x_1, x_2, \ldots, x_n) \in \mathbb{B}^n$:

$$1 \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_n = 1 \quad \Leftrightarrow \quad card\,(\{i \mid x_i = 1\}) \text{ est pair}$$

(*b*) *(0.25 point)* En déduire le support de la fonction $f_1 : (x,y,z) \mapsto 1 \oplus x \oplus y \oplus z \ \in \mathbb{F}_3$ puis celui de la fonction $f_2 : (x,y,z,t) \mapsto 1 \oplus x \oplus y \oplus z \ \in \mathbb{F}_4$.

(*c*) *(0.5 point)* Soit $f_3 \in \mathbb{F}_n$ une fonction autoduale. Si $x \in S_n(f_3)$, que dire de $\overline{x}$? *(le justifier)*

(*d*) *(2 points)* Trouver la fonction f la plus simple répondant à ces conditions :

$$\begin{array}{rlcl} i) & \overline{f}(x,y,z,t) & \Rightarrow & (x+y+z+t)(x+\overline{y}+\overline{z}+t) \\ ii) & x\,(\,y\,\overline{z} + \overline{y}\,z\,\overline{t}\,) & \Rightarrow & f(x,y,z,t) \\ iii) & f(x,y,z,t) & \Rightarrow & \big(\,x \Rightarrow (\,y+z\,)\,\big) \\ iv) & \big(\,\overline{x}\,y\,\overline{z} + z\,\overline{t}\,(\,x \Leftrightarrow y\,)\,\big) & \Rightarrow & \overline{f}(x,y,z,t) \end{array}$$

(*e*) *(0.25 point)* Rappeler la définition d'une partie génératrice.

(*f*) *(0.5 point)* Donner les 5 propriétés stables par composition. Quelle est leur utilité ?

(*g*) *(1.5 points)* Démontrer que $\{f, 0, +\}$ est une partie génératrice (on pourra mettre à profit les réponses aux questions précédentes en observant de près le support de f).

(*h*) *(0.5 point)* $\{f, 0, +\}$ est-elle une partie génératrice minimale ? *(le justifier)*

2. Théorie des langages et des automates */* 14 points */*

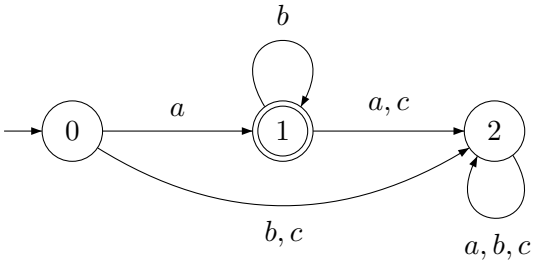
2.1. Composition d'AFD complets */* 9.5 points */*

Définition (Automate fini déterministe complet) On rappelle qu'un automate fini déterministe $\mathcal{A} = (Q, A, \delta, i, T)$ est complet si et seulement la fonction δ est une application (*i.e.* si $\forall (p, a) \in Q \times A,\ \exists q \in Q \mid \delta(p, a) = q$).

(*a*) *(0.5 point)* Construire un automate fini indéterministe sur l'alphabet $A = \{a, b, c\}$ reconnaissant le langage dénoté par l'expression $(ac)^*a$

(*b*) *(1 point)* Transformer l'automate obtenu en automate fini déterministe complet que l'on nommera $\mathcal{E}_1 = (Q_1, A, \delta_1, i_1, T_1)$.

(*c*) *(0.25 point)* Soit l'automate fini $\mathcal{E}_2 = (Q_2 = \{0, 1, 2\}, A = \{a, b, c\}, \delta_2, 0, T_2 = \{1\})$ où δ_2 est donné par le graphe sagittal suivant :



Expliquer en quoi $\mathcal{E}_2$ est déterministe.

