

Université Pierre et Marie Curie
Licence d'Informatique – 2011-2012

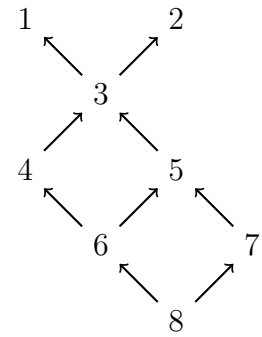
LI214

Structures Discrètes

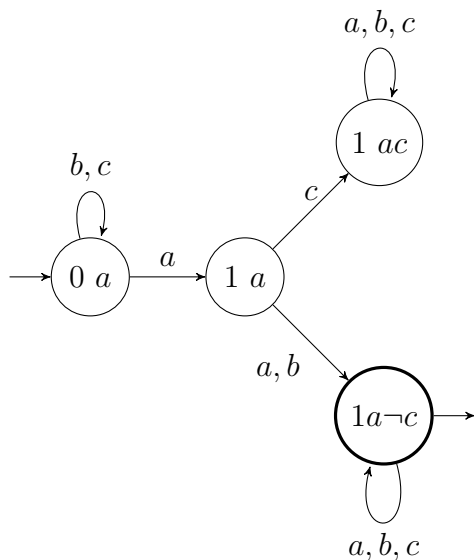
Travaux dirigés, travaux sur machine encadrés, rappels
--

Fascicule enseignants

$$\begin{cases} P(0) \\ P(n) \implies P(n+1) \end{cases} \iff \begin{cases} P(0) \\ \forall k \leq n, P(k) \implies P(n+1) \end{cases}$$



TD LI214



1. $\exists x \forall y R(x, y)$
2. $\forall x \exists y R(x, y)$
3. $\forall x \forall y (R(x, y) \Rightarrow \exists z (R(x, z) \wedge R(z, y)))$
4. $\forall x \exists y (R(x, y) \wedge \forall z (R(x, z) \Rightarrow (z = y \vee R(y, z))))$

Ensembles et relations

Auteurs

IG, transformé en L^AT_EX par VMM, révision MB/BB/MJ

Version du 27 juillet 2012

Exercice 1

1. Donner $S_1 \times S_1$ pour $S_1 = \{0, 1, 2\}$.

Solution :

$$S_1 \times S_1 = \{(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)\}$$

Noter que cet ensemble contient $3 \times 3 = 9$ éléments.

2. Calculer l'ensemble $\mathcal{P}(S)$ des parties de S pour S_1 puis pour $S_2 = \{1, \{1, 4\}\}$.

Solution :

L'ensemble $\mathcal{P}(S)$ des parties de S est l'ensemble de tous les sous-ensembles de S . Il est aussi appelé ensemble puissance de S , noter que $\mathcal{P}(S)$ contient 2^n éléments si S en contient n . Donc :

$$\mathcal{P}(S_1) = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0, 1\}, \{0, 2\}, \{1, 2\}, S_1\}$$

contient $2^3 = 8$ éléments.

Comme S_2 contient deux éléments : 1 et l'ensemble $\{1, 4\}$, l'ensemble $\mathcal{P}(S_2)$ contient $2^2 = 4$ éléments.

$$\mathcal{P}(S_2) = \{S_2, \{1\}, \{\{1, 4\}\}, \emptyset\}$$

3. Démontrer que $A \times (B \cap C) = (A \times B) \cap (A \times C)$.

Solution :

$$A \times (B \cap C) = \{(x, y) \mid x \in A, y \in B \cap C\} = \{(x, y) \mid x \in A, y \in B, y \in C\} = \{(x, y) \mid (x, y) \in A \times B, (x, y) \in A \times C\} = (A \times B) \cap (A \times C).$$

Mais une démonstration c'est un texte ...

Il s'agit de montrer l'égalité de deux ensembles, on montre la double inclusion.

$$- A \times (B \cap C) \subseteq (A \times B) \cap (A \times C)$$

Soit $(x_1, x_2) \in A \times (B \cap C)$. Par définition on a $x_1 \in A$ et $x_2 \in B \cap C$ et donc $x_2 \in B$ et $x_2 \in C$ et on a donc $(x_1, x_2) \in A \times B$ et $(x_1, x_2) \in A \times C$ ce qui permet d'obtenir $(x_1, x_2) \in (A \times B) \cap (A \times C)$.

$$- (A \times B) \cap (A \times C) \subseteq A \times (B \cap C)$$

Soit $(x_1, x_2) \in (A \times B) \cap (A \times C)$. Par définition, on a $(x_1, x_2) \in A \times B$ et $(x_1, x_2) \in A \times C$ et donc il vient $x_1 \in A$, $x_2 \in B$ et $x_2 \in C$ ce qui permet d'obtenir $(x_1, x_2) \in A \times (B \cap C)$.

4. Soit E un sous-ensemble d'un ensemble F et x un élément de F qui n'est pas dans E . Montrer que :

$$\mathcal{P}(E \cup \{x\}) = \mathcal{P}(E) \cup \{\{x\} \cup A \mid A \in \mathcal{P}(E)\}$$

Solution :

Une démonstration c'est un texte ...

On montre la double inclusion.

$$- \mathcal{P}(E \cup \{x\}) \subseteq \mathcal{P}(E) \cup \{\{x\} \cup A \mid A \in \mathcal{P}(E)\}$$

Soit B une partie de $E \cup \{x\}$, on distingue deux cas. Si $x \in B$, alors $B \setminus \{x\}$ est une partie de E , et on a donc $B \in \{\{x\} \cup A \mid A \in \mathcal{P}(E)\}$ ce qui permet d'obtenir $B \in \mathcal{P}(E) \cup \{\{x\} \cup A \mid A \in \mathcal{P}(E)\}$. Sinon, si $x \notin B$, alors B est une partie de E et on obtient directement $B \in \mathcal{P}(E) \cup \{\{x\} \cup A \mid A \in \mathcal{P}(E)\}$.

$$- \mathcal{P}(E) \cup \{\{x\} \cup A \mid A \in \mathcal{P}(E)\} \subseteq \mathcal{P}(E \cup \{x\})$$

Soit $B \in \mathcal{P}(E) \cup \{\{x\} \cup A \mid A \in \mathcal{P}(E)\}$, on a $B \in \mathcal{P}(E)$ ou $B \in \{\{x\} \cup A \mid A \in \mathcal{P}(E)\}$ et donc il vient $B \subseteq E$ ou $B \subseteq \{x\} \cup A$ pour une certaine partie A de E , c'est à dire pour un ensemble $A \subseteq E$. Dans les deux cas on a bien $B \subseteq E \cup \{x\}$ ce qui permet finalement d'obtenir $B \in \mathcal{P}(E \cup \{x\})$.

Remarque. Cette démonstration est l'étape de récurrence qui permet de montrer que pour tout ensemble fini E à n éléments, l'ensemble $\mathcal{P}(E)$ des parties de E a pour cardinal 2^n . En effet, pour $n = 0$, l'ensemble E est vide, donc $\mathcal{P}(E) = \{\emptyset\}$ a un seul élément. Considérons un ensemble ayant $n + 1$ éléments, il est de la forme $E \cup \{x\}$ pour un élément x n'appartenant pas à E . L'hypothèse de récurrence donne $\text{card}(\mathcal{P}(E)) = 2^n$ et on déduit de la démonstration ci-dessus que le cardinal de $\mathcal{P}(E \cup \{x\})$ est égal à $2 \times \text{card}(\mathcal{P}(E))$, ce qui permet de conclure.

Exercice 2

Soient A, B, C trois parties de E .

1. Montrer que $A \cap \overline{A \cap B} = A \cap \overline{B}$.

Solution :

$$\text{On a } A \cap \overline{A \cap B} = A \cap (\overline{A} \cup \overline{B}) = (A \cap \overline{A}) \cup (A \cap \overline{B}) = \emptyset \cup (A \cap \overline{B}) = A \cap \overline{B}$$

2. Montrer que si $A \cap B = A \cap C$ alors $A \cap \overline{B} = A \cap \overline{C}$.

Solution :

Si $A \cap B = A \cap C$ alors $\overline{A \cap B} = \overline{A \cap C}$ donc $A \cap \overline{A \cap B} = A \cap \overline{A \cap C}$ et $A \cap \overline{B} = A \cap \overline{C}$, d'après la question 1.

3. En déduire que

$$A \cap \overline{B} = A \cap \overline{C} \text{ si et seulement si } A \cap B = A \cap C$$

Solution :

Supposons que $A \cap \overline{B} = A \cap \overline{C}$ alors $A \cap \overline{\overline{B}} = A \cap \overline{\overline{C}}$, d'après la question 2. Puisque $\overline{\overline{X}} = X$ pour toute partie X de E , on a donc $A \cap B = A \cap C$.

Exercice 3

Soient A, B, C trois parties de E . Montrer que si $(A \cup B \subseteq A \cup C \text{ et } A \cap B \subseteq A \cap C)$ alors $B \subseteq C$. Dans quel cas a-t-on l'égalité $B = C$?

Solution :

Supposons $A \cup B \subseteq A \cup C$ et $A \cap B \subseteq A \cap C$. On a $B \subseteq A \cup B \subseteq A \cup C$, donc

$$B \subseteq B \cap (A \cup C) = (B \cap A) \cup (B \cap C) \subseteq (A \cap C) \cup (B \cap C) = (A \cup B) \cap C \subseteq C$$

Montrons que $B = C \iff (A \cup B = A \cup C \text{ et } A \cap B = A \cap C)$. L'implication \implies est immédiate. Réciproquement, en appliquant deux fois le résultat précédent on obtient $B \subseteq C$ et $C \subseteq B$, donc $B = C$.

Exercice 4

On considère la relation (ternaire) S définie sur $D = \{0, 1, 2, 3, 4\}$ par $S(a, b, c)$ si $c = a + b$. Décrire S sous la forme d'un sous-ensemble de D^3 .

Solution :

$$S = \{(0, 0, 0), (0, 1, 1), (1, 0, 1), (2, 0, 2), (0, 2, 2), (1, 1, 2), (3, 0, 3), (0, 3, 3), (1, 2, 3), (2, 1, 3), (4, 0, 4), (0, 4, 4), (1, 3, 4), (3, 1, 4), (2, 2, 4)\}$$

Exercice 5

On considère la relation (binaire) $R = \{(1, 1), (2, 3), (3, 2)\}$ sur $X = \{1, 2, 3\}$. Déterminer si R est

1. réflexive

Solution :

R n'est pas réflexive parce que $(2, 2) \notin R$.

2. symétrique

Solution :

R est symétrique parce que $R^{-1} = R$.

3. transitive

Solution :

R n'est pas transitive parce que $(3, 2) \in R$ et $(2, 3) \in R$ mais $(3, 3) \notin R$.

Exercice 6

Soient $A = \{1, 2, 3, 4\}$ et $B = \{1, 3, 5\}$, et soit R le sous-ensemble de $A \times B$ défini par $(a, b) \in R$ ssi $a < b$.

1. Écrire la relation R comme un ensemble de paires ordonnées.

Solution :

R est constituée par l'ensemble des paires $(a, b) \in A \times B$ telles que $a < b$; on a alors :

$$R = \{(1, 3), (1, 5), (2, 3), (2, 5), (3, 5), (4, 5)\}$$

2. Représenter R sur un diagramme $A \times B$.

Solution :

$A \setminus B$	1	3	5
1		x	x
2		x	x
3			x
4			x

3. Déterminer les relations $R^{-1}.R$ et $R.R^{-1}$.

Solution :

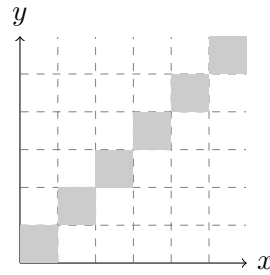
$$R^{-1}.R = \{(3, 3), (3, 5), (5, 3), (5, 5)\} \text{ et } R.R^{-1} = A \times A.$$

Exercice 7

Soit T la relation sur l'ensemble des réels \mathbb{R} définie comme suit : xTy s'il existe $n \in \mathbb{N}$ tel que $x \in [n, n+1]$ et $y \in [n, n+1]$.

Tracer le graphe de la relation T .

Solution :



Graphe de T

Exercice 8

La relation R sur \mathbb{N} définie par " nRm si $m = n + 1$ " est-elle symétrique ? réflexive ? transitive ? Quelles sont les relations R^+ et R^* ?

Solution :

Si R était symétrique, on aurait $m = n + 1$ si et seulement si $n = m + 1$, ce qui est impossible. Si elle était réflexive on aurait $n = n + 1$. Si elle était transitive, on aurait

$$m = n + 1 \text{ et } p = m + 1 \text{ implique } p = n + 1,$$

ce qui n'est pas le cas. On montre par récurrence que $nR^i m$ si et seulement si $m = n + i$ donc R^+ est la relation d'ordre strict sur les entiers ($<$) et R^* la relation d'ordre large (voir le chapitre sur les relations d'ordre).

Exercice 9

La relation R sur \mathbb{N} définie par " nRm si n et m ont un diviseur commun différent de 1" est-elle transitive ?

Solution :

Cette relation n'est pas transitive. Par exemple, 2 et 6 ont un diviseur commun (2), 6 et 3 ont un diviseur commun (3), mais le diviseur commun à 2 et 3 est 1.

Exercice 10

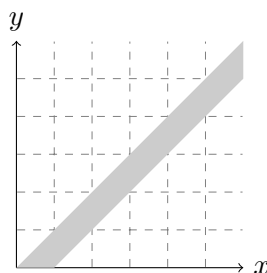
Soit T la relation sur l'ensemble des réels \mathbb{R} définie par

$$xTy \text{ ssi } 0 \leq x - y \leq 1$$

1. Exprimer T et T^{-1} comme sous-ensembles de $\mathbb{R} \times \mathbb{R}$ et tracer le graphe des relations.

Solution :

$$T = \{(x, y) : x, y \in \mathbb{R}, 0 \leq x - y \leq 1\} \text{ et } T^{-1} = \{(x, y) : (y, x) \in T\} = \{(x, y) : x, y \in \mathbb{R}, 0 \leq y - x \leq 1\}.$$



Graphe de T

Noter que le graphe de T^{-1} est le symétrique de celui de T par rapport à la première diagonale.

2. Montrer que $T^{-1}.T = \{(x, z) : |x - z| \leq 1\}$.

Solution :

Par définition de la composition,

$$\begin{aligned} T^{-1}.T &= \{(x, z) : \exists y \in \mathbb{R} \text{ t.q. } (x, y) \in T^{-1}, (y, z) \in T\} \\ &= \{(x, z) : \exists y \in \mathbb{R} \text{ t.q. } (y, x), (y, z) \in T\} \\ &= \{(x, z) : \exists y \in \mathbb{R} \text{ t.q. } 0 \leq y - x \leq 1, 0 \leq y - z \leq 1\} \end{aligned}$$

Supposons $S = \{(x, z) : |x - z| \leq 1\}$. Nous voulons montrer que $T^{-1}.T = S$. Soit $(x, z) \in T^{-1}.T$. Alors il existe y t.q. $0 \leq y - x, y - z \leq 1$. Mais

$$\begin{aligned} 0 \leq y - x, y - z \leq 1 &\Rightarrow y - z \leq 1 \\ &\Rightarrow y - z \leq 1 + y - x \\ &\Rightarrow x - z \leq 1. \end{aligned}$$

Mais aussi

$$\begin{aligned} 0 \leq y - x, y - z \leq 1 &\Rightarrow y - x \leq 1 \\ &\Rightarrow y - x \leq 1 + y - z \\ &\Rightarrow -1 \leq x - z. \end{aligned}$$

Cela signifie que $0 \leq y - x, y - z \leq 1 \Rightarrow -1 \leq x - z \leq 1$ soit $|x - z| \leq 1$. En conséquence, $(x, z) \in S$, c'est-à-dire $T^{-1}.T \subseteq S$.

Supposons que $(x, z) \in S$, alors $|x - z| \leq 1$. Soit $y = \max(x, z)$, alors $0 \leq y - x \leq 1$ et $0 \leq y - z \leq 1$. Alors $(x, z) \in T^{-1}.T$, c'est-à-dire que $S \subseteq T^{-1}.T$. En conclusion $T^{-1}.T = S$.

Exercice 11

Soit $E = \{e_1, \dots, e_n\}$ un ensemble fini et soit R une relation binaire sur E . On représente R par une matrice M_R de dimension $n \times n$, à éléments dans $\{0, 1\}$ de la façon suivante :

$$m_{i,j} = \begin{cases} 1 & \text{si } e_i R e_j, \\ 0 & \text{sinon.} \end{cases}$$

1. Quelle est la propriété de M_R caractérisant le fait que la relation R est symétrique ? réflexive ? irréflexive ? antisymétrique ?

Solution :

La relation R est symétrique si la matrice M_R est symétrique, c'est-à-dire si pour tout couple (i, j) , $m_{i,j} = m_{j,i}$. Elle est réflexive s'il n'y a que des 1 sur la diagonale, et elle est irréflexive s'il n'y a que des 0. Elle est antisymétrique si pour tout couple (i, j) , si $m_{i,j} = 1$ et $m_{j,i} = 1$ alors $i = j$.

2. Si on connaît M_R et $M_{R'}$, comment peut-on calculer $M_{R^{-1}}$, $M_{\bar{R}}$ et $M_{R.R'}$.

Solution :

$M' = M_{R^{-1}}$ est la matrice transposée de M : $m'_{i,j} = m_{j,i}$.

$M' = M_{\bar{R}}$ est la matrice définie par $m'_{i,j} = 1 - m_{i,j}$.

Si $M = M_R$ et $M' = M_{R'}$, alors $M'' = M_{R.R'}$ est la matrice définie par $m''_{i,j} = 1$ si et seulement si $\exists k : m_{i,k} m'_{k,j} = 1$.

Exercice 12

Démontrer que pour deux relations quelconques $R \subseteq X \times Y$ et $S \subseteq Y \times Z$, on a : $(R.S)^{-1} = S^{-1}.R^{-1}$.

Solution :

$$\begin{aligned} (R.S)^{-1} &= \{(z, x) : (x, z) \in R.S\} \\ &= \{(z, x) : \exists y \in Y \text{ t.q. } (x, y) \in R \text{ et } (y, z) \in S\} \\ &= \{(z, x) : \exists y \in Y \text{ t.q. } (y, x) \in R^{-1} \text{ et } (z, y) \in S^{-1}\} \\ &= S^{-1}.R^{-1} \end{aligned}$$

Exercice 13

Soit A un ensemble. On appelle *relation diagonale* la relation $\Delta_A = \{(a, a) : a \in A\}$. Soit $R \subseteq A \times A$ une relation. Démontrer que :

1. R est réflexive ssi $\Delta_A \subseteq R$.

Solution :

Noter que la relation diagonale $\Delta_A = \{(a, a) : a \in A\}$ correspond à l'identité sur A . Donc, par définition, R est réflexive ssi pour chaque $a \in A$, $(a, a) \in R$ ssi $\Delta_A \subseteq R$.

2. R est symétrique ssi $R = R^{-1}$

Solution :

Conséquence directe de la définition de R^{-1} et de la symétrie : supposons d'abord R symétrique, alors pour tout couple (x, y) de $A \times A$, $R^{-1}(x, y)$ ssi $R(y, x)$ (par définition de R^{-1}), et $R(y, x)$ ssi $R(x, y)$ (propriété de symétrie de R appliquée 2 fois) donc pour tout couple (x, y) de $A \times A$, $R^{-1}(x, y)$ ssi $R(x, y)$, i.e. $R = R^{-1}$.

Réciproquement, supposons $R = R^{-1}$. Alors, pour tout couple (x, y) , $R(x, y)$ ssi $R^{-1}(x, y)$, soit $R(x, y)$ ssi $R(y, x)$, donc R symétrique.

3. R est transitive ssi $R.R \subseteq R$

Solution :

Soit $(a, c) \in R.R$. Alors il existe $b \in A$ t.q. $(a, b) \in R$ et $(b, c) \in R$. Par transitivité, $(a, b) \in R$ et $(b, c) \in R$ implique $(a, c) \in R$. On conclut que $R.R \subseteq R$.

Supposons $R.R \subseteq R$. Si $(a, b) \in R$ et $(b, c) \in R$, alors $(a, c) \in R.R \subseteq R$, c'est-à-dire R est transitive.

4. R réflexive implique $R \subseteq R.R$ et $R.R$ réflexive

Solution :

Soit $(a, b) \in R$. Par définition $R.R = \{(a, c) \mid \text{il existe } b \in A \text{ t.q. } (a, b) \in R \text{ et } (b, c) \in R\}$. Mais $(a, b) \in R$ et comme R est réflexive, $(b, b) \in R$. Alors $(a, b) \in R.R$, i.e. $R \subseteq R.R$. En particulier, $\Delta_A \subseteq R \subseteq R.R$ implique que $R.R$ est réflexive.