(*d*) *(0.5 point)* Donner une expression rationnelle dénotant $\mathcal{L}(\mathcal{E}_2)$ *(il n'est pas nécessairement utile, sur cet exemple, d'utiliser un quelconque algorithme).*

(*e*) *(1.5 points)* On définit désormais la loi de composition interne + sur les automates finis déterministes complets :
Définition (Composition d'automates finis déterministes complets) Soient $\mathcal{A}_1 = (Q_1, A, \delta_1, i_1, T_1)$ et $\mathcal{A}_2 = (Q_2, A, \delta_2, i_2, T_2)$ deux automates finis déterministes complets. L'automate $\mathcal{A} = \mathcal{A}_1 + \mathcal{A}_2$ est défini par :

$$\mathcal{A} = (Q_1 \times Q_2, A, \delta_{1,2}, (i_1, i_2), T_1 \times Q_2 \ \cup \ Q_1 \times T_2)$$

où

$$\begin{array}{rccl} \delta_{1,2} : & (Q_1 \times Q_2) \times A & \to & Q_1 \times Q_2 \\ & ((p, q), a) & \mapsto & (p', q') \quad \text{si } \delta_1(p, a) = p' \text{ et } \delta_2(q, a) = q' \end{array}$$

Construire l'automate $\mathcal{E}_{1+2} = \mathcal{E}_1 + \mathcal{E}_2$ (il est recommandé de construire la table de l'application transition avant d'établir un graphe sagittal).

(*f*) *(1.5 points)* Montrer que $\mathcal{E}_{1+2}$ est minimal.

(*g*) *(1.5 points)* Donner une expression rationnelle dénotant $\mathcal{L}(\mathcal{E}_{1+2})$.

(*h*) *(0.5 point)* Montrer que si $\mathcal{A}_1$ et $\mathcal{A}_2$ sont des automates finis déterministes complets, alors $\mathcal{A}_1 + \mathcal{A}_2$ est un automate fini déterministe complet.

(*i*) *(1.25 point)* Montrer *(par récurrence sur la longueur du mot ou par induction structurelle)* que le prolongement de l'application $\delta_{1,2}$ vérifie : $\forall((p,q),\alpha) \in (Q_1 \times Q_2) \times A^*$,

$$\delta^*_{1,2}((p,q),\alpha) = (p',q') \quad \Leftrightarrow \quad \delta^*_1(p,\alpha) = p' \text{ et } \delta^*_2(q,\alpha) = q'$$

(*j*) *(1 point)* Soient $\mathcal{A}_1$ et $\mathcal{A}_2$ deux automates finis déterministes complets. Montrer que :

$$m \in \mathcal{L}(\mathcal{A}_1 + \mathcal{A}_2) \quad \Leftrightarrow \quad m \in \mathcal{L}(\mathcal{A}_1) \text{ ou } m \in \mathcal{L}(\mathcal{A}_2)$$

En déduire une relation entre les langages reconnus par les automates $\mathcal{A}_1$, $\mathcal{A}_2$ et $\mathcal{A}_1 + \mathcal{A}_2$.

2.2. Grammaires algébriques

/ 4.5 points */*

(*a*) *(0.5 point)* Soit la grammaire $G_1 = (N = \{X\}, T = \{a,b,c\}, \rightarrow, X)$ où la relation $\rightarrow$ est définie par :

$$X \rightarrow aXa \ \mid \ aXb \ \mid \ bXa \ \mid \ bXb \ \mid \ c$$

Montrer que le mot *aabcbbb* est engendré par G_1.

(*b*) *(1.5 points)* Démontrer par induction structurelle que

$$\mathcal{L}(G_1) \subset \{\alpha\, c\, \beta \ \mid \ \alpha,\beta \in \{a,b\}^* \text{ et } |\,\alpha\,| = |\,\beta\,|\}$$

où $|\,m\,|$ désigne la longueur du mot m.