5. R symétrique implique $R^{-1}.R = R.R^{-1}$

Solution :

$R^{-1}.R = \{(a, c) : \exists b \in A \text{ t.q. } (a, b) \in R^{-1} \text{ et } (b, c) \in R\} = \{(a, c) : \exists b \in A \text{ t.q. } (a, b) \in R \text{ et } (b, c) \in R^{-1}\} = R.R^{-1}$

6. R transitive implique $R.R$ transitive.

Solution :

Soit $(a, b), (b, c) \in R.R$. Par 3, $R.R \subseteq R$ et donc $(a, b), (b, c) \in R$. Alors $(a, c) \in R.R$, c'est-à-dire $R.R$ est transitive.

Exercice 14

On considère l'ensemble $\mathbb{N} \times (\mathbb{N} \setminus \{0\})$, c'est-à-dire l'ensemble des couples d'entiers naturels dont le deuxième élément est non nul. Soit R définie sur cet ensemble par $(a, b)R(c, d)$ si $ad = bc$. Démontrer que R est une relation d'équivalence.

Solution :

Noter que $(a, b)R(a, b)$ pour tout $(a, b) \in \mathbb{N} \times (\mathbb{N} \setminus \{0\})$, puisque $ab = ba$; donc la relation R est réflexive.

Supposons que $(a, b)R(c, d)$. Alors $ad = bc$ et en particulier, $cb = da$. Donc, par définition, $(c, d)R(a, b)$ et la relation R est symétrique.

Supposons que $(a, b)R(c, d)$ et que $(c, d)R(e, f)$. Alors $ad = bc$ et $cf = de$. En particulier $(ad)(cf) = (bc)(de)$, et en

simplifiant par cd on obtient $af = be$. Il faut aussi vérifier le cas particulier où $c = 0$, puisque $d \neq 0$. On obtient alors $(a, b)R(e, f)$ d'où R est transitive.

Comme R est réflexive, transitive et symétrique, c'est une relation d'équivalence.

Notons que si la paire ordonnée (a, b) est écrite sous forme de fraction $\frac{a}{b}$, alors la relation R est la définition d'égalité de deux fractions, c'est-à-dire $\frac{a}{b} = \frac{c}{d}$ si et seulement si $ad = bc$.

Exercice 15

Soit R une relation d'équivalence dans A et soit $[a]$ la classe d'équivalence de $a \in A$. Alors :

1. pour chaque $a \in A$, $a \in [a]$

Solution :

Comme R est réflexive, $(a, a) \in R$ pour tout $a \in A$ et donc $a \in [a]$.

2. $[a] = [b]$ ssi $(a, b) \in R$

Solution :

Supposons que $(a, b) \in R$. On veut démontrer que $[a] = [b]$. Soit $x \in [b]$; alors $(b, x) \in R$. Mais par hypothèse, $(a, b) \in R$; donc par transitivité, $(a, x) \in R$. En conséquence, $x \in [a]$, c'est-à-dire $[b] \subset [a]$. Pour prouver que $[a] \subset [b]$, nous observons que $(a, b) \in R$ implique, par symétrie, que $(b, a) \in R$. Alors par un raisonnement similaire, nous obtenons $[a] \subset [b]$. On conclut que $[a] = [b]$.

3. si $[a] \neq [b]$ alors $[a]$ et $[b]$ sont disjointes.

Solution :

Nous prouvons la contraposée, c'est-à-dire si $[a] \cap [b] \neq \emptyset$, alors $[a] = [b]$. Si $[a] \cap [b] \neq \emptyset$, alors il existe un élément $x \in A$ tel que $x \in [a] \cap [b]$. En conséquence $(a, x) \in R$ et $(b, x) \in R$. Par symétrie, $(x, b) \in R$ et, par transitivité, $(a, b) \in R$. On conclut, par 2, que $[a] = [b]$.

4. Montrer que l'ensemble des classes d'équivalence forme une partition de A .

Solution :

Ceci résulte des questions précédentes.

Fonctions et opérations

Auteurs

IG, transformé en L^AT_EX par VMM, révision MB/BB/MP

Version du 27 juillet 2012

Exercice 1

1. Trouver un exemple d'application qui n'est ni injective, ni surjective.

Solution :

$f: \{1, 2\} \longrightarrow \{1, 2\}$, avec $f(1) = f(2) = 1$.

2. Soit $f: A \longrightarrow B$ une application. Montrer que f est injective si et seulement si pour tous sous-ensembles X, Y de A , $f(X \cap Y) = f(X) \cap f(Y)$.

Solution :

Remarquons d'abord que l'inclusion $f(X \cap Y) \subseteq f(X) \cap f(Y)$ est toujours vraie.

Supposons f injective. Soient X et Y des sous-ensembles de A et soit $z \in f(X) \cap f(Y)$. Il existe $x \in X$ tel que $f(x) = z$ et il existe $y \in Y$ tel que $f(y) = z$. Comme f est injective, $f(x) = z = f(y)$ implique $x = y$. Donc $x = y \in X \cap Y$ et $z = f(x) \in f(X \cap Y)$. Ainsi, $f(X) \cap f(Y) \subseteq f(X \cap Y)$ et par suite $f(X) \cap f(Y) = f(X \cap Y)$.

Réciproquement, soient $x, y \in A$ tels que $f(x) = f(y)$. Posons $X = \{x\}$ et $Y = \{y\}$. On a $f(x) = f(y) \in f(X) \cap f(Y) = f(X \cap Y)$. Donc $f(X \cap Y) \neq \emptyset$ et par suite, $X \cap Y \neq \emptyset$. Ceci implique $x = y$.

Exercice 2

Soient A et B deux parties disjointes d'un ensemble E . Réaliser une bijection entre $\mathcal{P}(A) \times \mathcal{P}(B)$ et $\mathcal{P}(A \cup B)$.

Solution :

Soit $f: \mathcal{P}(A) \times \mathcal{P}(B) \rightarrow \mathcal{P}(A \cup B)$ définie par $f(X, Y) = X \cup Y$.

Montrons que f est surjective. Soit $Z \subseteq A \cup B$ alors $Z = (Z \cap A) \cup (Z \cap B)$. En posant $X = Z \cap A$ et $Y = Z \cap B$, on a $(X, Y) \in \mathcal{P}(A) \times \mathcal{P}(B)$ et $f(X, Y) = Z$.

Montrons que f est injective. Supposons que $f(X, Y) = f(X', Y')$ alors $X \cup Y = X' \cup Y'$ donc $(X \cup Y) \cap A = (X' \cup Y') \cap A$, d'où $(X \cap A) \cup (Y \cap A) = (X' \cap A) \cup (Y' \cap A)$. Or $(X \cap A) \cup (Y \cap A) = X$ puisque $X \subseteq A$, $Y \subset B$ et A, B disjointes. De même $(X' \cap A) \cup (Y' \cap A) = X'$ Donc $X = X'$. De manière analogue $Y = Y'$.

Exercice 3

Si $n < m$, il n'existe pas d'injection d'un ensemble à m éléments dans un ensemble à n éléments. Une formulation plus imagée de ce résultat est le *principe des tiroirs* (ou encore *pigeon-hole principle*), à savoir : si $n < m$ et que l'on veut placer m chemises dans n tiroirs alors un tiroir au moins contiendra plusieurs chemises. Plus précisément, montrer qu'un tiroir contiendra au moins un nombre entier $p \geq \frac{m}{n}$ chemises.

Solution :

Soit $A = \{1, 2, \dots, m\}$ l'ensemble des chemises à répartir et soit A_i l'ensemble des chemises contenues dans le tiroir i , pour $i \in [n]$. A_1, \dots, A_n forment une partition de A , donc

$$|A| = \sum_{i=1}^n |A_i|.$$

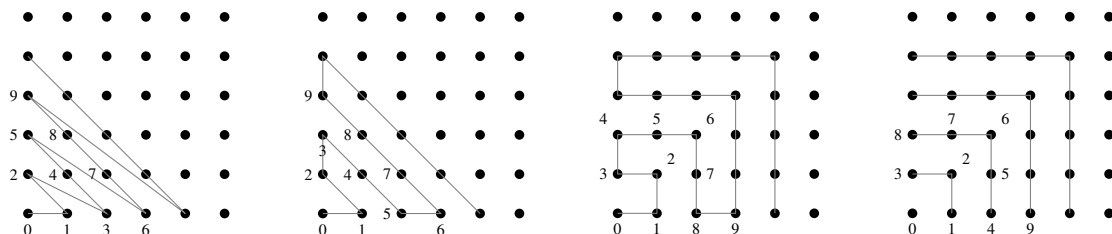
Si l'on avait : $\forall i \in [n], |A_i| < \frac{m}{n}$, on pourrait en déduire $|A| = \sum_{i=1}^n |A_i| < n \times \frac{m}{n} = m$, une contradiction ; donc on a $|A_i| \geq \frac{m}{n}$ pour un i au moins.

Exercice 4

On dit qu'un ensemble E est *dénombrable* s'il existe une bijection entre cet ensemble et \mathbb{N} . Montrer que $\mathbb{N}^2 = \mathbb{N} \times \mathbb{N}$ est dénombrable. On pourra montrer que les fonctions f et g suivantes sont des bijections de \mathbb{N}^2 dans \mathbb{N} : $f(x, y) = y + (0 + 1 + 2 + \dots + (x + y))$ et $g(x, y) = 2^y(2x + 1) - 1$.

Solution :

Une première façon de mettre en bijection $\mathbb{N} \times \mathbb{N}$ avec \mathbb{N} consiste à suivre un chemin "continu" passant par tous les points de $\mathbb{N} \times \mathbb{N}$ en suivant les diagonales (bijection f) ou en suivant une sorte de spirale, etc. Le premier dessin représente le chemin correspondant à la bijection f , les trois autres représentent d'autres chemins possibles.



Une autre façon consiste à utiliser la "moitié" de \mathbb{N} pour numéroté $\mathbb{N} \times \{0\}$, puis la "moitié" de ce qui reste pour numéroté $\mathbb{N} \times \{1\}$, puis la "moitié" de ce qui reste pour numéroté $\mathbb{N} \times \{2\}$, etc (bijection g).

$\mathbb{N} = \{0, 1, 2, 3, \dots\}$, on compte de 2 en 2 pour numéroté $\mathbb{N} \times \{0\}$.

Il reste :

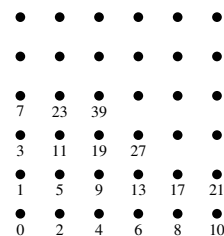
$\mathbb{N}_1 = \{1, 3, 5, 7, \dots\}$, on compte de 2 en 2 pour numéroté $\mathbb{N} \times \{1\}$.

Il reste :

$\mathbb{N}_2 = \{3, 7, 11, 15, \dots\}$, on compte de 2 en 2 pour numéroté $\mathbb{N} \times \{2\}$.

Il reste :

$\mathbb{N}_3 = \{7, 15, 23, 31, \dots\}$, etc



Pour g , soit $n \in \mathbb{N}$. Alors, il existe un unique couple (x, y) tel que $g(x, y) = n$, c'est-à-dire $n + 1 = 2^y(2x + 1)$. En effet, si $n + 1$ est impair on a $y = 0$ et il existe un unique x tel que $n + 1 = 2x + 1$. Si $n + 1$ est pair alors soit y le plus grand entier tel que 2^y divise $n + 1$. On a donc $n + 1 = 2^y h$ avec h impair et le résultat découle du cas précédent.

Pour f , on considère d'abord la suite définie par $u_p = 1 + \dots + p$ pour $p \geq 1$, et $u_0 = 0$. On montre facilement que cette suite est strictement croissante. Ainsi, pour tout entier n , il existe un unique entier p tel que $u_p \leq n < u_{p+1}$. Le couple (x, y) antécédent de n par f est alors défini par : $y = n - u_p$ et $x = p - y$.

Exercice 5

Montrer que l'ensemble des nombres rationnels est dénombrable.

Solution :

Soit \mathbb{Q}^+ l'ensemble des nombres rationnels positifs et soit \mathbb{Q}^- l'ensemble des nombres rationnels négatifs. Alors $\mathbb{Q} = \mathbb{Q}^+ \cup \{0\} \cup \mathbb{Q}^-$ est l'ensemble des nombres rationnels. Soit $f : \mathbb{Q}^+ \rightarrow \mathbb{N} \times \mathbb{N}$ défini par $f(p/q) = (p, q)$, où p/q est un nombre rationnel positif exprimé comme rapport de deux entiers positifs. Notons que f est injective ; donc \mathbb{Q}^+ est équivalent à un sous-ensemble de $\mathbb{N} \times \mathbb{N}$. Mais $\mathbb{N} \times \mathbb{N}$ est un ensemble dénombrable donc \mathbb{Q}^+ est aussi dénombrable. Dans la même manière on démontre que \mathbb{Q}^- est dénombrable. Donc l'union $\mathbb{Q}^+ \cup \{0\} \cup \mathbb{Q}^-$ est dénombrable.

Exercice 6

Soit F un ensemble et E un sous-ensemble de $\mathcal{P}(F)$. On définit sur E la relation \sim par : $A \sim B$ s'il existe une bijection $f : A \rightarrow B$. Démontrer que la relation \sim est une relation d'équivalence. En se restreignant à un sous-ensemble de $\mathcal{P}_f(F)$, l'ensemble des parties finies de F , la relation $|A| = |B|$ est une relation d'équivalence.

Solution :

La fonction identité $1_A : A \rightarrow A$ est injective et surjective. Donc $A \sim A$.

Si $A \sim B$ alors il existe une fonction $f : A \rightarrow B$ surjective et injective. Mais alors la fonction inverse de f , $f^{-1} : B \rightarrow A$, est aussi injective et surjective. Donc $A \sim B$ implique $B \sim A$.

Si $A \sim B$ et $B \sim C$ alors il existe une fonction $f : A \rightarrow B$ et une fonction $g : B \rightarrow C$ qui sont injectives et surjectives. La fonction composée $g \circ f : A \rightarrow C$ est aussi injective et surjective. En conséquence $A \sim B$ et $B \sim C$ implique $A \sim C$.

La relation de cardinalité est un cas particulier de la relation \sim sur les parties finies, donc $|A| = |B|$ est une relation d'équivalence.

Exercice 7

Soit $f : E \rightarrow F$ une application. Montrer que :

1. Si $E \neq \emptyset$ alors f est injective si et seulement si f a un inverse à gauche, c'est-à-dire il existe une application $r : F \rightarrow E$ telle que $r \circ f = id_E$. L'application r est surjective et s'appelle une rétraction de f .

Solution :

Supposons d'abord f injective. On choisit un élément x_0 arbitraire dans E (c'est possible car $E \neq \emptyset$). Soit $y \in F$, si y admet un antécédent x par f , il est unique puisque f est injective ; on pose alors $r(y) = x$. Sinon, on pose $r(y) = x_0$. Pour tout x de E , x est l'unique antécédent de $f(x)$, on a donc : $\forall x \in E, \quad r \circ f(x) = r(f(x)) = x$.

Réciproquement, montrons que f est injective. Soient x_1 et x_2 tels que $f(x_1) = f(x_2)$. Alors, en appliquant r qui est surjective, on obtient $r \circ f(x_1) = r \circ f(x_2)$, et comme $r \circ f = id_E$, on conclut que $x_1 = x_2$.

2. f est surjective si et seulement si f a un inverse à droite, c'est-à-dire il existe une application $s : F \rightarrow E$ telle que $f \circ s = id_F$. L'application s est injective et s'appelle une section de f .

Solution :

Supposons d'abord f surjective. Alors, pour $y \in F$, il existe des x dans E tels que $y = f(x)$. Choisissons un élément arbitraire parmi ces x et posons $s(y) = x$. $\forall y \in F$ on a $f(s(y)) = y$, donc $f \circ s = id_F$.

Réciproquement, montrons que f est surjective. Soit $y \in F$, alors $f \circ s(y) = y$. Posons $x = s(y)$, on obtient $y = f(x)$.

3. f est bijective si et seulement si f a un inverse, c'est-à-dire il existe une application $f^{-1} : F \rightarrow E$ telle que $f \circ f^{-1} = id_F$ et $f^{-1} \circ f = id_E$. L'application f^{-1} est une bijection et s'appelle la bijection réciproque de f .

Solution :

Supposons f bijective. Il suffit de montrer que l'application s définie ci-dessus vérifie aussi $s \circ f = id_E$. Soit $x \in E$ et $y = f(x)$, puisque f est injective x est l'unique antécédent de y par f . On a donc $s(y) = x$ et par suite, $s(f(x)) = x$.

La réciproque se déduit des deux questions précédentes.

Exercice 8

Soit $*$ une opération binaire sur un ensemble E . Démontrer que si $*$ admet un élément neutre, cet élément neutre est unique.

Solution :

Soient 1 et $1'$ deux éléments neutres, on a $1' = 1' * 1 = 1$.

Exercice 9

Montrer que $(\mathbb{N}, +, 0)$ et $(\mathcal{P}(E), \cup, \emptyset)$ sont des monoïdes commutatifs.

Exercice 10

Soit A un alphabet. L'ensemble des mots de A^* de longueur paire est-il un monoïde pour la concaténation ? Même question pour l'ensemble des mots de A^* de longueur impaire.

Solution :

Longueur paire : stable pour la concaténation puisque la somme de deux nombres pairs est paire et ε l'élément neutre de la concaténation en fait partie, donc sous-monoïde de A^* .

Longueur impaire : instable la concaténation puisque la somme de deux nombres impairs est paire et ε l'élément neutre de la concaténation n'en fait pas partie, donc pas un sous-monoïde de A^* .

Induction sur \mathbb{N}

Auteurs

IG, transformé en L^AT_EX par VMM, révision MB/BB

Version du 27 juillet 2012

Exercice 1

Montrer que, pour $n \geq 2$,

$$\overline{A_1 \cup A_2 \cup \dots \cup A_n} = \overline{A_1} \cap \overline{A_2} \cap \dots \cap \overline{A_n}$$

$$\overline{A_1 \cap A_2 \cap \dots \cap A_n} = \overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_n}$$

(Lois de de Morgan généralisées)

Solution :

Base : Pour $n = 2$, lois de de Morgan.

Induction : supposons $\overline{A_1 \cup A_2 \cup \dots \cup A_{n-1}} = \overline{A_1} \cap \overline{A_2} \cap \dots \cap \overline{A_{n-1}}$, et soit $A = A_1 \cup A_2 \cup \dots \cup A_{n-1}$; $\overline{A_1 \cup A_2 \cup \dots \cup A_n} = \overline{A \cup A_n} = \overline{A} \cap \overline{A_n} = \overline{A_1 \cap A_2 \cap \dots \cap A_{n-1}} \cap \overline{A_n}$. (cas de base puis hypothèse induction)

Exercice 2

Soit a un entier. Soit $P_a(n)$ la propriété “ $9 \mid (10^n + a)$ ”. Montrer que, pour tout entier a et tout $n \in \mathbb{N}$, $P_a(n)$ implique $P_a(n+1)$. A-t-on $P_a(n)$ pour tout $n \in \mathbb{N}$?

Solution :

$(10^{n+1} + a) - (10^n + a) = 10 \cdot 10^n - 10^n = 9 \cdot 10^n$ donc $P_a(n)$ implique $P_a(n+1)$.

Pour que la propriété soit vraie pour tout n , il faut qu'elle soit vraie en 0, soit $9 \mid a + 1$ et $a \equiv -1 \pmod{9}$. C'est le cas pour $a \equiv -1$ par exemple mais pas pour $a \equiv +1$.

On voit ici combien le cas de base est important.

Exercice 3

La suite harmonique est définie par $H_n = 1 + 1/2 + \dots + 1/n$ pour $n \geq 1$.

1. Montrer que $H_{2^n} \geq 1 + n/2$.

Solution :

Base : Pour $n = 0$, $H_1 = 1 = 1 + 0/2$.

Induction : montrons $P(n) \implies P(n+1)$ pour tout $n \geq 0$, où $P(n) : H_{2^n} \geq 1 + n/2$. On a

$$H_{2^{n+1}} = H_{2^n} + \underbrace{1/(2^n + 1) + \dots + 1/2^{n+1}}_{2^n \text{ termes } \geq 1/2^{n+1}} \geq 1 + n/2 + 1/2 = 1 + (n+1)/2$$

2. Montrer par induction que $H_n = p_n/q_n$ avec p_n entier positif impair et q_n entier positif pair pour $n \geq 2$. En déduire que H_n n'est pas entier pour $n \geq 2$.

Solution :

Il faut ici une induction généralisée pour le cas (ii) de l'étape d'induction.

Base : $H_2 = 3/2$.

Induction :

$$H_{n+1} = \frac{p_n}{q_n} + \frac{1}{n+1} = \frac{(n+1)p_n + q_n}{(n+1)q_n}.$$

(i) si $n + 1$ impair, $(n + 1)p_n + q_n$ impair et $(n + 1)q_n$ pair.

(ii) si $n + 1$ pair, soit $n + 1 = 2k$, alors :

$$\begin{aligned} H_{n+1} &= 1 + 1/2 + \cdots + 1/(n + 1) = 1 + 1/3 + \cdots + 1/(2k - 1) + 1/2(1 + 1/2 + \cdots + 1/k) \\ &= p_k/2q_k + a/b = \frac{bp_k + 2aq_k}{2q_kb} \text{ avec } b = 1 \times 3 \times \cdots \times (2k - 1) \text{ impair} \end{aligned}$$

et on obtient bien que H_{n+1} est le quotient d'un terme impair par un terme pair dans les deux cas, ce qui achève notre raisonnement.

Exercice 4

On rappelle que les nombres de Fibonacci sont définis par $F_n = F_{n-1} + F_{n-2}$ pour $n > 1$ et $F_0 = 0$, $F_1 = 1$.

Solution :

Remarquons que le terme général d'une suite définie par une récurrence à deux termes s'obtient en résolvant l'équation du second degré associée. Ici $r^2 - r - 1 = 0$ a deux racines : $r_1 = \frac{1+\sqrt{5}}{2}$ et $r_2 = \frac{1-\sqrt{5}}{2}$, donc la suite est de la forme $F_n = \alpha r_1^n + \beta r_2^n$ pour tout n , où α et β sont obtenus en résolvant le système correspondant aux deux premiers termes.

Montrer que pour tout $n > 0$:

1. $F_1 + F_3 + \cdots + F_{2n-1} = F_{2n}$.

Solution :

$$F_2 = F_1 \text{ et } F_{2n+2} = F_{2n} + F_{2n+1}.$$

2. $F_{n+1}F_{n-1} - F_n^2 = (-1)^n$.

Solution :

$$\text{Récurrence et remarquer que } F_{n+2}F_n - F_{n+1}^2 = (F_{n+1} + F_n)F_n - F_{n+1}^2 = F_{n+1}(F_n - F_{n+1}) + F_n^2 = -F_{n+1}F_{n-1} + F_n^2.$$

3. $F_1^2 + F_2^2 + \cdots + F_n^2 = F_n F_{n+1}$

Solution :

$$F_1 F_2 = F_1^2 \text{ et } F_{n+1} F_{n+2} = F_{n+1}(F_n + F_{n+1}) = F_{n+1}^2 + F_{n+1} F_n.$$

4. F_{3n} est pair.

Solution :

$$F_3 = 2, \text{ et } F_{3n+3} = F_{3n+2} + F_{3n+1} = F_{3n} + 2F_{3n+1}.$$

5. $\varphi^{n-2} \leq F_n \leq \varphi^{n-1}$, où $\varphi = \frac{1+\sqrt{5}}{2}$ est la solution positive de $r^2 - r - 1 = 0$.

Solution :

On doit supposer l'inégalité vraie pour deux termes consécutifs, donc il s'agit d'une induction généralisée. $\frac{2}{1+\sqrt{5}} \leq F_1 \leq 1$, et $1 \leq F_2 \leq \frac{1+\sqrt{5}}{2}$, puis $F_{n+1} = F_{n-1} + F_n \leq \varphi^{n-2} + \varphi^{n-1} = \varphi^{n-2}(1 + \varphi) = \varphi^n$ car $\varphi^2 = 1 + \varphi$. Pour la minoration de F_n : $F_{n+1} = F_{n-1} + F_n \geq \varphi^{n-3} + \varphi^{n-2} = \varphi^{n-3}(1 + \varphi) = \varphi^{n-1}$.

Exercice 5

Donner une définition inductive de $f(n) = a^{2^n}$.

Indication : On pourra remarquer que $a^{2^{n+1}} = (a^{2^n})^2$.

Solution :

Immédiat après avoir remarqué que $a^{2^{n+1}} = (a^{2^n})^2$.

$$\text{On a : } f(n) = \begin{cases} a & \text{si } n = 0 \\ [f(n-1)]^2 & \text{si } n > 0 \end{cases}$$

Exercice 6

On considère le polynôme à coefficients réels $P(x) = \frac{1}{3}x^3 + ax^2 + bx$.

1. Trouver a et b pour que $\forall x \in \mathbb{R}, P(x+1) - P(x) = x^2$. On suppose dans la suite que cette propriété est vérifiée.

Solution :

On a $P(x+1) - P(x) = x^2 + (2a+1)x + (a+b+1/3)$. La propriété est donc vérifiée si et seulement si $2a+1 = 0$ et $a+b+1/3 = 0$, c'est-à-dire $a = -1/2$ et $b = 1/6$.

2. Montrer que pour tout $n \in \mathbb{N}$, $P(n)$ est un entier.

Solution :

On le montre par récurrence sur n . Soit $Q(n)$ la propriété " $P(n)$ est un entier". $P(0) = 0$ donc $Q(0)$ est vraie. Soit $n \geq 0$ et supposons $Q(n)$ vraie. $P(n+1) = P(n) + n^2$ donc $Q(n+1)$ est vraie. On en déduit que pour tout $n \geq 0$, $Q(n)$ est vraie.

3. Pour tout $n \geq 0$, on pose $S_n = \sum_{k=0}^n k^2$. Montrer que

$$\forall n \geq 0, S_n = P(n+1) = \frac{n(n+1)(2n+1)}{6}.$$

Solution :

Montrons par récurrence la propriété $R(n) : S_n = P(n+1)$.

$S_0 = 0 = P(1)$ donc $R(0)$ est vraie.

Soit $n \geq 0$, supposons $R(n)$ vraie. On a $S_{n+1} = S_n + (n+1)^2 = P(n+1) + (n+1)^2 = P(n+2)$, donc $R(n+1)$ est vraie. On en déduit que pour tout $n \geq 0$, on a $R(n)$.

Enfin,

$$\frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n = P(n) + n^2 = P(n+1).$$

Exercice 7

1. Montrer que pour tout $n \in \mathbb{N}$, $(n+1)^2 - (n+2)^2 - (n+3)^2 + (n+4)^2 = 4$.

Solution :

On développe l'expression (astucieusement :-)) et on obtient le résultat.

2. En déduire que tout entier m peut s'écrire comme somme et différence des carrés $1^2, 2^2, \dots, n^2$ pour un certain n , c'est-à-dire que pour tout m :

$P(m)$: il existe $n \in \mathbb{N}$, et $\varepsilon_1, \dots, \varepsilon_n \in \{-1, 1\}$, tels que $m = \varepsilon_1 1^2 + \varepsilon_2 2^2 + \dots + \varepsilon_n n^2$.

Indication : montrer d'abord le résultat pour $m \in \{0, 1, 2, 3\}$.

Solution :

On voit (tout de suite...) que $0 = 1^2 + 2^2 - 3^2 + 4^2 - 5^2 - 6^2 + 7^2$, que $1 = 1^2$, que $2 = -1^2 - 2^2 - 3^2 + 4^2$ et que $3 = -1^2 + 2^2$. Puis $P(m)$ implique $P(m+4)$ car $4 = (n+1)^2 - (n+2)^2 - (n+3)^2 + (n+4)^2$.

Exercice 8

1. On suppose qu'une propriété P définie sur \mathbb{N} vérifie :

- (i) $P(1)$ est vraie
- (ii) si $P(n)$ est vraie alors $P(2n)$ est vraie (pour $n \geq 1$)
- (iii) si $P(n)$ est vraie alors $P(n-1)$ est vraie (pour $n \geq 2$).

Montrer par récurrence sur n que $P(n)$ est vraie pour tout $n \geq 1$.

Solution :

Base : $P(1)$ vraie (par (i))

Induction : supposons $P(n)$ vraie pour un entier $n \geq 1$, alors par (ii) $P(2n)$ vraie et par (iii) $P(2n-1), \dots, P(n+1)$ vraies, car $2n, 2n-1, \dots, n+1$ sont supérieurs ou égaux à 2. Donc $P(n)$ implique $P(n+1)$ pour $n \geq 1$ et on obtient le résultat.

2. On veut montrer que la moyenne arithmétique est supérieure à la moyenne géométrique.

Soient a_1, \dots, a_n n nombres réels positifs, avec $n \geq 1$; on pose $A = (a_1 + \dots + a_n)/n$ et

$G = (a_1 \dots a_n)^{1/n}$, montrer que $A \geq G$.

Solution :

Par induction sur n on montre $P(n)$ vraie pour tout n , où :

$$P(n) : \text{pour tout } n\text{-uplet } a_1, \dots, a_n > 0, A \geq G$$

Base : ($n = 1$, et $A = G$ inutile pour le raisonnement) $n = 2$ et $(a_1 + a_2)/2 \geq \sqrt{a_1 a_2}$ en effet $(\sqrt{a_1} - \sqrt{a_2})^2 = a_1 + a_2 - 2\sqrt{a_1 a_2} \geq 0$.

Induction $P(n) \implies P(n-1)$ et $P(n) \implies P(2n)$ pour tout $n \geq 2$, d'où le résultat.

– $P(n) \implies P(2n)$:

$$\begin{aligned} (a_1 + \dots + a_n + a_{n+1} + \dots + a_{2n})/2n &= 1/2 ((a_1 + \dots + a_n)/n + (a_{n+1} + \dots + a_{2n})/n) \\ &\geq 1/2 \left((a_1 \dots a_n)^{1/n} + (a_{n+1} \dots a_{2n})^{1/n} \right) \\ &\quad \text{par hypothèse de récurrence appliquée deux fois} \\ &\geq (a_1 \dots a_n a_{n+1} \dots a_{2n})^{1/2n} \text{ par } P(2) \end{aligned}$$

– $P(n) \implies P(n-1)$: pour montrer $P(n-1)$ pour a_1, \dots, a_{n-1} donnés, on choisit d'ajouter $a_n = (a_1 + \dots + a_{n-1})/(n-1)$ et on va utiliser $P(n)$ sur le n -uplet ainsi formé.

On a

$$A_{n-1} = \frac{(a_1 + \dots + a_{n-1})}{(n-1)} = a_n$$

et

$$\begin{aligned} A_n &= \frac{(a_1 + \dots + a_{n-1}) + \frac{(a_1 + \dots + a_{n-1})}{(n-1)}}{n} \\ &= \frac{(n-1)(a_1 + \dots + a_{n-1}) + (a_1 + \dots + a_{n-1})}{n(n-1)} \\ &= \frac{(a_1 + \dots + a_{n-1})}{(n-1)} = a_n \end{aligned}$$

d'où $A = (a_1 + \dots + a_n)/n = (a_1 + \dots + a_{n-1})/(n-1) = a_n$. Par $P(n)$, $A \geq (a_1 \dots a_n)^{1/n} = (a_1 \dots a_{n-1})^{1/n} A^{1/n}$, d'où $A^{1-1/n} \geq (a_1 \dots a_{n-1})^{1/n}$ et en élevant à la puissance $n/(n-1)$, on obtient $A = (A^{1-1/n})^{n/(n-1)} \geq (a_1 \dots a_{n-1})^{(1/n) \times (n/(n-1))} = (a_1 \dots a_{n-1})^{1/(n-1)}$, ce qui correspond à $P(n-1)$.

Relations d'ordre

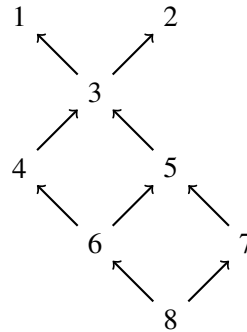
Auteurs

IG, transformé en \LaTeX par VMM, révision BB

Version du 27 juillet 2012

Exercice 1

Soit $W = \{1, 2, \dots, 7, 8\}$ ordonné selon le diagramme suivant :



On considère le sous-ensemble $V = \{4, 5, 6\}$ de W .

1. Trouver l'ensemble des majorants de V .

Solution :

Chaque élément dans $\{1, 2, 3\}$, et seulement ces éléments, est plus grand que tous les éléments dans V : 1, 2, 3 sont les majorants de V .

2. Trouver l'ensemble des minorants de V .

Solution :

Seulement les éléments 6 et 8 précèdent tous les éléments de V . Donc $\{6, 8\}$ est l'ensemble des minorants.

3. Est-ce que $\sup(V)$ existe ?

Solution :

Comme 3 est le plus petit élément dans l'ensemble des majorants de V , alors on a la borne supérieure $\sup(V) = 3$. Notons que $3 \notin V$.

4. Est-ce que $\inf(V)$ existe ?

Solution :

Comme 6 est le plus grand élément dans l'ensemble des minorants de V , alors on a la borne inférieure $\inf(V) = 6$. Notons que $6 \in V$.

Exercice 2

Soit $E = \{2, 3, 4, \dots\} = \mathbb{N} \setminus \{0, 1\}$, ordonné par la relation " x divise y ".

1. Vérifier que cette relation est un ordre.
2. Déterminer les éléments minimaux de E

Solution :

Si $p \in E$ est un nombre premier, alors seulement p divise p (noter que $1 \notin E$). En conséquence tous les nombres premiers sont des éléments minimaux. De plus, si $a \in E$ n'est pas un nombre premier, alors il existe $b \in E$ t.q. b divise a , c'est-à-dire que b est inférieur ou égal à a pour l'ordre $|$; en conséquence a n'est pas minimal. On conclut que les éléments minimaux sont précisément les nombres premiers.

3. Déterminer les éléments maximaux de E .

Solution :

Il n'y a pas d'élément maximal parce que pour chaque $a \in E$, a divise $2a$ par exemple.

Exercice 3

On se place dans $F = \mathbb{N} \setminus \{0\}$ ordonné par la relation " x divise y ".

1. Existe-t-il une borne sup et une borne inf pour tout sous-ensemble de 2 éléments ?

Solution :

Soit un sous-ensemble $A = \{a, b\}$, la borne sup de A est le ppcm de a et b et la borne inf est leur pgcd.

2. Soient les ensembles $A = \{6, 15, 21\}$ et $B = \{1, 6, 14, 21\}$. Donner les minorants et majorants de A (resp. B). A (resp. B) possède-t-il un minimum ? un maximum ?

Solution :

A Majorants : 210 et ses multiples, minorants : 1 et 3. Pas de maximum (ni de minimum) car aucun majorant (ni minorant) n'appartient à l'ensemble.

B Majorants : 42 et ses multiples, minorants : 1. Minimum 1, pas de maximum.

3. Soit $A = \{3, 6, 12, 15\}$. Donner les majorants, minorants, la borne supérieure, la borne inférieure, les éléments maximaux, minimaux. Discuter.

Solution :

Majorant : 60 et ses multiples,

Minorants : 1, 3

Borne sup : 60

Borne inf : 3

Maximum : aucun

Minimum : 3

Maximaux : 12, 15

Minimaux : 3.

Exercice 4

Donner un exemple d'ensemble ordonné qui a exactement un élément maximal mais qui n'a pas de maximum.

Solution :

Considérer l'ensemble $\{a, 1, 2, 3, 4, \dots\}$ t.q. $1 < a$ et $i < i + 1$ pour chaque $i \in \mathbb{N}$. L'élément a est maximal mais il n'est pas le maximum.

Exercice 5

Soit $A = \{a, b, c\}$ ordonné comme l'indique le diagramme suivant :

$$b \longrightarrow a \longleftarrow c$$

Soit \mathcal{A} l'ensemble de tous les sous-ensembles non-vides et totalement ordonnés de A ; \mathcal{A} est partiellement ordonné par inclusion. Représenter graphiquement l'ordre de \mathcal{A} .

Solution :

Les sous-ensembles totalement ordonnés de A sont $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{a, c\}$. Comme \mathcal{A} est ordonné par inclusion, l'ordre de \mathcal{A} est le suivant :

$$\{b\} \longrightarrow \{a, b\} \longleftarrow \{a\} \longrightarrow \{a, c\} \longleftarrow \{c\}$$

Exercice 6

Montrer que les ensembles ordonnés $\mathcal{P}(\{a, b, c\})$ muni de la relation d'inclusion et $\{1, 2, 3, 6, 7, 14, 21, 42\}$ muni de la relation de division (dans \mathbb{N}) sont isomorphes (au sens d'isomorphisme d'ensembles ordonnés).

Solution :

- Ensemble ordonné $\mathcal{P}\{a, b, c\}$ muni de la relation d'inclusion :

Éléments : \emptyset , $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{a, c\}$, $\{b, c\}$, $\{a, b, c\}$.

Paires ordonnées (un niveau de profondeur) : $\emptyset \subset \{a\}$, $\emptyset \subset \{b\}$, $\emptyset \subset \{c\}$, $\{a\} \subset \{a, b\}$, $\{a\} \subset \{a, c\}$, $\{b\} \subset \{a, b\}$, $\{b\} \subset \{b, c\}$, $\{c\} \subset \{a, c\}$, $\{c\} \subset \{b, c\}$, $\{a, b\} \subset \{a, b, c\}$, $\{a, c\} \subset \{a, b, c\}$, $\{b, c\} \subset \{a, b, c\}$.

- Ensemble ordonné $\{1, 2, 3, 6, 7, 14, 21, 42\}$ muni de la relation de division (dans \mathbb{N})

Éléments 1, 2, 3, 6, 7, 14, 21, 42.

Paires ordonnées (un niveau de profondeur) :

- 1 divise 2, 1 divise 3, 1 divise 7
- 2 divise 6, 2 divise 14, 3 divise 6, 3 divise 21, 7 divise 14, 7 divise 21
- 6 divise 42, 14 divise 42, 21 divise 42

Deux ensembles ordonnés sont isomorphes s'il existe une bijection f entre les deux et si f et f^{-1} sont monotones (cf. cours).

Bijection : $\emptyset \longrightarrow 1$, $\{a\} \longrightarrow 2$, $\{b\} \longrightarrow 3$, $\{c\} \longrightarrow 7$, $\{a, b\} \longrightarrow 6$, $\{a, c\} \longrightarrow 14$, $\{b, c\} \longrightarrow 21$, $\{a, b, c\} \longrightarrow 42$.

Elle est monotone (cf. les paires ordonnées de chaque ensemble et transitivité pour les autres paires).

Exercice 7

On considère un ensemble E muni d'une opération binaire notée \sqcup telle que \sqcup est commutative, associative et idempotente (c'est-à-dire pour tout $x \in E$, $x \sqcup x = x$). On définit la relation \preceq sur E par : $x \preceq y$ si $x \sqcup y = y$.

1. Montrer que \preceq est une relation d'ordre.

Solution :

La réflexivité est due à l'idempotence, l'antisymétrie résulte de la commutativité et la transitivité est une conséquence de l'associativité.

Remarquer que si A est un ensemble, l'ensemble E des parties de A muni de l'union satisfait ces hypothèses, la relation d'ordre étant l'inclusion. De même, l'ensemble $E = \{0, 1\}$ muni de l'opération "ou" satisfait ces hypothèses, la relation d'ordre est la relation naturelle sur $\{0, 1\}$ avec $0 \preceq 1$.

2. Montrer que toute paire d'éléments admet une borne supérieure.

Solution :

Soit $D = \{x, y\}$ un ensemble à deux éléments. Alors $z = x \sqcup y$ est bien un majorant de D car $x \sqcup z = z$ et $y \sqcup z = z$, par commutativité et associativité. De plus, tout majorant z' de D (qui satisfait donc $x \sqcup z' = z'$ et $y \sqcup z' = z'$) vérifie : $z' = z \sqcup z'$ donc $z \preceq z'$ et z est le plus petit majorant.

Exercice 8

On dit qu'un ensemble ordonné (E, \preceq) est un *treillis* si tout sous-ensemble fini d'éléments de E admet une borne supérieure et une borne inférieure.

Montrer qu'un ensemble ordonné est un treillis si et seulement si tout sous-ensemble à deux éléments de E admet une borne supérieure et une borne inférieure.

Indication : le sens de n vers 2 est immédiat. Pour la réciproque, utiliser une récurrence sur n .

Exercice 9

1. Définir la relation "est un préfixe de" sur A^* . S'agit-il d'une relation d'ordre ? si oui, s'agit-il d'un ordre total ou d'un ordre partiel ?

Solution :

L'ordre préfixe sur A^* est un ordre partiel défini par :

$$u_1 u_2 \dots u_n \preceq_{pref} v_1 v_2 \dots v_m \quad \text{si} \quad n \leq m \quad \text{et} \quad \forall i \leq n \quad u_i = v_i$$

2. En supposant que A est muni d'un ordre total \preceq_A , définir l'ordre lexicographique sur A^* . S'agit-il d'un ordre total ou d'un ordre partiel ?

Solution :

L'ordre lexicographique \preceq_{lex} sur A^* est un ordre total défini comme suit. Soit $u = u_1 u_2 \dots u_n$ et $v = v_1 v_2 \dots v_m$ deux éléments de A^* . $u \preceq_{lex} v$ si :

- soit u est un préfixe de v : $u \preceq_{pref} v$
- soit u et v coïncident jusqu'à la position k (pour tout $i \leq k$, $u_i = v_i$) et $u_{k+1} \neq v_{k+1}$ et $u_{k+1} \preceq_A v_{k+1}$

3. Soit $A = \{a, b\}$ tel que $a \preceq_A b$. Montrer que l'ordre lexicographique défini à la question précédente n'est pas un ordre bien fondé.

Solution :

Il suffit d'exhiber une suite infinie strictement décroissante d'éléments de A^* pour \preceq_{lex} ... par exemple $(a^n b)_{n \in \mathbb{N}}$.