(*c*) *(1 point)* On suppose que $\mathcal{L}(G_1) = \{\alpha\, c\, \beta \ \mid \ \alpha,\beta \in \{a,b\}^* \text{ et } |\,\alpha\,| = |\,\beta\,|\}$, démontrer que $\mathcal{L}(G_1)$ n'est pas rationnel *(vous pourrez utiliser la méthode de votre choix).*

(*d*) *(1.5 points)* Écrire une grammaire algébrique G_2, non ambigüe, qui engendre toutes les expressions arithmétiques sur les variables x, y, z, t en notation infixées respectant les priorités usuelles des opérateurs

$$-\text{ (unaire)} \ > \ *, / \ > \ +, -$$

6.2 Corrigé 2007-2008

École Supérieure d'Informatique et Applications de Lorraine – 1ère année
Rédacteur : Tony Bourdier
Date : Mercredi 5 décembre 2007
Durée : 2 heures

Correction – Mathématiques Discrètes 1

Calculatrices, machines électroniques et ordinateurs portables **non autorisés**.
Remarques : La clareté de la rédaction est un élément important de l'évaluation. Le barême est donné à titre indicatif. Cet énoncé est composé de plusieurs parties indépendantes. Il n'est pas demandé de traiter les parties dans l'ordre dans lequel elles sont présentées.

1. Algèbre de Boole

/ 6.5 points */*

(a) *(6 minutes)* On montre le résultat par récurrence sur n :
- $n = 0$:

$$1 = 1 \quad \Leftrightarrow \quad card(\varnothing) = 0 \text{ pair}$$

Le résultat est donc vrai au rang 0.
- On suppose la propriété vraie au rang n, *i.e.*

$$1 \bigoplus_{i=1}^{n} x_i = 1 \quad \Leftrightarrow \quad card(\{i \leq n \ \mid \ x_i = 1\} \text{ pair}$$

On a :

$$1 \bigoplus_{i=1}^{n} x_i \oplus x_{n+1} = 1 \quad \Leftrightarrow \quad \begin{cases} 1 \bigoplus_{i=1}^{n} x_i = 0 \text{ et } x_{n+1} = 1 & (i) \\ 1 \bigoplus_{i=1}^{n} x_i = 1 \text{ et } x_{n+1} = 0 & (ii) \end{cases}$$

$$\overset{H.R}{\Leftrightarrow} \quad \begin{cases} card(\{i \leq n \ \mid \ x_i = 1\} \text{ impair et } x_{n+1} = 1 & (i) \\ card(\{i \leq n \ \mid \ x_i = 1\} \text{ pair et } x_{n+1} = 0 & (ii) \end{cases}$$

$$\Rightarrow \quad \begin{cases} card(\{i \leq n+1 \ \mid \ x_i = 1\} \text{ pair} & (i) \\ card(\{i \leq n+1 \ \mid \ x_i = 1\} \text{ pair} & (ii) \end{cases}$$

Donc la propriété se transmet du rang n au rang $n+1$.
On a donc, pour tout $n \in \mathbb{N}$:

$$1 \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_n = 1 \quad \Leftrightarrow \quad card\left(\{i \mid x_i = 1\}\right) \text{ est pair}$$

(b) *(1 minute)* On en déduit que :

$$S_3(f_1) = \{(0,0,0), (0,1,1), (1,0,1), (1,1,0)\}$$

et

$$S_4(f_2) = \{(0,0,0,0), (0,0,0,1), (0,1,1,0), (0,1,1,1), (1,0,1,0), (1,0,1,1), (1,1,0,0), (1,1,0,1)\}$$

(*c*) *(1 minute)* Soit $f_3 \in \mathbb{F}_n$ telle que $f_3^* = f_3$.

$$\begin{array}{rcl} x \in S_n(f_3) & \Leftrightarrow & f_3(x) = 1 \\ & \Leftrightarrow & f_3^*(x) = 1 \\ & \Leftrightarrow & \overline{f}_3(\overline{x}) = 1 \\ & \Leftrightarrow & f_3(\overline{x}) = 0 \\ & \Leftrightarrow & \overline{x} \notin S_n(f_3) \end{array}$$