Exercice 10

Soient (A, \leq_1) et (B, \leq_2) deux ordres bien fondés. Les ordres suivants sont-ils bien fondés ?

1. Sur $A \times B$, l'ordre produit $(a, b) \leq (a', b')$ si et seulement si $a \leq_1 a'$ et $b \leq_2 b'$.

Solution :

Pour prouver qu'un ordre est bien fondé, il faut montrer qu'il n'existe pas de suite infinie strictement décroissante. On peut raisonner par l'absurde (comme ici) ou utiliser la caractérisation du cours : toute partie non vide admet (au moins) un élément minimal (question suivante).

Soit (a_n, b_n) une suite décroissante de $A \times B$. Si (a_n, b_n) est une suite décroissante stricte, a_n et b_n sont toutes deux décroissantes mais pourraient être décroissantes au sens large. Montrons qu'on peut extraire de a_n et b_n des sous-suites infinies a_i et b_j décroissantes strictement. Remarquons que "on ne peut pas extraire de a_n une sous-suite décroissante stricte" équivaut à " a_n est ultimement stationnaire" et faisons un raisonnement par cas.

- s'il existe a_i et b_j , sous-suites de a_n et b_n qui sont décroissantes strictes : rien à faire, c'est prouvé. SINON
- si a_n et b_n sont ultimement stationnaires toutes deux : impossible car (a_n, b_n) est une suite décroissante stricte.
- si a_n décroissante strict et b_n ultimement stationnaire : comme A est bien fondé, a_n ne peut pas être infinie, donc elle est ultimement stationnaire, et on est ramené au cas 2.
- si b_n décroissante stricte et a_n ultimement stationnaire : *mutatis mutandis* sur le 3. en échangeant A et B .

Maintenant, a_i et b_j étant décroissantes strictes dans des ordres bien fondés, elles sont finies (c'est-à-dire ultimement stationnaires toutes les deux) et donc il en est de même de (a_n, b_n) .

2. Sur $A \times B$, l'ordre lexicographique.

Solution :

Soit X une partie non vide de $A \times B$, montrons qu'elle a un élément minimal. Soit $X_A = \{a | \exists b(a, b) \in X\}$: comme A est bien fondé, X_A a un élément minimal m_A . Soit $X_{m_A} = \{b | (m_A, b) \in X\}$: comme B est bien fondé X_{m_A} a un élément minimal m_B . On vérifie que (m_A, m_B) est minimal dans X . Soit $(a, b) \in X$, $(a, b) \leq (m_A, m_B)$, comme $a \in X_A$ et m_A minimal dans X_A , $a = m_A$: donc $b \in X_B$ et comme m_B minimal dans X_B , $b = m_B$.

Exercice 11

Les ordres suivants sont-ils bien fondés ?

1. sur A^2 , l'ordre lexicographique (A alphabet totalement ordonné).
2. sur \mathbb{N} $m \leq n$ ssi m divise n .
3. sur l'ensemble des diviseurs d'un entier donné, la relation du 2.
4. sur A^* , l'ordre préfixe.
5. sur A^* , l'ordre $u \leq v$ ssi u est un sous-mot de v .
6. sur A^* , l'ordre lexicographique (A alphabet totalement ordonné).
7. sur A^*/\equiv , l'ordre des longueurs $u \leq v$ ssi $|u| \leq |v|$ (où \equiv est défini par $u \equiv v$ si et seulement si $|u| = |v|$).

Solution :

tous b.f. sauf l'ordre lexicographique sur A^* . Remarque : pour le 2. on peut prendre aussi sur \mathbb{N} , l'ordre est tout aussi b.f. on a cette fois un ordre qui a un élément maximum 0 (puisque tous les entiers divisent 0).

Exercice 12 – Fonction d'Ackerman

Soit $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ définie par :

$$\begin{cases} f(0, n) = n + 1, \\ f(m, 0) = f(m - 1, 1), \text{ si } m \geq 1 \\ f(m, n) = f(m - 1, f(m, n - 1)), \text{ si } m, n \geq 1 \end{cases}$$

1. Calculer $f(1, n)$ et $f(2, n)$.

Solution :

$f(1, n) = f(0, f(1, n - 1)) = 1 + f(1, n - 1)$, donc (par rec.) $f(1, n) = n + 2$ car on s'arrête à $f(1, 0) = f(0, 1) = 2$.
 $f(2, n) = f(1, f(2, n - 1)) = 2 + f(2, n - 1)$, donc $f(2, n) = 2n + 3$ car $f(2, 0) = f(1, 1) = 3$.

2. Montrer que $f(m, n)$ est définie pour tout couple $(m, n) \in \mathbb{N} \times \mathbb{N}$.

Solution :

Par induction généralisée sur $\mathbb{N} \times \mathbb{N}$ muni de son ordre lexicographique.

3. Soit $g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ définie par :

$$\begin{cases} g(0, n) = n + 1, \\ g(m, 0) = g(m - 1, 1), \text{ si } m \geq 1 \\ g(m, n) = g(m - 1, g(m, n + 1)), \text{ si } m, n \geq 1 \end{cases}$$

Pour quels couples $(m, n) \in \mathbb{N} \times \mathbb{N}$ la fonction $g(m, n)$ est-elle définie ?

Solution :

Pour tous les couples $(0, n)$, $n \in \mathbb{N}$ et pour $(1, 0)$.

Induction structurelle

Auteurs

IG, transformé en L^AT_EX par VMM, révision MB/BB

Version du 27 juillet 2012

Exercice 1

On considère le sous-ensemble D de $\mathbb{N} \times \mathbb{N}$ défini inductivement par :

(B) $(n, 0) \in D$

(I) si $(n, n') \in D$, alors $(n, n + n') \in D$

1. Donner quelques éléments de D .

Solution :

$(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), \dots (2, 0), (2, 2), (2, 4), \dots$

2. Montrer que pour deux entiers n et n' , $(n, n') \in D$ si et seulement s'il existe $k \in \mathbb{N}$, tel que $n' = kn$.

Solution :

- Montrons d'abord par récurrence sur k que si $n' = kn$ alors $(n, n') \in D$
 - Si $k = 0$, alors $n' = 0$ et on peut conclure d'après (B).
 - Considérons $n' = (k + 1)n$. Alors $n' = kn + n$. Or par hypothèse de récurrence, on a $(n, kn) \in D$ et on peut alors conclure que $(n, n') \in D$ par (I).
- Réciproquement, montrons par induction sur la structure de D que si $(n, n') \in D$ alors il existe $k \in \mathbb{N}$ tel que $n' = kn$.
 - (B) Pour $(n, 0)$, il suffit de prendre $k = 0$.
 - (I) Pour $(n, n + n')$, par hypothèse d'induction, il existe $k \in \mathbb{N}$ tel que $n' = kn$ et donc $n + n' = (k + 1)n$, et on peut conclure en choisissant $k' = k + 1$.

Exercice 2

On considère l'ensemble AB des arbres binaires (0, 1 ou 2 fils par nœud) sur un alphabet A .

1. Donner une définition inductive de la hauteur $h(t)$, du nombre de nœuds $n(t)$, du nombre d'arêtes $ar(t)$ et du nombre de feuilles $f(t)$ d'un arbre binaire.

Solution :

Base : pour $t = \emptyset$, on a : $h(\emptyset) = 0$, $n(\emptyset) = 0$, $ar(\emptyset) = 0$ et $f(\emptyset) = 0$.

Pour l'induction, avec $t = (a, g, d)$:

$h((a, g, d)) = \max(h(g), h(d)) + 1$,

$n((a, g, d)) = 1 + n(g) + n(d)$,

$$ar((a, g, d)) = \begin{cases} 0 & \text{si } g = d = \emptyset \\ 1 + ar(g) & \text{si } g \neq \emptyset \text{ et } d = \emptyset \\ 1 + ar(d) & \text{si } d \neq \emptyset \text{ et } g = \emptyset \\ 2 + ar(g) + ar(d) & \text{si } d \neq \emptyset \text{ et } g \neq \emptyset \end{cases}$$

$$f((a, g, d)) = \begin{cases} 1 & \text{si } g = d = \emptyset \\ f(g) + f(d) & \text{si } d \neq \emptyset \text{ ou } g \neq \emptyset \end{cases}$$

2. Montrer que pour tout arbre t de AB , $n(t) \leq 2^{h(t)} - 1$, et que $f(t) \leq 2^{h(t)-1}$.

Solution :

- 1) On prouve par induction structurelle sur AB la propriété : $P(t) : n(t) \leq 2^{h(t)} - 1$.
L'unique élément de la base est l'arbre vide et on a bien $2^{n(\emptyset)} - 1 = 2^0 - 1 = 0 \geq 0 = n(\emptyset)$.
Soit $t = (a, g, d)$ et supposons $P(g)$ et $P(d)$. On a :

$$n(t) = 1 + n(g) + n(d) \leq 2^{h(g)} + 2^{h(d)} - 1 \leq 2 \times 2^{\max(h(g), h(d))} - 1 = 2^{h(t)} - 1.$$

Autre solution : avec une récurrence généralisée sur $n = h(t)$.

- 2) On procède de même pour la propriété $Q(t) : f(t) \leq 2^{h(t)-1}$.
Vu la définition de la fonction f , il faut vérifier Q directement pour l'arbre vide et l'arbre réduit à un seul élément.
- $f(\emptyset) = 0 \leq 2^{-1} = 2^{h(\emptyset)-1}$ et $f((a, \emptyset, \emptyset)) = 1 = 2^0 = 2^{h((a, \emptyset, \emptyset))-1}$.
 - Puis on vérifie l'étape inductive. Soit $t = (a, g, d) \in AB$ avec $g \neq \emptyset$ ou $d \neq \emptyset$ et supposons $Q(g)$ et $Q(d)$.
Alors $f(t) = f(g) + f(d) \leq 2^{h(g)-1} + 2^{h(d)-1} \leq 2 \times 2^{\max(h(g), h(d))-1} = 2^{h(t)-1}$.

3. Définir le parcours préfixe d'un arbre binaire.

Solution :

La définition inductive du parcours préfixe d'un arbre binaire est

$$\text{Pref}(t) = \begin{cases} \varepsilon & \text{si } t = \emptyset, \\ a \cdot \text{Pref}(g) \cdot \text{Pref}(d) & \text{si } t = (a, g, d) \end{cases}$$

Exercice 3

Soit A un alphabet et t un arbre binaire *strict* sur A , c'est-à-dire que t est non vide et chaque nœud de t a exactement 0 ou 2 fils (il n'y a aucun nœud avec un seul fils non vide).

1. Donner une définition inductive de l'ensemble ABS des arbres stricts et adapter les définitions inductives des fonctions $n(t)$ (nombre de nœuds), $f(t)$ (nombre de feuilles) et $ar(t)$ (nombre d'arêtes).

Solution :

La définition inductive de l'ensemble ABS est :

- (B) pour tout $a \in A$, $(a, \emptyset, \emptyset) \in ABS$.
- (I) si $g, d \in ABS$ et $a \in A$, alors $(a, g, d) \in ABS$

2. Montrer que si t est un arbre binaire *strict*, alors $n(t) = ar(t) + 1$.

Solution :

On raisonne par induction sur la structure de ABS .

Pour le cas de base, $t = (a, \emptyset, \emptyset)$, alors $n(t) = 1$ et $ar(t) = 0$.

Pour l'induction, soit $t = (a, g, d)$ où les deux sous-arbres g et d sont dans ABS . On a $n(t) = 1 + n(g) + n(d) = 1 + ar(g) + 1 + ar(d) + 1 = ar(t) + 1$ (faire le dessin).

3. Montrer que si t est un arbre binaire *strict*, alors $n(t) = 2f(t) - 1$.

Solution :

Soit $P(t)$ la propriété $n(t) = 2f(t) - 1$.

- Pour le cas de base, $t = (a, \emptyset, \emptyset)$ alors $n(t) = 1 = 2 \times 1 - 1 = 2f(t) - 1$.
- Pour l'induction, $t = (a, g, d)$ où $g, d \in ABS$ et $a \in A$. Par hypothèse d'induction, $P(g)$ et $P(d)$ sont vraies.
On a $n(t) = 1 + n(g) + n(d) = 1 + 2f(g) - 1 + 2f(d) - 1 = 2(f(g) + f(d)) - 1 = 2f(t) - 1$.

Donc pour tout $t \in ABS$, $P(t)$ est vraie.

Exercice 4

Soit A^* le monoïde libre engendré par l'alphabet A . Donner une définition inductive de A^* .

Le miroir d'un mot $u = a_1a_2 \dots a_n$ est le mot $\tilde{u} = a_n \dots a_2a_1$. Donner une définition inductive du miroir.

Solution :

Le miroir est une application de A^* dans A^* . Pour en donner une définition inductive, on utilise une définition inductive de A^* , c'est-à-dire :

(B) ε appartient à A^*

(I) pour tout m de A^* et toute lettre $a \in A$, $m.a$ appartient à A^*

On obtient alors :

$$\tilde{u} = \begin{cases} \varepsilon & \text{si } u = \varepsilon \\ a.\tilde{v} & \text{si } u = v.a \end{cases}$$

Exercice 5

L'ordre préfixe sur A^* peut être défini de deux manières différentes :

(1) Pour tous mots $m_1, m_2 \in A^*$, $m_1 \preceq_{pref}^1 m_2$ s'il existe $m_3 \in A^*$ tel que $m_1m_3 = m_2$.

(2) Définition inductive :

(B) pour tout mot $m \in A$, $\varepsilon \preceq_{pref}^2 m$

(I) Si $m_1, m_2 \in A$ sont tels que $m_1 \preceq_{pref}^2 m_2$, alors pour toute lettre $a \in A$, $am_1 \preceq_{pref}^2 am_2$

Montrer que ces deux définitions sont équivalentes.

Solution :

On montre l'implication dans les deux sens.

(1) \implies (2) Si $m_1.m_3 = m_2$, la preuve s'obtient par récurrence sur la longueur de m_1 . Si $lg(m_1) = 0$, alors m_1 est le mot vide et on peut conclure. Sinon ($lg(m_1) = n + 1$), alors m_1 s'écrit am'_1 pour un certain $a \in A$ et un certain mot m'_1 de longueur n . Par hypothèse de récurrence, pour tout mot m'_2 , si il existe un mot m'_3 tel que $m'_1.m'_3 = m'_2$ alors on a $m'_1 \preceq_{pref}^2 m'_2$. Or $am'_1.m_3 = m_2$ et donc m_2 s'écrit nécessairement am'_2 . Il existe donc bien un mot $m'_3 = m_3$ tel que $m'_1.m'_3 = m'_2$. On peut alors conclure puisqu'on a $m'_1 \preceq_{pref}^2 m'_2$ et donc

$$\underbrace{am'_1}_{m_1} \preceq_{pref}^2 \underbrace{am'_2}_{m_2}$$

(1) \Leftarrow (2) On procède par induction sur $m_1 \preceq_{pref}^2 m_2$.

(B) Si $m_1 = \varepsilon$, alors m_3 existe donc bien, il s'agit du mot m_2 .

(I) Sinon $m_1 = am'_1$ et $am'_1 \preceq_{pref}^2 am'_2$ avec $m'_1 \preceq_{pref}^2 m'_2$; alors, par hypothèse d'induction, il existe un mot m'_3 tel que $m'_1.m'_3 = m'_2$ et il vient $m_1m'_3 = am'_1m'_3 = am'_2 = m_2$. Le mot m_3 existe donc bien, il s'agit de m'_3 .

Exercice 6

Soit $F_0 = \{a\}$ et $F_1 = \{s\}$. On considère l'ensemble \mathcal{T} des termes construits sur $F_0 \cup F_1$.

1. Démontrer que $\mathcal{T} = \{s^n(a) \mid n \in \mathbb{N}\}$, où $s^n(a)$ est défini inductivement par :

$s^0(a) = a$ et pour tout $n \in \mathbb{N}$, $s^{n+1}(a) = s(s^n(a))$.

Solution :

L'ensemble \mathcal{T} est défini inductivement par :

(B) a appartient à \mathcal{T}

(I) pour tout $t \in \mathcal{T}$, $s(t)$ appartient à \mathcal{T}

Montrons d'abord que $\{s^n(a) \mid n \in \mathbb{N}\}$ est inclus dans \mathcal{T} , par récurrence sur n . Pour $n = 0$, comme $s^0(a) = a$, c'est vrai d'après (B). Supposons que $s^n(a)$ appartient à \mathcal{T} . Alors, d'après (I), $s(s^n(a))$ appartient aussi à \mathcal{T} , mais $s(s^n(a)) = s^{n+1}(a)$ d'où le résultat.

Pour l'inclusion réciproque, on montre par induction sur la structure de \mathcal{T} que pour tout terme $t \in \mathcal{T}$, il existe k tel que $t = s^k(a)$. Pour le terme de base a , c'est vrai pour $k = 0$ puisque $a = s^0(a)$. Supposons la propriété vraie pour un terme $t = s^k(a)$. Alors $s(t) = s(s^k(a)) = s^{k+1}(a)$, donc la propriété est vraie aussi pour $s(t)$ avec $k' = k + 1$.

Sur le domaine $D = \mathbb{N}$, on associe à la constante a la valeur $a_D \in \mathbb{N}$ et au symbole de fonction s la fonction $s_D : D \longrightarrow D$; On rappelle que l'interprétation h^* des termes T (à valeur dans D) est telle que :

(B') Si $t = a \in F_0$, $h^*(t) = a_D$,

(I') Si $t = s(t_1)$, $h^*(t) = s_D(h^*(t_1))$.

Calculer h^* dans les cas suivants :

2. $a_D = 0$, $s_D(n) = n + 1$.

Solution :

Dans ces trois questions, il faut deviner la forme générale de la valeur d'un terme, en calculant $h^*(s(a))$, $h^*(s^2(a))$, etc. Le cas de base (à vérifier) est toujours obtenu par $h^*(a) = a_D$.

On vérifie par induction que $h^*(s^n(a)) = n$ pour tout n :

$$h^*(s^{n+1}(a)) = h^*(s(s^n(a))) = h^*(s^n(a)) + 1 = n + 1.$$

3. $a_D = 1$, $s_D(n) = 2n$.

Solution :

On vérifie par induction que $h^*(s^n(a)) = 2^n$ pour tout n :

$$h^*(s^{n+1}(a)) = h^*(s(s^n(a))) = 2h^*(s^n(a)) = 2 * 2^n = 2^{n+1}.$$

4. $a_D = 1$, $s_D(n) = n + 2$.

Solution :

On vérifie par induction que $h^*(s^n(a)) = 2n + 1$ pour tout n :

$$h^*(s^{n+1}(a)) = h^*(s(s^n(a))) = h^*(s^n(a)) + 2 = 2n + 1 + 2 = 2(n + 1) + 1.$$

Langages et automates

Auteurs

IG, transformé en L^AT_EX par VMM, révision MB/BB

Version du 27 juillet 2012

1 Langages

Rappels. Un langage sur un alphabet A est une partie de A^* . Si L_1 et L_2 sont deux langages de A^* , on définit leur concaténation par :

$$L_1.L_2 = \{u.v \mid u \in L_1, v \in L_2\}$$

La concaténation de langages est une opération associative admettant $\{\varepsilon\}$ comme élément neutre (voir exercice ci-dessous). On peut alors définir les puissances d'un langage L par : $L^0 = \{\varepsilon\}$ et $L^{n+1} = L^n.L = L.L^n$. Enfin, l'étoile d'un langage L est le sous-monoïde de A^* engendré par L , c'est-à-dire :

$$L^* = \bigcup_{n \in \mathbb{N}} L^n$$

Exercice 1

1. Soit A un alphabet. Montrer que $(\mathcal{P}(A^*), \cdot, \{\varepsilon\})$ est un monoïde.

Solution :

Le produit de langages est associatif :

$$(L \cdot L') \cdot L'' = L \cdot (L' \cdot L'') = \{uu'u'' \mid u \in L, u' \in L', u'' \in L''\}.$$

Le langage $\{\varepsilon\}$ est bien l'élément neutre puisque

$$L \cdot \{\varepsilon\} = \{uv \mid u \in L, v \in \{\varepsilon\}\} = \{u\varepsilon \mid u \in L\} = L.$$

Pour les mêmes raisons, $L = \{\varepsilon\} \cdot L$.

2. Montrer que si $(L_i)_{i \in I}$ est une famille quelconque de langages, alors

$$\left(\bigcup_{i \in I} L_i\right) \cdot L = \bigcup_{i \in I} (L_i \cdot L)$$

Solution :

$u \in \left(\bigcup_{i \in I} L_i\right) \cdot L$ si et seulement si il existe $v \in \bigcup_{i \in I} L_i$, il existe $w \in L$ tels que $u = vw$. Ceci est équivalent à : il existe $i \in I$ tel que $v \in L_i$, il existe $w \in L$ tels que $u = vw$. Donc $u \in \left(\bigcup_{i \in I} L_i\right) \cdot L$ si et seulement si il existe $i \in I$ tel que $u \in L_i \cdot L$.

3. Montrer que $L^* = (L + \{\varepsilon\})^*$ et que $L^* = \{\varepsilon\} + L \cdot L^*$.

Solution :

On démontre aisément par induction que $(L + \{\varepsilon\})^n = \bigcup_{i=0}^n L^i$. C'est vrai pour $n = 0$. En utilisant l'hypothèse d'induction, $(L + \{\varepsilon\})^{n+1} = \left(\bigcup_{i=0}^n L^i\right) \cdot (L + \{\varepsilon\})$ qui est égal, en utilisant le résultat de la question précédente et le fait que $\{\varepsilon\}$ est l'élément neutre du produit, à $\bigcup_{i=0}^{n+1} L^i$.

4. Montrer que $\emptyset^* = \{\varepsilon\}$.

Solution :

$$\emptyset^* = \{\varepsilon\} + \emptyset \cdot L^* = \{\varepsilon\}.$$

Exercice 2

Soit A un alphabet contenant la lettre b . Soit $X = \{b\}$ et $Y = (A \setminus \{b\}) \cdot \{b\}^*$.

1. Décrire informellement les éléments de X^* , Y et Y^* .

Solution :

X^* est l'ensemble des mots (y compris le mot vide) formés uniquement de b .

Y est l'ensemble des mots non vides dont la première lettre n'est pas un b et dont toutes les autres sont des b . En effet $u \in Y$ si et seulement si $u = vw$ avec $v \in A \setminus \{b\}$ et $w \in \{b\}^*$.

Y^* est l'ensemble formé du mot vide et de tous les mots non vides ne commençant pas par b .

2. Montrer que tout mot de A^* commençant par une lettre distincte de b appartient à Y^* .

Solution :

Soit un mot u de A^* dont la première lettre n'est pas un b . Il s'écrit donc

$$x_1 b^{p_1} x_2 b^{p_2} \dots b^{p_{n-1}} x_n b^{p_n},$$

avec $x_i \in A \setminus \{b\}$. Chacun des mots $x_i b^{p_i}$ est dans Y donc $u \in Y^*$.

3. Montrer que tout mot u de A^* s'écrit de façon unique sous la forme $u = vw$, où $v \in X^*$ et $w \in Y^*$.

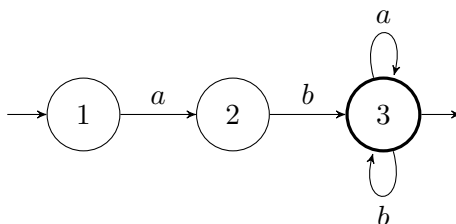
Solution :

Soit u un mot de A^* . S'il contient au moins une lettre différente de b , il s'écrit $b^p x u'$ avec $x \neq b$. D'après 1), $b^p \in X^*$, et d'après 2), $x u' \in Y^*$. Il est clair que la décomposition est unique. En effet, si $u = vw$ alors $w \in Y^*$ commence par une lettre y différente de b ; mais comme $v \in X^*$ ne contient que des b , il en résulte que y est la première lettre de u différente de b donc $x = y$, $v = b^p$ et $w = x u'$.

Si u ne contient que des b , alors il appartient à X^* , et comme $\varepsilon \in Y^*$, $u \in X^* \cdot Y^*$. Cette décomposition est unique : puisque u ne contient que des b , le seul mot de Y^* qui peut apparaître dans u est le mot vide.

2 Automates complets/déterministes**Exercice 3**

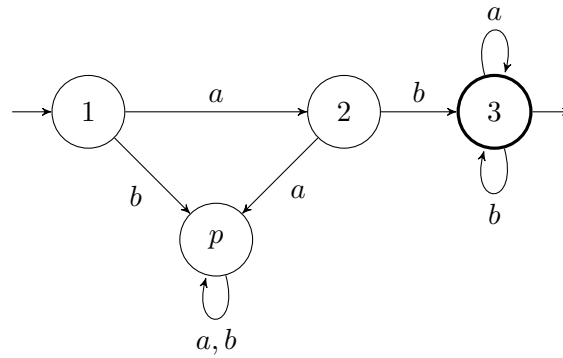
Expliquez pourquoi l'automate suivant sur $\{a, b\}$ n'est pas complet. Quel langage reconnaît-il ? Donnez un automate complet équivalent.

**Solution :**

Il n'y a pas de chemin partant de l'état 1 et étiqueté b , ni de chemin partant de l'état 2 et étiqueté a .

L'automate reconnaît l'ensemble des mots commençant par ab , c'est-à-dire le langage $ab(a+b)^*$.

On complète en ajoutant un état-puits comme suit.



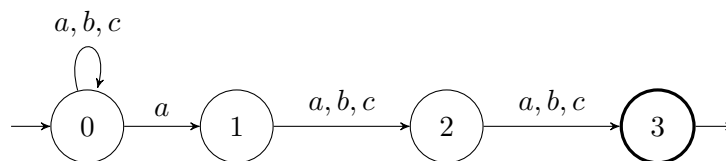
Exercice 4

Représenter l'automate \mathcal{A} sur l'alphabet $\{a, b, c\}$ d'états 0, 1, 2, 3, d'état initial 0, d'état terminal 3 et de transitions $(0, a, 0)$, $(0, a, 1)$, $(0, b, 0)$, $(0, c, 0)$, $(1, a, 2)$, $(1, b, 2)$, $(1, c, 2)$, $(2, a, 3)$, $(2, b, 3)$, $(2, c, 3)$.

- Cet automate est-il complet ? déterministe ? justifier.
- Les mots *baba* et *cabcb* sont-ils reconnus par \mathcal{A} ?
- Décrire $L(\mathcal{A})$ en langage ordinaire.

Solution :

Remarque : il s'agit de l'exercice 4 du TME.



L'automate \mathcal{A} n'est pas complet car aucune transition ne part de l'état 3. Il n'est pas déterministe car il existe deux transitions d'origine 0 et d'étiquette *a*.

Le mot *baba* $\in L(\mathcal{A})$ mais pas le mot *cabcb*.

Le langage $L(\mathcal{A})$ des mots acceptés par \mathcal{A} est l'ensemble des mots qui se terminent par un *a* suivi de deux lettres (les mots dont l'antépénultième lettre est un *a*) : $L(\mathcal{A}) = \{a, b, c\}^* a \{a, b, c\}^2$.

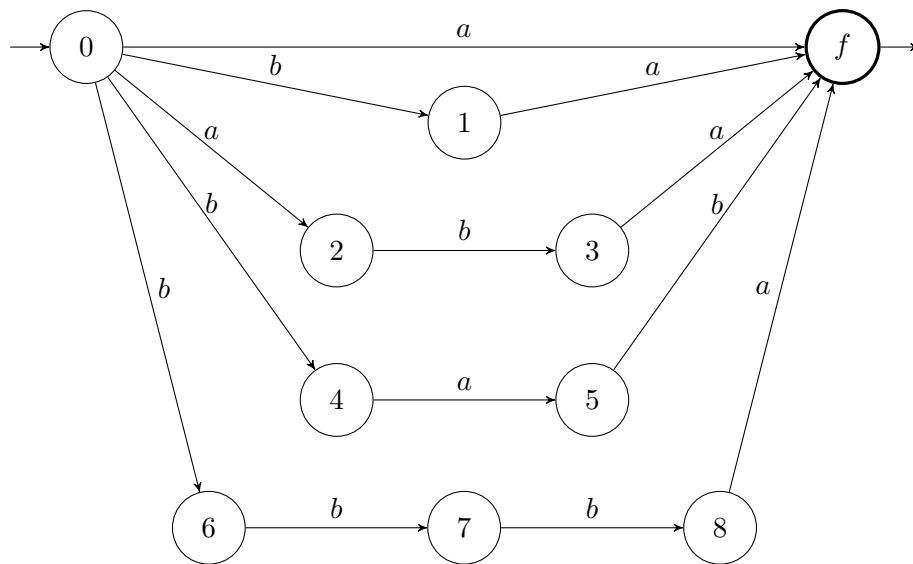
3 Construction d'automates

Exercice 5

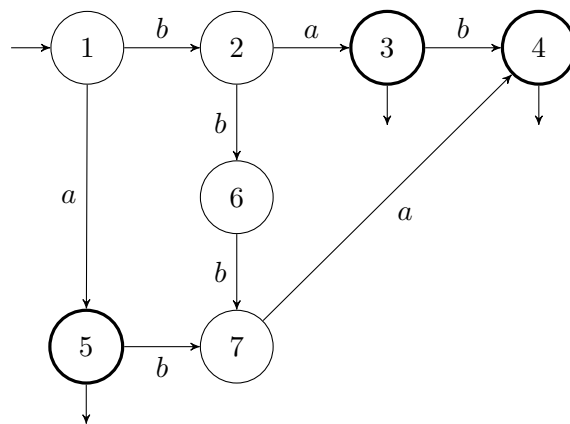
Construire un automate déterministe reconnaissant le langage fini : $\{a, ba, aba, bab, bbba\}$.

Solution :

Tout d'abord un automate non déterministe :



et puis la version optimisée déterministe :



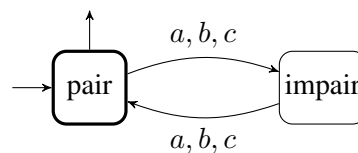
Exercice 6

Soit $A = \{a, b, c\}$. Donner des automates finis reconnaissant les langages suivants.

1. L'ensemble des mots de longueur paire.

Solution :

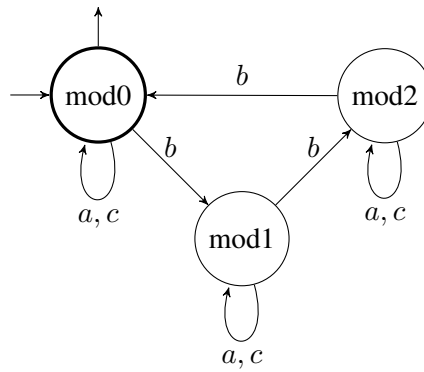
On obtient directement un automate déterministe complet :



2. L'ensemble des mots où le nombre d'occurrences de "b" est divisible par 3.

Solution :

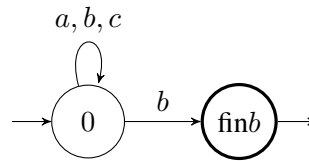
Là aussi, on obtient directement un automate déterministe complet :



3. L'ensemble des mots se terminant par "b".

Solution :

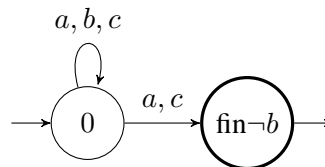
Ici, le plus simple est de construire un automate non déterministe :



4. L'ensemble des mots non vides ne se terminant pas par "b".

Solution :

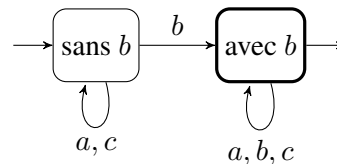
Ici aussi, le plus simple est de construire un automate non déterministe. Attention, ce n'est pas le complémentaire du précédent. En effet, le mot vide ϵ ne se termine pas par "b".



5. L'ensemble des mots contenant au moins un "b".

Solution :

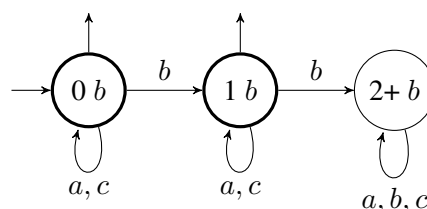
La transition repère le premier "b" :



6. L'ensemble des mots contenant au plus un "b".

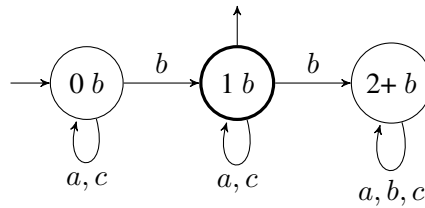
Solution :

Les trois automates qui suivent sont déterministes et complets, les états $2+b$ ou $1+b$ étant des puits.



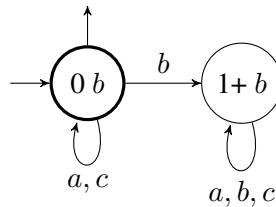
7. L'ensemble des mots contenant exactement un "b".

Solution :



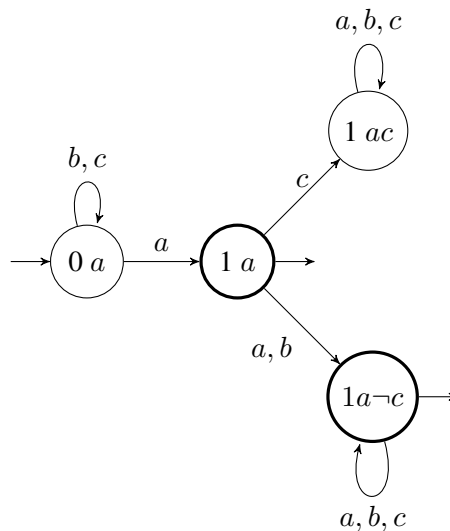
8. L'ensemble des mots ne contenant aucun "b".

Solution :



9. L'ensemble des mots contenant au moins un "a" et dont la première occurrence de "a" n'est pas suivie par un "c".

Solution :

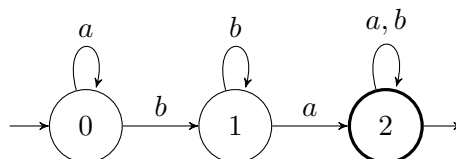


4 Opérations

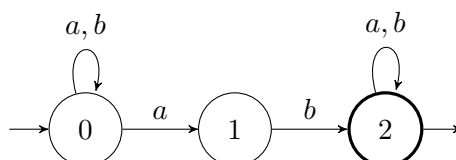
Exercice 7 – Intersection et déterminisation

On considère les automates \mathcal{A}_1 et \mathcal{A}_2 suivants sur l'alphabet $\{a, b\}$.

L'automate \mathcal{A}_1



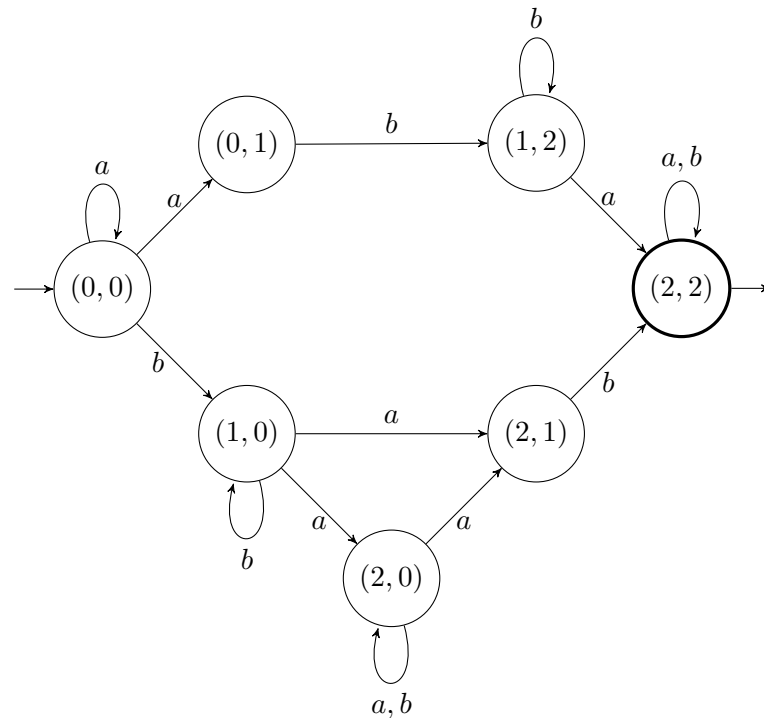
L'automate \mathcal{A}_2



1. Construire à partir de \mathcal{A}_1 et \mathcal{A}_2 un automate acceptant l'intersection $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

Solution :

La construction pour l'intersection est celle d'un automate produit : les états sont les couples de $S_1 \times S_2$ en notant S_1 l'ensemble des états de \mathcal{A}_1 et S_2 l'ensemble des états de \mathcal{A}_2 . Les états terminaux sont les couples formés d'un état terminal de \mathcal{A}_1 et d'un état terminal de \mathcal{A}_2 . On obtient alors :



2. Les automates \mathcal{A}_1 et \mathcal{A}_2 sont-ils déterministes ? Expliquez pourquoi et si ce n'est pas le cas, déterminez-les.

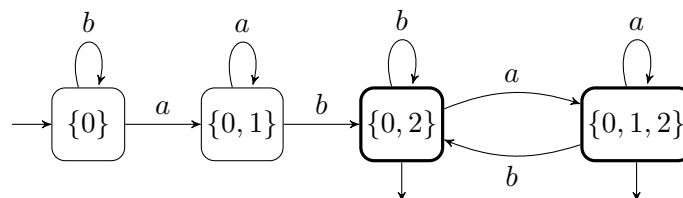
Solution :

\mathcal{A}_1 est déterministe mais \mathcal{A}_2 ne l'est pas car il y a deux transitions étiquetées par b au départ de l'état 1. Remarquons que la construction précédente d'intersection s'applique telle quelle.

Appelons S_2 l'ensemble $\{0, 1, 2\}$ des états de \mathcal{A}_2 . Dans la procédure de déterminisation, l'ensemble des états du nouvel automate déterministe \mathcal{D}_2 à construire est un sous-ensemble de $\mathcal{P}(S_2)$, l'ensemble des parties de S_2 . Si P est une partie de S_2 , et ℓ une lettre, alors l'état P' obtenu par la transition $P \xrightarrow{\ell} P'$ contient tous les états de S_2 images par ℓ des états de P . Par exemple, si deux transitions étiquetées par la même lettre ℓ partent d'un unique état initial $s_0 : s_0 \xrightarrow{\ell} s_1$ et $s_0 \xrightarrow{\ell} s_2$, on obtient la transition $\{s_0\} \xrightarrow{\ell} \{s_1, s_2\}$. Une partie P est un état final de \mathcal{D}_2 si elle contient au moins un état final de \mathcal{A}_2 .

Notons que l'automate déterminisé peut donc contenir au maximum 2^n états si l'automate de départ a n états.

Le déterminisé de \mathcal{A}_2 est donc le suivant :



3. Les automates \mathcal{A}_1 , \mathcal{A}_2 et les automates déterministes construits sont-ils complets ? Que remarquez-vous ?

Solution :

L'automate \mathcal{A}_1 est complet, mais \mathcal{A}_2 ne l'est pas car il n'y a pas de transition étiquetée par a au départ de l'état 1. Par contre on remarque que la déterminisation nous a fourni un automate qui est en plus complet.

Dans le cas général, le sous-ensemble \emptyset de l'ensemble d'états de l'automate de départ fournit toujours un puits potentiel.

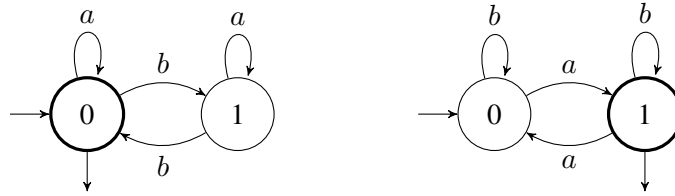
Exercice 8 – Intersection et concaténation

Sur $A = \{a, b\}$, soient L_1 le langage comprenant tous les mots contenant un nombre pair de b et L_2 le langage comprenant tous les mots contenant un nombre impair de a .

1. Donner pour chaque L_i un automate \mathcal{A}_i reconnaissant L_i .

Solution :

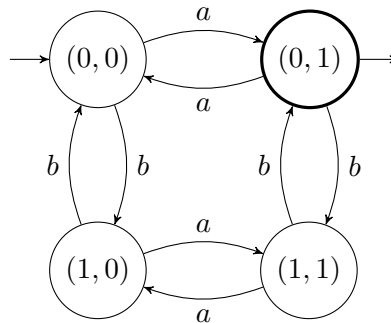
Les automates ci-dessous acceptent L_1 (à gauche) et L_2 (à droite) :



2. Calculer à partir des \mathcal{A}_i un automate reconnaissant $L_1 \cap L_2$.

Solution :

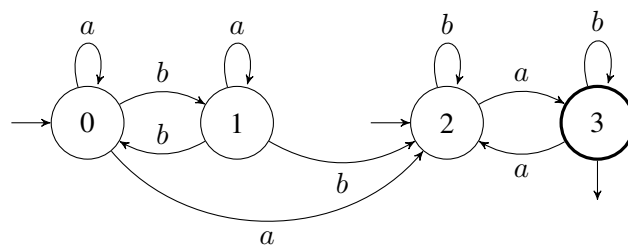
La construction déjà expliquée donne l'automate suivant pour $L_1 \cap L_2$:



3. (**Concaténation**) Construire à partir des \mathcal{A}_i un automate reconnaissant $L_1.L_2$.

Solution :

Il faut renommer les états de \mathcal{A}_2 et ajouter vers l'état initial du second automate les transitions allant vers un état final du premier. De plus, ici, il faut garder initial l'état 2 (initial de \mathcal{A}_2) car L_1 contient le mot vide.