(*d*) *(8 minutes)* On a :

$$\begin{array}{rlcl} i) & \overline{(x+y+z+t)(x+\overline{y}+\overline{z}+t)} = \overline{x}\,\overline{y}\,\overline{z}\,\overline{t} + \overline{x}yz\overline{t} & \Rightarrow & f(x,y,z,t) \\ ii) & x\,(\,y\,\overline{z} + \overline{y}\,z\,\overline{t}\,) = xy\overline{z} + x\overline{y}z\overline{t} & \Rightarrow & f(x,y,z,t) \\ iii) & \big(\,x \Rightarrow (\,y+z\,)\,\big) = \overline{(\,\overline{x}+y+z\,)} = x\overline{y}\,\overline{z} & \Rightarrow & \overline{f}(x,y,z,t) \\ iv) & \big(\,\overline{x}\,y\,\overline{z} + z\,\overline{t}\,(\,x \Leftrightarrow y\,)\,\big) = \overline{x}\,y\,\overline{z} + z\,\overline{t}\,xy + z\,\overline{t}\,\overline{x}\,\overline{y} & \Rightarrow & \overline{f}(x,y,z,t) \end{array}$$

On obtient le tableau de Karnaugh suivant :

	x	x	$\overline{x}$	$\overline{x}$	
y	1	0	1	0	$\overline{t}$
y	1	$*$	$*$	0	t
$\overline{y}$	0	$*$	$*$	$*$	t
$\overline{y}$	0	1	0	1	$\overline{t}$
	$\overline{z}$	z	z	$\overline{z}$	

On note f_{min} la fonction obtenue lorsque $* = 0$ et f_{max} lorsque $* = 1$.
$M(f_{min}/f_{max}) = \{xy\overline{z}, x\overline{y}z, \overline{x}yz, \overline{x}\,\overline{y}\,\overline{z}\}$.
$C(f_{min}/f_{max}) = M(f_{min}/f_{max})$.
Donc $f : (x,y,z,t) \mapsto xy\overline{z} + x\overline{y}z + \overline{x}yz + \overline{x}\,\overline{y}\,\overline{z}$
On s'aperçoit que f est la fonction appelée f_2 à la question (*b*).

(*e*) *(0,5 minute)* $E \subset \mathbb{F}_n$ génératrice $\Leftrightarrow \text{comp}(E) = \mathbb{F}_n$.

(*f*) *(3 minutes)* Soit $f \in \mathbb{F}$:

$$\begin{array}{rcl} P_1(f) & \Leftrightarrow & f(0,\ldots,0) = 0 \\ P_2(f) & \Leftrightarrow & f(1,\ldots,1) = 1 \\ P_3(f) & \Leftrightarrow & f^* = f \\ P_4(f) & \Leftrightarrow & \forall x,y \in \mathbb{B}^n,\ x \geq y \Rightarrow f(x) \geq f(y) \\ P_5(f) & \Leftrightarrow & f \in \mathit{comp}(\{0,1,\oplus\}) \end{array}$$

E génératrice $\Leftrightarrow \forall i \in [1,5],\ \exists f \in E \mid \neg P_i(f)$.