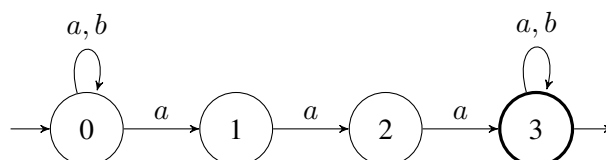


Exercice 9

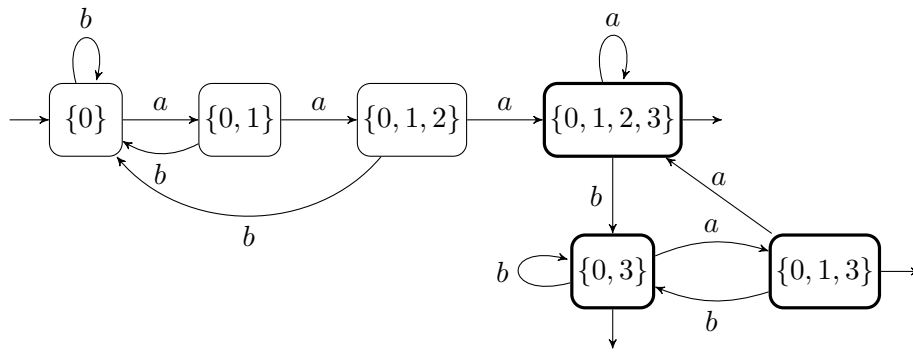
Soit $A = \{a, b\}$ et soit L le langage comprenant tous les mots ayant trois occurrences successives de "a". Donner un automate non déterministe reconnaissant L et construire un automate déterministe acceptant L .

Solution :

Un automate non déterministe \mathcal{A} :



Son déterminisé \mathcal{D} :



Exercice 10 – Complémentaire et différence

- Soient L_1 et L_2 des langages sur un alphabet A . Montrer que si L_1 et L_2 sont respectivement reconnaissables par des automates \mathcal{A}_1 et \mathcal{A}_2 alors le langage $L_1 \setminus L_2$ est reconnaissable par un automate.

Solution :

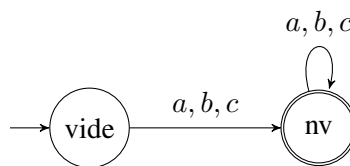
La première étape est d'obtenir à partir de \mathcal{A}_2 un automate **déterministe et complet** $\mathcal{D}_2 = (S, T, \{s_0\}, F)$ acceptant L_2 . Alors le langage $L_3 = A^* \setminus L_2$ (complémentaire de L_2) est reconnu par $\mathcal{A}_3 = (S, T, \{s_0\}, S \setminus F)$. Il suffit ensuite de construire à partir de \mathcal{A}_1 et de \mathcal{A}_3 l'automate pour l'intersection de L_1 et de L_3 par la méthode classique. Si \mathcal{A}_2 est déjà déterministe et complet, la première étape est omise.

- Construire un automate déterministe sur l'alphabet $A = \{a, b, c\}$ pour l'ensemble des mots non vides ne se terminant pas par "b". Cette construction sera faite de deux façons.
 - En utilisant le résultat ci-dessus à partir d'un automate \mathcal{A}_1 acceptant les mots non vides et de l'automate non déterministe (qu'on appellera \mathcal{A}_2) de l'exercice 6.3.
 - En déterminisant l'automate (qu'on appellera \mathcal{A}_4) de l'exercice 6.4.

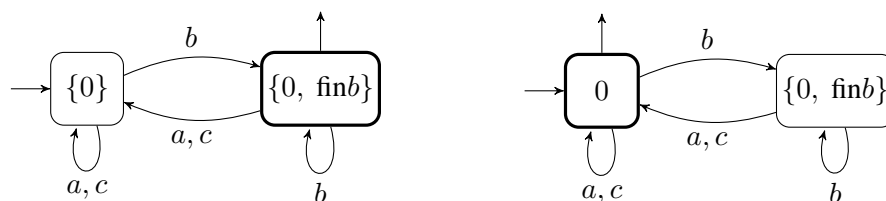
Solution :

- Méthode 1.

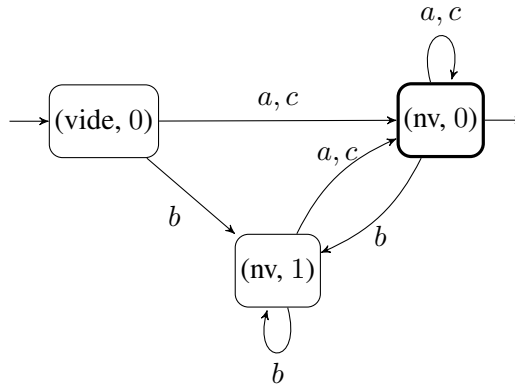
L'automate \mathcal{A}_1 acceptant les mots non vides sur $A = \{a, b, c\}$:



A gauche le déterminisé \mathcal{D}_2 de \mathcal{A}_2 , à droite l'automate \mathcal{A}_3 pour le langage complémentaire (remarquer qu'il contient le mot vide).

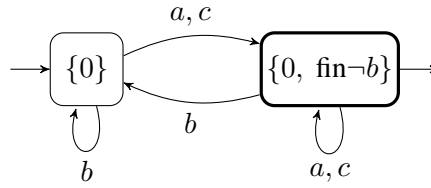


Puis, l'automate pour l'intersection de L_1 et de L_3 , en renommant 1 l'état $\{0, \text{finb}\}$:



2. Méthode 2.

L'automate \mathcal{D}_4 , déterminisé de \mathcal{A}_4 (qui est minimal, voir plus loin) :



5 Systèmes d'équations et expressions rationnelles

Exercice 11 – Lemme d'Arden

Soient K et M deux langages de A^* tels que $\varepsilon \notin K$, alors l'équation $X = K.X + M$ (qui s'écrit aussi $X = K.X \cup M$, la notation $+$ représentant l'union) admet pour unique solution le langage $K^*.M$.

Solution :

On vérifie que $K^*.M$ est solution de $X = K.X \cup M$:

$$\begin{aligned}
 K^*.M &= \bigcup_{n \in \mathbb{N}} K^n.M \\
 &= \left(K^0 \cup \bigcup_{n \geq 1} K^n \right).M \\
 &= \left(\{\varepsilon\} \cup \bigcup_{n \in \mathbb{N}} K.K^n \right).M \\
 &= (\{\varepsilon\} \cup K.K^*).M \\
 &= (\{\varepsilon\}.M) \cup (K.K^*.M) \\
 &= M \cup (K.K^*.M) \\
 &= (K.K^*.M) \cup M.
 \end{aligned}$$

Montrons à présent que c'est l'unique solution. Soit X un langage de A^* vérifiant $X = K.X \cup M$, montrons donc que $K^*.M = X$ ce qui revient à montrer deux inclusions.

(1) Pour montrer $K^*.M \subseteq X$, on montre par récurrence sur n que :

$$\forall n \in \mathbb{N} \quad K^n.M \subseteq X$$

(B) pour $n = 0$, on a $K^0.M = \{\varepsilon\}.M = M \subseteq K.X \cup M = X$

(I) on suppose la propriété vraie pour n et on la montre pour $n + 1$:

$$K^{n+1}.M = K.K^n.M \subseteq K.K^n.M \cup M \subseteq K.X \cup M = X$$

(2) Pour montrer $X \subseteq K^*.M$, on montre par récurrence que pour tout entier n ,

$$\text{si } w \in X \text{ et } \lg(w) = n \text{ alors } w \in K^*.M$$

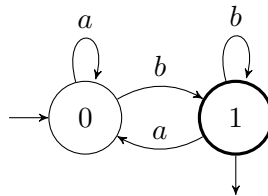
où $\lg(w)$ désigne la longueur du mot w (c'est-à-dire son nombre de lettres). Soit donc un entier n , supposons la propriété vérifiée pour tous les entiers $k < n$ et montrons la pour n . Soit donc $w \in X$ tel que $\lg(w) = n$. Puisque $X = K.X \cup M$, deux cas se présentent :

- (i) $w \in M$ et on peut conclure puisque $M \subseteq K^*.M$
- (ii) $w \in K.X$ et puisque $\varepsilon \notin K$, il existe $(u, v) \in K \times X$ tel que $w = u.v$ et $\lg(u) \neq 0$, il vient donc $\lg(v) = \lg(w) - \lg(u) < \lg(w)$ et, par hypothèse d'induction, on a $v \in K^*.M$ et donc $u.v \in K.K^*.M \subseteq K^*.M$

Exercice 12

- Soit l'automate \mathcal{A} d'états 0, 1, d'état initial 0, d'état terminal 1 et de transitions $(0, a, 0)$, $(0, b, 1)$, $(1, a, 0)$ et $(1, b, 1)$. Dessiner l'automate \mathcal{A} . Soit L le langage reconnu par \mathcal{A} . Donner le système d'équations associé à \mathcal{A} et en déduire une expression rationnelle pour L .

Solution :



Les équations :

$$L_0 = aL_0 + bL_1 \quad (1)$$

$$L_1 = aL_0 + bL_1 + \varepsilon \quad (2)$$

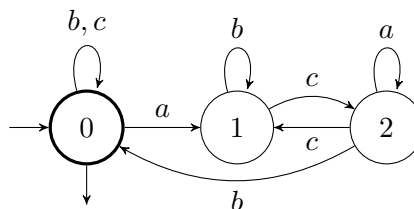
D'après le lemme d'Arden appliqué à (2), on a $L_1 = b^*(aL_0 + \varepsilon)$, combiné avec (1) on a :

$$L_0 = (a + bb^*a)L_0 + bb^* \quad (3)$$

On applique le lemme d'Arden à cette nouvelle équation et on obtient $L_0 = (a + bb^*a)^*bb^*$ ou encore $L_0 = ((\varepsilon + bb^*)a)^*bb^* = (b^*a)^*bb^*$.

- Mêmes questions avec \mathcal{B} d'états 0, 1, 2 d'état initial 0, d'état terminal 0 et de transitions $(0, a, 1)$, $(0, b, 0)$, $(0, c, 0)$, $(1, b, 1)$, $(1, c, 2)$, $(2, a, 2)$, $(2, b, 0)$, $(2, c, 1)$.

Solution :



Les équations :

$$L_0 = (b + c)L_0 + aL_1 + \varepsilon \quad (4)$$

$$L_1 = bL_1 + cL_2 \quad (5)$$

$$L_2 = bL_0 + cL_1 + aL_2 \quad (6)$$

Méthode “de bas en haut”.

En appliquant le lemme d'Arden à (6), on obtient : $L_2 = a^*(bL_0 + cL_1)$. En remplaçant dans (5), on a :

$L_1 = bL_1 + ca^*bL_0 + ca^*cL_1 = (b + ca^*c)L_1 + ca^*bL_0$, d'où $L_1 = (b + ca^*c)^*ca^*bL_0$ par le lemme d'Arden. En remplaçant maintenant dans (4), on obtient :

$$L_0 = (b + c)L_0 + a(b + ca^*c)^*ca^*bL_0 + \varepsilon = (b + c + a(b + ca^*c)^*ca^*b)L_0 + \varepsilon \quad (7)$$

donc par le lemme d'Arden, $L_0 = [b + c + a(b + ca^*c)^*ca^*b]^*$.

Autre méthode.

On applique le lemme d'Arden à (5) et on obtient (5') : $L_1 = b^*cL_2$. On combine ce résultat avec (6) ce qui nous donne $L_2 = (a + cb^*c)L_2 + bL_0$.

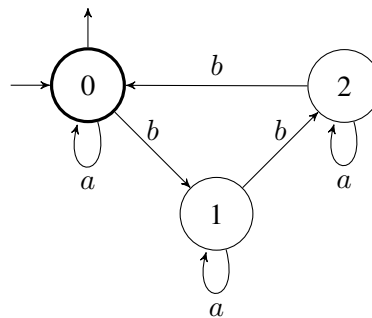
Le lemme d'Arden appliqué à cette nouvelle équation nous donne $L_2 = (a + cb^*c)^*bL_0$. On combine à nouveau ce résultat avec (5') puis (4) ce qui nous donne $L_0 = (b + c + ab^*c(a + cb^*c)^*b)L_0 + \varepsilon$ et on applique une dernière fois le lemme d'Arden, d'où $L_0 = [b + c + ab^*c(a + cb^*c)^*b]^*$.

Exercice 13

Soit L l'ensemble des mots sur l'alphabet $\{a, b\}$ où le nombre d'occurrences de "b" est divisible par 3. Il y a un automate \mathcal{A} à trois états tel que $L = L(\mathcal{A})$ (cf. exercice 6.2). Donner le système d'équations associé à l'automate, et résoudre ce système, pour donner une expression rationnelle dénotant L .

Solution :

Rappel de l'automate en l'adaptant à l'alphabet :



Les équations :

$$L_0 = aL_0 + bL_1 + \varepsilon \quad (8)$$

$$L_1 = aL_1 + bL_2 \quad (9)$$

$$L_2 = aL_2 + bL_0 \quad (10)$$

où L_i désigne le langage reconnu à partir de l'état i considéré comme initial. On doit calculer $L = L_0$ puisque 0 est l'état initial de \mathcal{A} .

Méthode "de bas en haut".

En appliquant le lemme d'Arden à l'équation (10), on obtient : $L_2 = a^*bL_0$, qu'on remplace dans l'équation (9).

Ceci donne : $L_1 = aL_1 + ba^*bL_0$, auquel on applique à nouveau le lemme d'Arden : $L_1 = a^*ba^*bL_0$.

On remplace enfin cette expression de L_1 dans l'équation (8) : $L_0 = aL_0 + ba^*ba^*bL_0 + \varepsilon = (a + ba^*ba^*b)L_0 + \varepsilon$, et on applique une troisième fois le lemme d'Arden, ce qui donne : $L_0 = (a + ba^*ba^*b)^*$ puisque ε est neutre pour la concaténation. On reconnaît dans cette expression rationnelle les deux "boucles" possibles à partir de l'état 0.

Autre méthode possible. On déduit du lemme d'Arden

- appliqué à l'équation (8) : $L_0 = a^*(bL_1 + \varepsilon)$
- appliqué à l'équation (9) : $L_1 = a^*bL_2$,
- appliqué à l'équation (10) : $L_2 = a^*bL_0$,

d'où $L_0 = a^* + a^*bL_1 = a^* + a^*ba^*bL_2 = a^* + a^*ba^*ba^*bL_0$ et en appliquant une dernière fois le lemme d'Arden on obtient $L_0 = (a^*ba^*ba^*b)^*a^*$.

Exercice 14

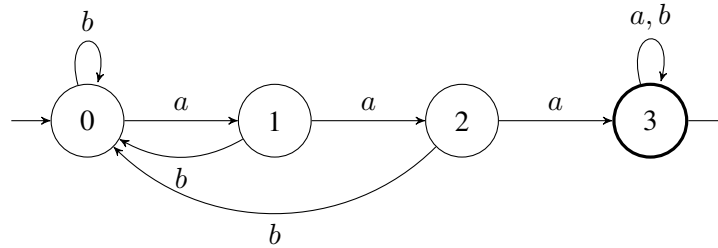
Soit $A = \{a, b\}$ et L le langage comprenant tous les mots ayant trois occurrences successives de "a".

1. Donner une expression rationnelle pour L associée à l'automate non déterministe obtenu à l'exercice 9.

Solution :

L'automate non déterministe donne immédiatement l'expression rationnelle $(a + b)^*a^3(a + b)^*$.

2. On admet que l'automate minimal \mathcal{M} de L est le suivant :



Calculer une autre expression rationnelle pour L à partir de \mathcal{M} .

Solution :

Pour l'automate \mathcal{M} , on obtient les équations :

$$L_0 = bL_0 + aL_1 \quad (11)$$

$$L_1 = bL_0 + aL_2 \quad (12)$$

$$L_2 = bL_0 + aL_3 \quad (13)$$

$$L_3 = (a + b)L_3 + \varepsilon \quad (14)$$

Méthode “de bas en haut”.

Le lemme d'Arden appliqué à (14) donne $L_3 = (a + b)^*$.

En remplaçant dans (13), on a : $L_2 = bL_0 + a(a + b)^*$ qu'on peut remplacer dans (12), d'où $L_1 = bL_0 + abL_0 + aa(a + b)^*$.

En remplaçant dans (11), on obtient : $L_0 = bL_0 + abL_0 + aabL_0 + a^3(a + b)^* = (b + ab + a^2b)L_0 + a^3(a + b)^*$, d'où finalement en réappliquant le lemme d'Arden : $L_0 = (b + ab + a^2b)^*a^3(a + b)^*$.

Autre méthode possible.

D'après le lemme d'Arden appliqué à (11), on a $L_0 = b^*aL_1$.

En utilisant (12) puis (13), on a

$$L_0 = b^*a(bL_0 + aL_2) = b^*abL_0 + b^*a^2L_2 = b^*abL_0 + b^*a^2bL_0 + b^*a^3L_3 = (b^*ab + b^*a^2b)L_0 + b^*a^3L_3. \quad (15)$$

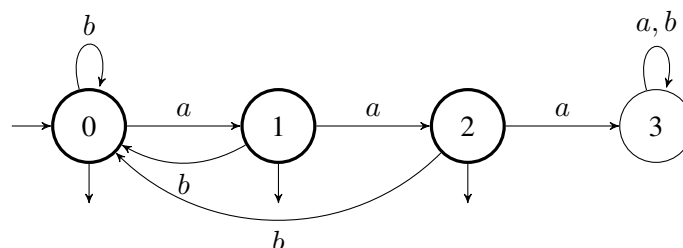
On utilise le lemme d'Arden appliqué à (14), qui nous donne $L_3 = (a + b)^*$.

On applique le lemme d'Arden à (15) et on obtient $L_0 = (b^*ab + b^*a^2b)^*b^*a^3(a + b)^*$.

3. Donner un automate déterministe pour le complémentaire de L , c'est-à-dire l'ensemble des mots sur l'alphabet $A = \{a, b\}$ qui n'ont pas trois occurrences successives de “a”. En déduire une expression rationnelle pour le complémentaire de L .

Solution :

L'automate cherché accepte le complémentaire du langage de l'exercice précédent, on l'obtient donc en échangeant les états finals et non finals, ce qui transforme l'état 3 en puits :



On peut donc supprimer l'état 3, ce qui fournit le système d'équations :

$$L_0 = bL_0 + aL_1 + \varepsilon \quad (16)$$

$$L_1 = bL_0 + aL_2 + \varepsilon \quad (17)$$

$$L_2 = bL_0 + \varepsilon \quad (18)$$

On remplace L_2 dans (17), ce qui donne $L_1 = bL_0 + abL_0 + a + \varepsilon = (b + ab)L_0 + a + \varepsilon$, puis on remplace L_1 dans (16) :

$L_0 = bL_0 + a(b + ab)L_0 + a^2 + a + \varepsilon = (b + ab + a^2b)L_0 + a^2 + a + \varepsilon$, et en appliquant le lemme d'Arden : $L_0 = (b + ab + a^2b)^*(\varepsilon + a + a^2)$.

Noter que l'équation qui correspondrait à l'état 3 serait $L_3 = (a+b)L_3$ qui est de la forme $L_3 = KL_3 + \emptyset$. L'application du lemme d'Arden donne $L_3 = K^*.\emptyset = \emptyset$, ce qui est normal puisque cet état n'est pas final et ne mène nulle part.

6 Langages reconnaissables

Exercice 15 – Préfixe

Soit L un langage et $\text{Pref}(L) = \{u \in A^* \text{ tel que } \exists v \in A^* : uv \in L\}$ l'ensemble des préfixes des mots de ce langage L . Montrer que si un langage L est reconnaissable l'ensemble $\text{Pref}(L)$ est aussi reconnaissable.

Solution :

Soit $\mathcal{A} = (S, T, I, F)$ un automate reconnaissant L , on obtient un automate $\mathcal{A}' = (S, T, I, F')$ reconnaissant $\text{Pref}(L)$ en mettant dans F' tout état s de \mathcal{A} tel qu'il existe un chemin c d'origine un état de I et de but s , et un chemin c' d'origine s et de but un état de F .

Exercice 16

Montrer que le langage $L = \{a^n b^n, n \geq 0\}$ sur l'alphabet $\{a, b\}$ ne peut pas être reconnu par un automate fini.

Solution :

Par l'absurde, on suppose qu'il existe un automate déterministe complet $\mathcal{A} = (S, T, \{s_0\}, F)$ reconnaissant L . Soit N le nombre d'états de cet automate. On considère alors le mot $w = a^N b^N$ de L , et le chemin associé dans l'automate : $s_0 \xrightarrow{a} s_1 \xrightarrow{a} s_2 \cdots \xrightarrow{a} s_N \xrightarrow{b} \cdots s_f$ où s_f est un état final. Parmi les $N + 1$ états s_0, \dots, s_N , il y en a deux égaux : $s_i = s_j$ pour $i < j$ (ce qui correspond à une boucle dans l'automate). Donc on peut réécrire le chemin acceptant w en : $s_0 \xrightarrow{a^i} s_i \xrightarrow{a^{j-i}} s_j \xrightarrow{a^{N-j}} s_N \xrightarrow{b^N} s_f$. Le mot a^{j-i} , qui est l'étiquette de la boucle, peut être supprimé (ou répété un nombre arbitraire de fois), ce qui produit des mots w' acceptés par l'automate, mais qui n'appartiennent pas à L puisqu'ils n'ont pas le même nombre de a et de b . On obtient donc une contradiction.