(*g*) *(9 minutes)* On a vu que $f : (x,y,z,t) \mapsto 1 \oplus x \oplus y \oplus z$ donc :

$$\begin{array}{lll} \neg P_1(f) & \text{car} & (0,0,0,0) \in S_4(f) \\ \neg P_2(f) & \text{car} & (1,1,1,1) \notin S_4(f) \\ P_3(f) & \text{car} & \forall x \in \mathbb{B}^4,\ x \in S_4(f) \Leftrightarrow \overline{x} \notin S_4(f) \text{ donc (question } (c)) \ f^* = f \\ \neg P_4(f) & \text{car} & f(0,0,0,0) = 1 > f(1,1,1,1) = 0 \\ P_5(f) & \text{car} & f : (x,y,z,t) \mapsto 1 \oplus x \oplus y \oplus z \in \mathit{comp}(\{0,1,\oplus\}) \end{array}$$

Ainsi :

	f	0	$+$
$\neg P_1$	$*$		
$\neg P_2$	$*$	$*$	
$\neg P_3$		$*$	$*$
$\neg P_4$	$*$		
$\neg P_5$			$*$

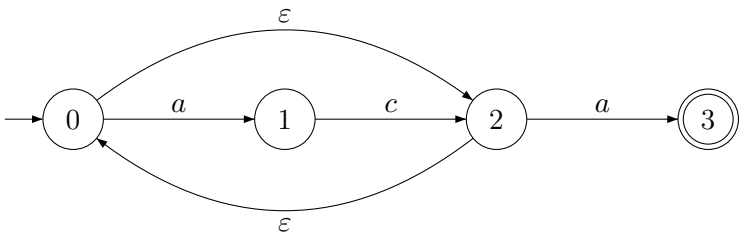
Donc $\{f, 0, +\}$ est génératrice.

(h) *(1 minute)* $\{f, 0, +\}$ n'est pas minimale puisque $\{f, +\}$ est génératrice.

2. Théorie des langages et des automates */* 14 points */*

2.1. Composition d'AFD complets */* 9.5 points */*

(a) *(2 minutes)*



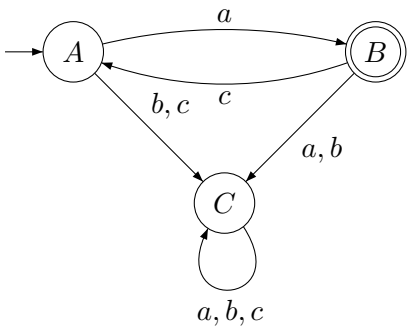
(b) *(4 minutes)* Déterminisation :

	a	b	c
$(initial)\{0,2\}$	$\{1,3\}$	–	–
$(final)\{1,3\}$	–	–	$\{0,2\}$

En renommant :

	a	b	c
$(initial)A$	B	–	–
$(final)B$	–	–	A

Minimalisation : $\approx_0$: $F = \{B\}$ et $Q \setminus F = \{A\}$
$\approx_1 = \approx_0$
Donc l'automate était déjà minimal. On le rend complet en ajoutant un état puit :



(c) *(1 minute)*
- $\mathcal{E}_2$ possède un unique état initial
- $\mathcal{E}_2$ ne possède aucune ε-transition
- $\forall (p, a, q_1, q_2) \in Q_2 \times A \times Q_2 \times Q_2 : \big(\delta_2(p, a) = q_1 \text{ et } \delta_2(p, a) = q_2\big) \Rightarrow (q_1 = q_2)$

(d) *(1 minute)* $\mathcal{L}(\mathcal{E}_2) = ab^*$

(*e*) *(6 minutes)* L'application $\delta_{1,2}$ de $\mathcal{E}_{1,2}$ est donnée par la table suivante

	a	b	c
$(0,A)$	$(1,B)$	$(2,C)$	$(2,C)$
$(1,B)$	$(2,C)$	$(1,C)$	$(2,A)$
$(1,C)$	$(2,C)$	$(1,C)$	$(2,C)$
$(2,A)$	$(2,B)$	$(2,C)$	$(2,C)$
$(2,B)$	$(2,C)$	$(2,C)$	$(2,A)$
$(2,C)$	$(2,C)$	$(2,C)$	$(2,C)$

où l'unique état initial est $(0,A)$ et les états finaux sont $(1,B)$, $(1,C)$ et $(2,B)$.