Ce raisonnement se généralise en une condition nécessaire pour qu'un langage soit reconnaissable, donnée dans l'exercice suivant.

Exercice 17 – Lemme de l'étoile

Soit L un langage reconnaissable par un automate fini. Montrer qu'il existe un entier N_0 tel que pour tout mot $w \in L$ vérifiant $|w| \geq N_0$ (où $|w|$ est la longueur de w), on a $w = w_1 u w_2$ avec

- (i) $u \neq \varepsilon$,
- (ii) $|u| < N_0$,
- (iii) $w_1 u^* w_2 \subseteq L$.

Solution :

Soit $\mathcal{A} = (S, T, s_0, F)$ un automate déterministe complet reconnaissant L . Prenons pour N_0 le nombre d'états de \mathcal{A} .
 Considérons un mot de $w = a_1 \dots a_p$ de L tel que $p \geq N_0$ et le chemin associé $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \dots \xrightarrow{a_p} s_p$ avec $s_p \in F$.
 La suite s_0, s_1, \dots, s_p comporte $p + 1 > N_0$ éléments donc il existe $i < j$ tels que $s_i = s_j$.
 Il y a donc dans \mathcal{A} une "boucle"

$$s_i \xrightarrow{a_{i+1}} s_{i+1} \xrightarrow{a_{i+2}} s_{i+2} \dots s_{j-1} \xrightarrow{a_j} s_j = s_i$$

Remarquons qu'il est possible de choisir i et j de sorte que $j - i < N_0$ (en appliquant un nombre fini de fois le même raisonnement). Prenons $u = x_{i+1} \dots x_j$, $w_1 = a_1 \dots a_i$ et $w_2 = a_{j+1} \dots a_p$. Le mot u vérifie bien les conditions (i), (ii) et (iii). Ainsi, tout mot de la forme $w = w_1 u^n w_2$ est reconnu en parcourant le chemin $s_0 \xrightarrow{w_1} s_i$, puis n fois la boucle, puis $s_i \xrightarrow{w_2} s_p$.

Exercice 18

Montrer, en utilisant le lemme de l'étoile, que les langages suivants sur l'alphabet $\{a, b\}$ ne peuvent pas être reconnus par un automate fini.

1. $L_1 = \{a^n b^n, n \geq 0\}$

Solution :

On utilise un raisonnement par l'absurde avec le lemme de l'étoile (un peu plus compliqué que la preuve directe car il faut distinguer plusieurs cas selon la position du facteur u dans le mot).

Supposons L_1 reconnaissable. Il existe N_0 tel que $a^{N_0} b^{N_0}$ se factorise sous la forme $w_1 u w_2$ vérifiant les conditions (i), (ii), (iii) du lemme. On est dans l'un des cas suivants :

1. $w_1 = a^{p_1}, u = a^{p_2}, w_2 = a^{p_3} b^{N_0}$ avec $p_2 > 0$ et $p_1 + p_2 + p_3 = N_0$,
2. $w_1 = a^{p_1}, u = a^{p_2} b^{q_2}, w_2 = b^{q_3}$ avec $p_2 > 0, q_2 > 0$ et $p_1 + p_2 = q_2 + q_3 = N_0$,
3. $w_1 = a^{N_0} b^{q_1}, u = b^{q_2}, w_2 = b^{q_3}$ avec $q_2 > 0$ et $q_1 + q_2 + q_3 = N_0$.

Dans les cas 1 et 3, $w_1 u^2 w_2$ n'est pas dans L car il ne contient pas le même nombre de b que de a . Dans le cas 2, $w_1 u^2 w_2$ n'est pas dans L car des lettres a apparaissent derrière des b . Ceci contredit la condition (iii).

2. $L_2 = \{a^p, p \text{ premier}\}$.

Solution :

On utilise un raisonnement par l'absurde avec le lemme de l'étoile. Si L_2 reconnaissable on a $w_1 = a^p, u = a^q, w_2 = a^r$ avec $q > 0$. Si p et r sont nuls alors $w_1 u^2 w_2 = a^{2q} \in L_2$, ce qui est absurde.

Si p ou r non nul alors $a^{p+(p+r)q+r} = w_1 u^{p+r} w_2 \in L_2$. Mais puisque $p + (p+r)q + r$ est divisible par $p + r > 0$ on aboutit encore à une absurdité.

Exercice 19

Les langages ci-dessous sont-ils reconnaissables ?

1. $L = \{a^n b^p \mid n, p \geq 0, n = p \bmod 3\}$

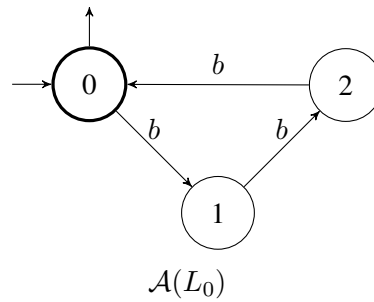
Solution :

En utilisant l'équivalence entre langages rationnels et reconnaissables : L peut s'écrire comme réunion des trois langages suivants :

- $L_0 = (b^3)^*$,
- $L_1 = ab(b^3)^*$,
- $L_2 = ab^2(b^3)^*$.

Ceux-ci étant rationnels, L est rationnel. Il est donc reconnaissable.

Sans utiliser le théorème d'équivalence, on construit facilement les automates reconnaissant les L_i à partir de



puis on peut construire un automate reconnaissant L par l'opération d'union.

2. $L' = \{a^m b^n \mid n \geq 1, n \geq 1, m \neq n\}$

Solution :

D'abord $L'_1 = \{a^n b^n, n \geq 1\}$ n'est pas reconnaissable. En effet, dans le cas contraire $L'_1 \cup \{\varepsilon\} = \{a^n b^n, n \geq 0\}$ serait aussi reconnaissable. Or un exercice précédent nous assure le contraire.

En utilisant le fait que l'ensemble des langages reconnaissables est fermé par différence (voir 10) on aboutit alors à une contradiction, puisque si L' était reconnaissable alors $L'_1 = a^+ b^+ \setminus L'$ le serait.

7 Automates minimaux

Rappelons que la minimisation part d'un automate déterministe complet dont tous les états sont accessibles depuis l'état initial et peuvent atteindre au moins un état final.

Exercice 20

Soit l'automate \mathcal{A} d'états 0, 1, 2, 3, 4, 5 d'état initial 0, d'état terminal 5 et de transitions :

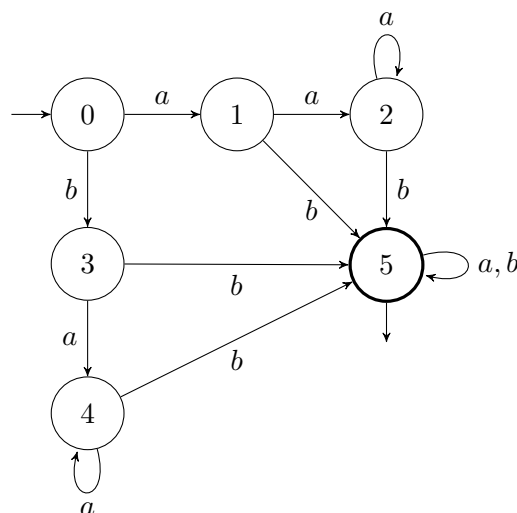
$(0, a, 1), (1, a, 2), (2, a, 2), (3, a, 4), (4, a, 4), (5, a, 5), (0, b, 3), (1, b, 5), (2, b, 5), (3, b, 5), (4, b, 5), (5, b, 5)$.

Dessiner l'automate \mathcal{A} . Minimiser \mathcal{A} .

Solution :

Remarque : il s'agit de l'exercice 8 du TME.

L'automate \mathcal{A} :



Rappel : à la première étape il y a deux classes d'équivalence : les états finaux d'une part, les autres d'autre part. Ensuite à chaque étape, on partitionne chaque classe d'équivalence en sous-classes : deux états s_1 et s_2 sont dans la même sous-classe si et seulement si pour toute lettre l , $s_1 \xrightarrow{l} s'_1$ et $s_2 \xrightarrow{l} s'_2$ avec s'_1 et s'_2 dans la même classe d'équivalence. Les sous-classes ainsi obtenues seront les classes d'équivalence de l'étape suivante. L'opération se termine lorsque les classes d'équivalence sont stationnaires d'une étape à l'autre.

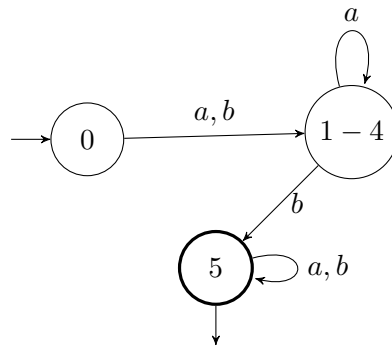
On utilise alors les classes d'équivalence comme états de l'automate minimal et il y a une transition entre deux états s_1 et s_2 dans l'automate minimal étiqueté par la lettre ℓ s'il y a des transitions entre les états de la classe d'équivalence de s_1 et ceux de la classe d'équivalence de s_2 étiquetées par ℓ .

Première étape : deux classes d'équivalence :

- les états finaux $\{5\}$
- les états non finaux $\{0, 1, 2, 3, 4\}$.

Deuxième étape : $\{0\}$, $\{1, 2, 3, 4\}$ et $\{5\}$.

puis les classes d'équivalence sont stationnaires à l'étape suivante donc l'automate minimal est :



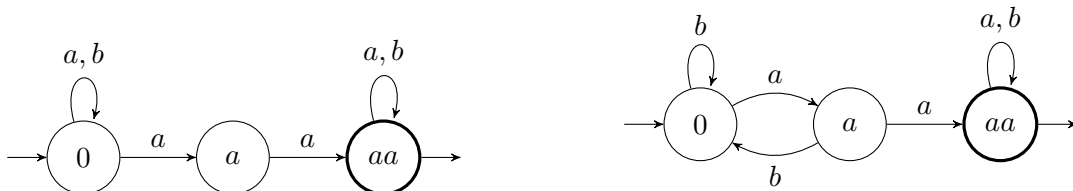
Exercice 21

Soit l'alphabet $A = \{a, b\}$. On veut calculer l'automate minimal du langage L comprenant tous les mots contenant "aa" mais ne contenant pas "bb". Pour cela, on va calculer L comme intersection des langages suivants : L_1 l'ensemble des mots contenant "aa" et L_2 l'ensemble des mots ne contenant pas "bb".

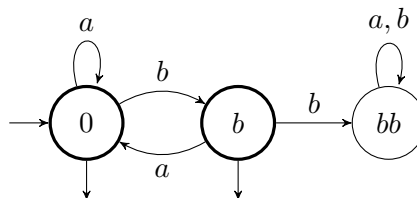
1. Construire d'abord un automate déterministe \mathcal{D}_i acceptant L_i , $i = 1, 2$.

Solution :

On obtient facilement un automate \mathcal{A}_1 non déterministe pour L_1 (ci-dessous à gauche). Remarquons que pour minimiser, on doit partir d'un automate déterministe complet. Or la construction de l'intersection préservant le déterminisme, on a intérêt ici à déterminer directement \mathcal{A}_1 (ci-dessous à droite \mathcal{D}_1).

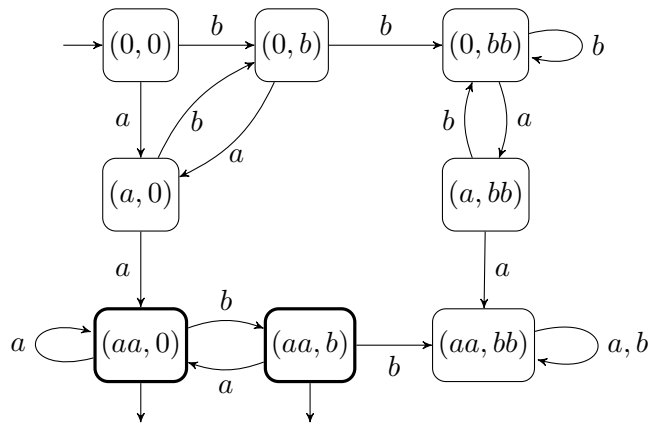


Un automate déterministe acceptant le complémentaire de L_2 est similaire à \mathcal{D}_1 en échangeant a et b . On obtient donc \mathcal{D}_2 pour L_2 :



2. Construire à partir des \mathcal{D}_i un automate déterministe et complet \mathcal{A} reconnaissant $L_1 \cap L_2$.

Solution :



On ne considère bien sûr que les états accessibles (donc (a, b) ne figure pas).

3. \mathcal{A} est-il minimal ? Justifiez votre réponse et si non, construisez un automate minimal \mathcal{B} reconnaissant $L = L_1 \cap L_2$.

Solution :

On minimise l'automate \mathcal{A} et on saura à la fin s'il était minimal ou non, selon qu'il nous reste autant de classes d'équivalence que d'états dans \mathcal{A} ou moins.

Première étape : deux classes d'équivalence :

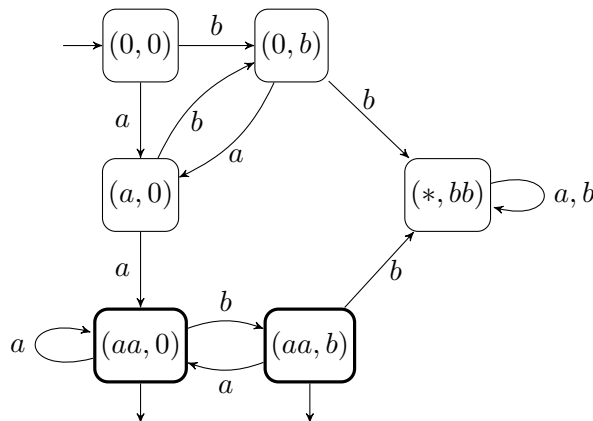
- les états finaux $\{(aa, 0), (aa, b)\}$
- les états non finaux $\{(0, 0), (0, b), (0, bb), (a, 0), (a, bb), (aa, bb)\}$.

Deuxième étape : quatre classes d'équivalence : $\{(aa, 0)\}, \{(aa, b)\}, \{(a, 0)\}$ et $\{(0, 0), (0, b), (0, bb), (a, bb), (aa, bb)\}$.

Troisième étape : cinq classes d'équivalence : $\{(aa, 0)\}, \{(aa, b)\}, \{(a, 0), \{(0, 0), (0, b)\}$ et $\{(0, bb), (a, bb), (aa, bb)\}$.

Quatrième étape : six classes d'équivalence : $\{(aa, 0)\}, \{(aa, b)\}, \{(a, 0), \{(0, 0)\}, \{(0, b)\}$ et $\{(0, bb), (a, bb), (aa, bb)\}$.

puis les classes d'équivalence sont stationnaires à l'étape suivante donc \mathcal{A} n'est pas minimal et on obtient l'automate minimal suivant :



8 Construction d'automates plus difficiles

Exercice 22

Soit $A = \{a, b, c\}$. Construire un automate déterministe complet qui reconnaît l'ensemble des mots de longueur paire qui se terminent par ab .

Solution :

On peut construire un automate non déterministe que l'on détermine et que l'on complète.

États : 0, 1, 2, 3, état initial : 0, état terminal : 3, transitions :

$(0, a, 2), (0, b, 1), (0, c, 1), (1, a, 0), (1, b, 0), (1, c, 0), (2, a, 0), (2, b, 3), (2, c, 0), (3, a, 2), (3, b, 1), (3, c, 1)$.

Exercice 23

Construire un automate déterministe complet minimal reconnaissant l'ensemble des mots sur $\{a, b, c\}$ comportant au moins trois lettres et dont la troisième lettre à partir de la fin est un "a" ou un "c".

Solution :

L'automate minimal a 8 états.

Exercice 24

Donner les automates déterministes complets minimaux reconnaissant les langages sur l'alphabet $A = \{a, b\}$ donnés par les expressions rationnelles suivantes

1. $(a + b)^* b (a + b)^*$.

Solution :

Cf. livre Figure 15.8.1 p. 365.

2. $((a + b)^2)^+ + ((a + b)^3)^+$.

Solution :

Cf. livre Figure 15.8.5 p. 365.

3. $ba^* + ab + (a + bb)ab^*$.

Solution :

Cf. livre Figure 15.9 p. 365.

Logique

Auteurs

The famous LI214 team - MJ/BB

Version du 27 juillet 2012

1 Fonctions booléennes

Exercice 1

1. Construire la table de vérité de la fonction booléenne correspondant au connecteur logique “ou exclusif”, noté XOR , et donner un polynôme booléen associé. Donner des formes disjonctives et conjonctives pour cette fonction.

Solution :

x	y	$XOR(x, y)$
0	0	0
0	1	1
1	0	1
1	1	0

$XOR(x, y) = \bar{x}y + x\bar{y}$. Ce polynôme correspond à une forme disjonctive.
Une forme conjonctive est : $XOR(x, y) = (x + y)(\bar{x} + \bar{y})$.

2. Donner un polynôme booléen pour la fonction f définie par :

x	y	$f(x, y)$
0	0	1
0	1	0
1	0	0
1	1	0

Solution :

$f(x, y) = \bar{x} \bar{y}$. Noter que ce polynôme correspond à la fois à une forme disjonctive et à une forme conjonctive. Ceci implique l'égalité : $f(x, y) = \bar{x} \bar{y} = (\bar{x} + y)(x + \bar{y})(\bar{x} + \bar{y})$.

3. Donner un polynôme booléen pour la fonction g définie par :

x_1	x_2	x_3	$g(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

et donner une forme conjonctive pour g .

Solution :

$g(x_1, x_2, x_3) = \bar{x}_1(\bar{x}_2x_3 + x_2\bar{x}_3 + x_2x_3) + x_1\bar{x}_2x_3$.

Une forme conjonctive : $g(x_1, x_2, x_3) = (x_1 + x_2 + x_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$.

4. Donner une forme conjonctive pour la fonction h définie par :

$$h(x, y, z) = xy + yz + xz.$$

Solution :

$$h(x, y, z) = (x + y)(y + z)(x + z).$$

Exercice 2

On rappelle que la *fonction caractéristique* d'une partie A de E est l'application $\mathbf{1}_A: E \longrightarrow \{0, 1\}$ définie par :

$$\mathbf{1}_A(e) = \begin{cases} 1 & \text{si } e \in A, \\ 0 & \text{sinon} \end{cases}$$

1. Exprimer en termes de fonction caractéristique les opérations d'union, intersection, complémentation, différence, différence symétrique.

Solution :

Si $a = \mathbf{1}_A$, $b = \mathbf{1}_B$, ce sont respectivement : $a + b$, ab , $\bar{a} = 1 - a$, $a(1 - b)$, $a \oplus b = a\bar{b} + \bar{a}b = (a + b)(\bar{a} + \bar{b})$.

2. Si E est fini de cardinal n montrer que la fonction caractéristique d'une partie A de E peut être écrite comme un n -uplet de nombres 0 ou 1 ; en déduire un isomorphisme entre les algèbres de Boole $\mathcal{P}(E)$ et $\{0, 1\}^n$.

Solution :

De manière générale, toute application f d'un ensemble $E = \{e_1, \dots, e_n\}$ (de cardinal n) dans un ensemble F est décrite par le n -uplet $(f(e_1), \dots, f(e_n))$, ce qui fournit une bijection entre le produit cartésien $F^n = F \times \dots \times F$ et l'ensemble F^E des applications de E dans F .

Remarquer que ceci fournit une autre démonstration du fait que le cardinal de $\mathcal{P}(E)$ est 2^n lorsque E est de cardinal n .

2 Logique des propositions

2.1 Syntaxe

Exercice 3

Les formules logiques peuvent être écrites en notation infixe, préfixe ou postfixe. Dans ces deux dernières notations, les parenthèses ne sont pas nécessaires. Parmi les mots suivants, écrits en notation préfixe, reconnaître ceux qui sont des formules, les écrire en notation infixe et dessiner l'arbre syntaxique les représentant.

$$\rightarrow \neg p \vee qr \quad \rightarrow \neg pq \vee r \quad \rightarrow \neg \rightarrow \vee \wedge pqr s \vee st$$

$$\rightarrow \vee \neg pqr \quad \rightarrow \neg ps \vee qr$$

Solution :

$\rightarrow \neg p \vee qr$	$\neg p \rightarrow (q \vee r)$
$\rightarrow \neg pq \vee r$	n'est pas une formule
$\rightarrow \neg \rightarrow \vee \wedge pqr s \vee st$	$\neg(((p \wedge q) \vee r) \rightarrow s) \rightarrow (s \vee t)$
$\rightarrow \vee \neg pqr$	$(\neg p \vee q) \rightarrow r$
$\rightarrow \neg ps \vee qr$	n'est pas une formule

2.2 Sémantique

Exercice 4

On définit l'opérateur booléen $NAND$ comme suit :

$$x \text{ NAND } y = \neg(x \wedge y)$$

Montrer que les connecteurs logiques \neg , \wedge , \vee et \rightarrow peuvent se définir en utilisant uniquement le connecteur $NAND$. Comment s'énonce cette propriété pour les fonctions booléennes ?