(*f*) *(6 minutes)*

$$\overset{\approx_0}{\left|\begin{array}{c|c}(0,A) & (1,B)\\ (2,A) & (1,C)\\ (2,C) & (2,B)\end{array}\right|}$$

Puis

$$\overset{\approx_1}{\left|\begin{array}{c|c|c|c|c}(0,A) & (1,B) & (1,C) & (2,C) & (2,B)\\ (2,A) & & & & \end{array}\right|}$$

Enfin

$$\overset{\approx_2}{\left|\ (0,A)\ \right|\ (2,A)\ \left|\ (1,B)\ \right|\ (1,C)\ \left|\ (2,C)\ \right|\ (2,B)\ \left|\right.}$$

Donc $\mathcal{E}_{1,2}$ est minimal.

(*g*) *(2 minutes)* Il ne faut pas inclure $L_{(2,C)}$ dans les équations puisque $(2,C)$ est un état puit et par conséquent : $L_{(2,C)} = \varnothing$! On a donc :

$$\begin{aligned}
L_{(0,A)} &= aL_{(1,B)}\\
L_{(1,B)} &= bL_{(1,C)} + cL_{(2,A)} + \varepsilon\\
L_{(1,C)} &= bL_{(1,C)} + \varepsilon\\
L_{(2,A)} &= aL_{(2,B)}\\
L_{(2,B)} &= cL_{(2,A)} + \varepsilon
\end{aligned}$$

Ce qui donne : $\mathcal{L}(\mathcal{E}_{1,2}) = L_{(0,A)} = ab^* + a(ca)^*$

(*h*) *(8 minutes)* $\delta^*_{1,2}$ est défini par :

$$\begin{aligned}
\delta^*_{1,2}((p,q),\varepsilon) &= (p,q)\\
\delta^*_{1,2}((p,q),a.\alpha) &= \delta^*_{1,2}(\delta_{1,2}((p,q),a),\alpha)
\end{aligned}$$

Raisonnons par récurrence sur la longueur du mot :
– $n = 0 : \forall (p,q) \in Q_1 \times Q_2,\ \delta^*_{1,2}((p,q),\varepsilon) = (p,q) = (\delta_1(p,\varepsilon),\delta_2(q,\varepsilon))$
– Soit α tel que $|\,\alpha\,| = n$. $\forall (p,q,a) \in Q_1 \times Q_2 \times A$:

$$\begin{aligned}
\delta^*_{1,2}((p,q),a.\alpha) &= \delta^*_{1,2}(\delta_{1,2}((p,q),a),\alpha)\\
&= (\text{par définition de } \delta_{1,2})\ \ \delta^*_{1,2}((\delta_1(p,a),\delta_2(q,a)),\alpha)\\
&= (H.R. \text{ car } |\,\alpha\,| = n)\ \ (\delta^*_1(\delta_1(p,a),\alpha),\delta^*_2(\delta_2(q,a),\alpha))\\
&= (\delta^*_1(p,a.\alpha),\delta^*_2(q,a.\alpha))
\end{aligned}$$

(*i*) *(2 minutes)* Soit $m \in A^*$:

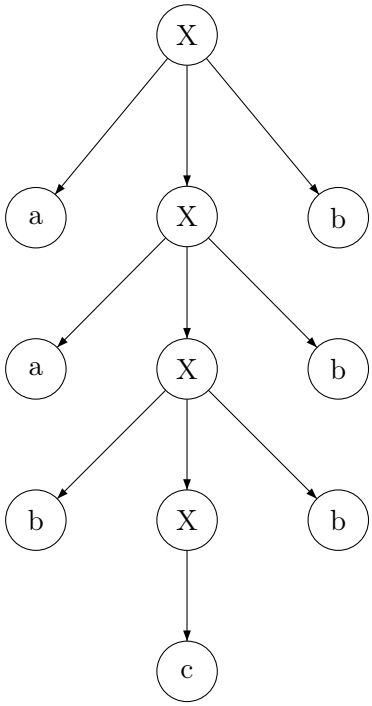
$$\begin{aligned} m \in \mathcal{L}(A_1 + A_2) &\Leftrightarrow \delta^*_{1,2}((i_1, i_2), m) \in T_1 \times Q_2 \cup Q_1 \times T_2 \\ &\Leftrightarrow \delta^*_{1,2}((i_1, i_2), m) \in T_1 \times Q_2 \\ &\quad \text{ou } \delta^*_{1,2}((i_1, i_2), m) \in Q_1 \times T_2 \\ &\Leftrightarrow \delta^*_1(i_1, m) \in T_1 \\ &\quad \text{ou } \delta^*_2(i_2, m) \in T_2 \\ &\Leftrightarrow m \in \mathcal{L}(A_1) \text{ ou } m \in \mathcal{L}(A_2) \end{aligned}$$

Donc $\mathcal{L}(A_1 + A_2) = \mathcal{L}(A_1) \cup \mathcal{L}(A_2)$

2.2. Grammaires algébriques

/ 4.5 points */*

(*a*) *(2 minutes)* Il existe un arbre syntaxique engendrant *aabcbbb* donc $aabcbbb \in \mathcal{L}(G_1)$:



(*b*) *(5 minutes)* Démonstration par induction structurelle :
- Base : $c = \varepsilon c \varepsilon = \alpha c \beta$ avec $\alpha, \beta \in \{a, b\}^*$ et $|\,\alpha\,| = 0 = |\,\beta\,|$
- Induction : Soit $\gamma \in \mathcal{L}(G_1)$. γ s'écrit donc (hypothèse d'induction) sous la forme $\alpha\, c\, \beta$ avec $\alpha, \beta \in \{a, b\}^*$ et $|\,\alpha\,| = |\,\beta\,| = n$.
  $a\,\gamma\,a = (H.I.) a\,\alpha\,c\,\beta\,a = \alpha'\,c\,\beta'$ avec $\alpha', \beta' \in \{a, b\}^*$ et $|\,\alpha'\,| = n + 1 = |\,\beta'\,|$.
  idem pour $a\gamma b,\ b\gamma a,\ b\gamma b$.

Donc tout mot γ de $\mathcal{L}(G_1)$ s'écrit $\alpha\, c\, \beta$ avec $\alpha, \beta \in \{a, b\}^*$ et $|\,\alpha\,| = |\,\beta\,|$.

(*c*) *(4 minutes)* Supposons que $\mathcal{L}(G_1)$ soit rationnel. Il est donc reconnu par un AFD à n états. On sait que $a^n\, c\, b^n \in \mathcal{L}(G_1)$. Le lemme de l'étoile nous assure que :

$$\exists x, y, z \text{ tels que } |\,xy\,| \leq n,\ a^n\, c\, b^n = xyz \text{ et } \forall k \in \mathbb{N},\ xy^k z \in \mathcal{L}(G_1)$$

Comme $|\,xy\,| \leq n$, x et y ne sont composés que de a. Donc $\forall k > 1$, $xy^k z$ s'écrit $a^p\, c\, b^q$ avec $p > q$ donc n'appartient pas à $\mathcal{L}(G_1)$ donc $\mathcal{L}(G_1)$ ne peut pas être rationnel.

(*d*) *(4 minutes)* $G_2 = (N = \{X, Y, Z, V\}, T = \{x, y, z, t, -, *, /, +, (,)\}, \rightarrow, X)$ où $\rightarrow$ est définie par :

$$\begin{array}{lcl} X & \rightarrow & X + Y \mid X - Y \mid Y \\ Y & \rightarrow & Y * Z \mid Y/Z \mid Z \\ Z & \rightarrow & -V \mid V \\ V & \rightarrow & x \mid y \mid z \mid t \mid (X) \end{array}$$