Solution :

Il suffit de montrer la propriété pour \neg et \wedge , puisque $x \vee y = \neg(\neg x \wedge \neg y)$ et $x \rightarrow y = y \vee \neg x$.

On a : $\neg x = x \text{ NAND } x$ et $x \wedge y = \neg(x \text{ NAND } y)$, c'est-à-dire $x \wedge y = (x \text{ NAND } y) \text{ NAND } (x \text{ NAND } y)$.

On en déduit que toute fonction booléenne peut s'exprimer uniquement en utilisant la fonction $NAND$ définie par la table de vérité :

x	y	$NAND(x, y)$
0	0	1
0	1	1
1	0	1
1	1	0

Exercice 5

Trouver une formule logique F contenant les trois symboles propositionnels p , q et r telle que F soit vraie uniquement lorsque p et q sont vrais et r est faux.

Solution :

La question est formulée avec du vocabulaire courant. Dans les termes du cours, on dirait que la formule F est telle que, pour toute interprétation I , $I(F) = 1$ ssi $I(p) = I(q) = 1$ et $I(r) = 0$.

$$F = p \wedge q \wedge \neg r$$

Exercice 6

Quel est le nombre maximum de formules logiques non équivalentes que l'on peut former avec n variables propositionnelles p_1, \dots, p_n distinctes ?

Solution :

C'est le nombre de fonctions booléennes à n arguments, c'est-à-dire le nombre d'applications de $\{0, 1\}^n$ dans $\{0, 1\}$, donc 2^{2^n} .

En effet (cf. cours), remarquer la bijection qui à la formule F associe la fonction $g_F : \{0, 1\}^n \rightarrow \{0, 1\}$ donnée par : $g_F(x_1, \dots, x_n)$ est la valeur de vérité de F pour l'interprétation x_1, \dots, x_n de p_1, \dots, p_n .

Autrement dit, pour l'interprétation I de p_1, \dots, p_n telle que $I(p_i) = x_i$, notée $I = (x_1, \dots, x_n)$, on a : $g_F(I) = I(F)$.

Exercice 7

Mettre en forme normale disjonctive et conjonctive les formules suivantes :

$$\neg(p \wedge \neg(q \vee s)) \quad \neg(p \wedge (q \vee s)) \quad \neg(p \vee (q \wedge \neg s)) \wedge s$$

Solution :

De manière analogue aux fonctions booléennes, étant donnée une formule F , on cherche une formule F' telle que $F \sim F'$ avec F' sous forme disjonctive (c'est-à-dire une disjonction de conjonctions) et une formule F'' telle que $F \sim F''$ et F'' sous forme conjonctive (c'est-à-dire une conjonction de disjonctions).

Le plus simple est de passer par les fonctions booléennes.

Exercice 8

On considère la formule F suivante :

$$(p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r)$$

1. Les formules F et $\neg F$ sont-elles satisfaisables ? Sont-elles valides (des tautologies) ? Donner des formes conjonctives et disjonctives pour ces formules.

Solution :

Remarquer que la formule F correspond à “exactement deux vrais parmi trois”, elle est associée à la fonction booléenne : $f(x, y, z) = xy\bar{z} + x\bar{y}z + \bar{x}yz$. Elle est donc satisfaisable mais non valide et il en est de même de $\neg F$ qui est associée à \bar{f} . On obtient facilement une forme conjonctive pour \bar{f} : $\bar{f}(x, y, z) = (\bar{x} + \bar{y} + z)(\bar{x} + y + \bar{z})(x + \bar{y} + \bar{z})$. On peut en déduire une forme disjonctive de $\neg F$ en développant puis une forme conjonctive pour F par négation.

2. Déterminer une formule G telle que $(F \wedge G) \vee (\neg F \wedge \neg G)$ soit une tautologie.

Solution :

Comme $F \vee \neg F$ est valide, il suffit de prendre $G = F$.

Exercice 9

On considère l'ensemble de formules suivant :

$$E = \{\neg\neg p, p \rightarrow q, \neg q \vee r, s \rightarrow r, \neg(s \vee t)\}$$

Parmi les formules suivantes, lesquelles sont conséquences de E . On choisira une seule réponse parmi les possibilités qui suivent :

1. $\neg p$
2. $\neg q$
3. q
4. t

Solution :

C'est q . En effet, on peut vérifier que $\{p, p \rightarrow q\} \models q$, donc aussi $\{\neg\neg p, p \rightarrow q\} \models q$: soit I une interprétation telle que $I(p) = 1$ et $I(p \rightarrow q) = 1$. Comme $I(p \rightarrow q) = I(q) + \overline{I(p)} = 1$ et $\overline{I(p)} = 0$, on en déduit $I(q) = 1$. Ainsi q est bien conséquence de $\{p, p \rightarrow q\}$, donc $E \models q$.

Exercice 10

Anna et Mathias sont accusés d'un crime. Il font les déclarations suivantes :

Anna : Mathias est coupable.

Mathias : Nous sommes tous les deux innocents.

1. On suppose que tous les deux ont menti. Peut-on déterminer qui est coupable, qui ne l'est pas ?
2. On suppose maintenant que les coupables mentent et que les innocents disent la vérité. Peut-on déterminer qui est coupable, qui ne l'est pas ?

Solution :

On note p : “Mathias est innocent” et q : “Anna est innocente”.

1. Anna a menti donc p est vrai, et Mathias a menti donc $\neg(p \wedge q)$ est vrai. Comme $\{p, \neg(p \wedge q)\} \models p \wedge \neg q$, on en déduit que p et $\neg q$ sont vrais, Mathias est donc innocent et Anna coupable.

2. Pour chaque déclaration on distingue le cas où l'auteur de la déclaration est coupable du cas où il est innocent. On obtient alors l'ensemble de formules $E = \{A, B, C, D\}$ avec :

$$A : q \rightarrow \neg p \quad B : \neg q \rightarrow p \quad C : p \rightarrow (p \wedge q) \quad D : \neg p \rightarrow \neg(p \wedge q)$$

Remarquons d'abord que D est valide.

En effet, pour toute interprétation I , $I(D) = \overline{I(p \wedge q)} + I(p) = \overline{I(p)} + \overline{I(q)} + I(p) = 1$.

Soit maintenant I une interprétation satisfaisant E . Elle vérifie les trois équations :

$$\begin{cases} I(A) = \overline{I(p)} + \overline{I(q)} = 1 & (1) \\ I(B) = I(p) + I(q) = 1 & (2) \\ I(C) = \overline{I(p)} + I(p)I(q) = 1 & (3) \end{cases}$$

Supposons que Mathias est innocent, c'est-à-dire $I(p) = 1$: on a alors $\overline{I(p)} = 0$ et $I(q) = 1$ d'après (3) et $\overline{I(q)} = 1$ donc $I(q) = 0$ d'après (1). C'est donc impossible. Si Mathias est coupable, c'est-à-dire $I(p) = 0$, on obtient $I(q) = 1$ et cette interprétation satisfait bien E donc on conclut que Mathias est coupable et Anna innocente.

Autre méthode. Ici, il est plus rapide de considérer la table de vérité :

p	q	A	B	C	D
0	0	1	0	1	1
0	1	1	1	1	1
1	0	1	1	0	1
1	1	0	1	1	1

dans laquelle on constate qu'une seule interprétation I rend vraies les formules A, B, C (et D). Cette interprétation nous permet de déduire que Mathias est coupable et Anna est innocente.

Exercice 11

Les conséquences suivantes sont-elles vérifiées ?

- (1) $p \rightarrow q \models \neg q \rightarrow \neg p$
- (2) $p \rightarrow q \models q \rightarrow p$
- (3) $p \rightarrow q \models \neg p \rightarrow \neg q$
- (4) $\{p \vee q, \neg p\} \models q$
- (5) $\{p \vee q, p \rightarrow r, q \rightarrow r\} \models r$
- (6) $\{\neg(p \vee q), r \rightarrow q\} \models \neg(p \vee r)$

Solution :

On raisonne comme dans l'exercice précédent. Par exemple pour (5), on se donne une interprétation I qui satisfait $I(p \vee q) = 1$, $I(p \rightarrow r) = 1$ et $I(q \rightarrow r) = 1$, donc $I(p) + I(q) = 1$, $I(r) + \overline{I(p)} = 1$ et $I(r) + \overline{I(q)} = 1$. Par l'absurde, si $I(r)$ était égal à 0, on aurait $\overline{I(p)} = \overline{I(q)} = 1$, donc $I(p) = I(q) = 0$, ce qui contredit $I(p) + I(q) = 1$. Donc $I(r) = 1$ et la conséquence est vérifiée.

De même, (1), (4) et (6) sont vérifiées, mais pas (2) ni (3) : un contre exemple est obtenu dans les deux cas pour l'interprétation I telle que $I(p) = 0$ et $I(q) = 1$.

Exercice 12

Soit φ et ψ deux formules de la logique des propositions. Montrer que : $\varphi \models \psi$ si et seulement si $\varphi \rightarrow \psi$ est une tautologie.

Solution :

Cf. cours. Le plus simple est de montrer l'équivalence des négations.

Exercice 13

Les formules suivantes sont-elles satisfaisables ? sont-elles valides (des tautologies) ?

- (1) $(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)$
- (2) $((p \vee q) \wedge \neg p) \rightarrow q$
- (3) $(p \rightarrow q) \rightarrow (q \rightarrow p)$
- (4) $(p \rightarrow q) \rightarrow (\neg p \rightarrow \neg q)$
- (5) $(p \rightarrow q) \rightarrow p$
- (6) $p \rightarrow (q \rightarrow p)$

Solution :

On peut dans certains cas utiliser les deux exercices qui précèdent. Par exemple pour (1), on sait par 11.(2) que $p \rightarrow q \models \neg q \rightarrow \neg p$ donc (1) est valide.

On peut aussi bricoler à la main, par exemple pour (6) : soit I une interprétation, alors $I(p \rightarrow (q \rightarrow p)) = I((q \rightarrow p) + \overline{I(p)}) = I(p) + \overline{I(q)} + \overline{I(p)} = 1$ donc la formule est valide.

Exercice 14

Un logicien dit à son fils : “si tu ne ranges pas ta chambre, tu n’iras pas au cours de logique”; le fils range sa chambre, et est envoyé au cours de gymnastique tout de suite après. Quelle erreur avait-il faite en pensant aller au cours de logique ?

Solution :

Il pensait que $\neg p \rightarrow \neg q \models p \rightarrow q$ ou encore que $(\neg p \rightarrow \neg q) \rightarrow (p \rightarrow q)$ est valide ce qui est faux d’après les exercices précédents.

Exercice 15

Formaliser par un ensemble E de formules de la logique des propositions le texte : “Si Roméo aime Juliette, elle l’épousera. Si Roméo n’aime pas Juliette, elle épousera Gaston. Or Juliette n’épouse pas Roméo.”. Si p est une variable propositionnelle désignant l’assertion “Juliette épouse Gaston.”, a-t-on $E \models p$? $E \models \neg p$?

Solution :

$E \models p$: même raisonnement que précédemment, avec $E = \{q \rightarrow r, \neg q \rightarrow p, \neg r\}$ où q représente “ R aime J ” et r : “ J épouse R ”.

Exercice 16

Anissa, Antoine et Jennifer ont un examen de logique à passer. On suppose que :

- A : L’un des trois au moins révisera pour l’examen.
- B : Si Anissa ne révise pas, alors Antoine non plus.
- C : Si Anissa révise, alors Jennifer aussi.

Formaliser ces trois hypothèses. Peut-on dire qui révisera ? qui ne révisera pas ?

Solution :

En notant p : “Anissa révise”, q : “Antoine révise” et r : “Jennifer révise”, on obtient $E = \{A, B, C\}$ avec :

$A : p \vee q \vee r$, $B : \neg p \rightarrow \neg q$ et $C : p \rightarrow r$.

Une interprétation I satisfaisant E vérifie donc les équations :

$$\begin{cases} I(A) = I(p) + I(q) + I(r) = 1 & (1) \\ I(B) = \overline{I(q)} + I(p) = 1 & (2) \\ I(C) = I(r) + \overline{I(p)} = 1 & (3) \end{cases}$$

Supposons $I(r) = 0$ alors $\overline{I(p)} = 1$ d’après (3) donc $I(p) = 0$ et donc $\overline{I(q)} = 1$ d’après (2) et $I(q) = 1$ d’après (3), ce qui aboutit à une contradiction. Pour $I(r) = 1$, on constate que, pour $I(p) = 1$ les équations sont satisfaites avec $I(q)$ quelconque, et pour $I(p) = 0$, les équations sont satisfaites pour $I(q) = 0$. On ne peut donc rien déduire pour Anissa et Antoine, mais on sait que Jennifer révisera.

Autre méthode. Ici, l’utilisation d’une table de vérité est plus fastidieuse :

p	q	r	A	B	C	
0	0	0	0	1	1	
0	0	1	1	1	1	I_1
0	1	0	1	0	1	
0	1	1	1	0	1	
1	0	0	1	1	0	
1	0	1	1	1	1	I_2
1	1	0	1	1	0	
1	1	1	1	1	1	I_3

on constate qu'il existe seulement 3 valuations I_1 , I_2 et I_3 qui rendent vraies les formules A, B et C. On a $I_1(p) = 0$ et $I_2(p) = 1$ donc on ne peut rien déduire sur Anissa. On a $I_2(q) = 0$ et $I_3(q) = 1$ donc on ne peut rien déduire sur Antoine. On a $I_1(r) = I_2(r) = I_3(r) = 1$ donc on peut en déduire que Jennifer révisera.

Exercice 17

Étant donnés deux ensembles \mathcal{F} et \mathcal{G} de formules logiques, on dit que \mathcal{G} est conséquence de \mathcal{F} , ce que l'on note $\mathcal{F} \models \mathcal{G}$ si et seulement si pour toute interprétation I , on a :

si $I(F) = 1$ pour toute formule $F \in \mathcal{F}$
alors $I(G) = 1$ pour toute formule $G \in \mathcal{G}$

Soient trois ensembles \mathcal{F} , \mathcal{F}' et \mathcal{F}'' de formules, montrer que si $\mathcal{F} \models \mathcal{F}'$ et $\mathcal{F}' \models \mathcal{F}''$ alors $\mathcal{F} \models \mathcal{F}''$.

Solution :

Montrons que $\mathcal{F} \models \mathcal{F}''$. Soit I une interprétation, supposons que pour toute formule $F \in \mathcal{F}$, $I(F) = 1$ et montrons que pour toute formule $F'' \in \mathcal{F}''$, $I(F'') = 1$. Puisque, par hypothèse, $\mathcal{F} \models \mathcal{F}'$, et puisque pour toute formule $F \in \mathcal{F}$, $I(F) = 1$, on sait que pour toute formule $F' \in \mathcal{F}'$, $I(F') = 1$, et donc, puisque, toujours par hypothèse, $\mathcal{F}' \models \mathcal{F}''$, on en déduit que pour toute formule $F'' \in \mathcal{F}''$, $I(F'') = 1$, ce qui permet de conclure.

Exercice 18

1. Montrer que quelles que soient les formules φ , φ_1 , φ_2 , ..., φ_n et l'interprétation I :

$$I(\varphi_n \rightarrow (\varphi_{n-1} \rightarrow (\cdots (\varphi_1 \rightarrow \varphi) \cdots))) = I((\varphi_n \wedge (\varphi_{n-1} \wedge (\cdots \wedge (\varphi_2 \wedge \varphi_1) \cdots))) \rightarrow \varphi)$$

2. À tout séquent $s = (E, \varphi)$, on associe la formule $f(s)$ définie comme suit :

- si $E = \emptyset$ alors $f(s) = \varphi$
- si $E = \{\psi\} \cup E'$ et $f((E', \varphi)) = \psi'$ alors $f(s) = \psi \rightarrow \psi'$

Montrer qu'un séquent s est valide si et seulement si, pour toute interprétation I , $I(f(s)) = 1$.

Exercice 19

Le raisonnement suivant est-il correct ?

Si Antoine a lu l'Iliade, Mathias aussi. Si Antoine a lu l'Odyssée, Mathias aussi. On sait qu'aucun de ces deux livres n'a été lu à la fois par Antoine et par Mathias, et qu'aucune de ces deux personnes n'a lu à la fois l'Iliade et l'Odyssée. Par conséquent, au moins l'un des deux livres n'a été lu ni par Antoine, ni par Mathias.

3 Logique du premier ordre

3.1 Syntaxe

Rappel : L'ensemble des formules de la logique du premier ordre est défini inductivement par :

- (B) Si R est un symbole de prédicat à n arguments, et si t_1, \dots, t_n sont des termes, alors $R(t_1, \dots, t_n)$ est une formule.
- (I) Si F et F' sont des formules, alors $\neg F$, $(F \rightarrow F')$, $(F \wedge F')$ et $(F \vee F')$ sont des formules.
Si F est une formule et x est une variable, alors $\forall x F$ et $\exists x F$ sont des formules.

Exercice 20

Déterminer l'ensemble des variables libres et l'ensemble des variables liées pour chacune des formules suivantes :

$$\begin{aligned} F : & p(f(x, y)) \vee \forall z r(a, z) \\ G : & \forall x s(x, y, z) \vee \forall z (p(z) \rightarrow q(z)) \\ H : & \forall x \exists y (r(x, y) \rightarrow \forall z s(x, y, z)) \end{aligned}$$

Quelles sont les formules closes parmi F , G et H ?

Solution :

$$L(F) = \{x, y\}, B(F) = \{z\}$$

$$L(G) = \{y, z\}, B(G) = \{x\}$$

$$L(H) = \emptyset, B(H) = \{x, y, z\}$$

Donc seule H est close (aucune variable libre).

Exercice 21

Donner une définition inductive de l'ensemble des variables libres d'une formule.

Solution :

Cf. cours.

Exercice 22

- Déterminer l'arbre syntaxique représentant la formule $\neg((\forall x P(x)) \wedge Q(x)) \rightarrow (R(x) \vee \neg P(x))$. Ecrire cette formule en notation préfixe (c'est-à-dire faisant figurer les opérateurs \wedge, \rightarrow , etc. avant les arguments sur lesquels ils s'appliquent).

Solution :

Notation préfixe (on peut aussi enlever les grosses parenthèses) : $\neg \rightarrow \left(\wedge (\forall x P(x)) Q(x) \right) \vee \left(R(x) \neg P(x) \right)$

- Donner une définition inductive de la fonction qui étant donnée une formule de la logique du premier ordre en notation infixe construit la même formule en notation préfixe.

Solution :

Soit $pref$ cette fonction :

(B) $pref(A) = A$ si A est une formule atomique,

(I) $pref(\neg F) = \neg pref(F)$,

$pref(F * G) = * pref(F) pref(G)$, pour $*$ $\in \{\vee, \wedge, \rightarrow\}$

$pref(QxF) = Qx pref(F)$, pour $Q \in \{\exists, \forall\}$

3.2 Sémantique**Exercice 23**

Préciser à l'aide d'un quantificateur le sens de "un" dans les phrases suivantes et formaliser ces phrase en logique des prédicats :

- Jean suit un cours.
- Un logicien a été champion du monde de cyclisme.
- Un entier naturel est pair ou impair.
- Un enseignant-chercheur a toujours un nouveau sujet à étudier.
- Dans un triangle isocèle une médiane est également hauteur.
- Dans un triangle équilatéral une médiane est également hauteur.
- Un étudiant a besoin d'avoir un idéal.

Solution :

1. \exists 2. \exists 3. \forall 4. \forall, \exists 5. \forall, \exists 6. \forall, \forall 7. \forall, \exists .

Par exemple pour 7.

$\forall x (Etudiant(x) \rightarrow \exists y (Ideal(y) \wedge R(x, y)))$ où le prédicat binaire R est défini par $R(x, y)$ si x a pour idéal y .

Exercice 24

Formaliser les énoncés suivants dans le langage de la logique du premier ordre :

1. Tous les étudiants sont doués de raison.
2. Seuls les êtres humains sont doués de raison.
3. Aucun éléphant n'est doué de raison.
4. Tous les poissons, sauf les requins, sont gentils avec les enfants.
5. Chacun cherche son éléphant.
6. Chaque individu aime quelqu'un et personne n'aime tout le monde.
7. Tout homme a des amis et des ennemis.
8. N'importe qui peut apprendre la logique, s'il travaille assez.

Exercice 25

Formaliser en logique des prédicats l'énoncé : "il existe un unique x tel que $p(x)$ " (on n'utilisera pas la notation $\exists!$ mais seulement le langage de la logique des prédicats vu en cours).

Exercice 26

On considère un langage avec la constante 1, la fonction unaire s (successeur), les prédicats unaires "premier" et "pair", et le prédicat binaire "égalité". Traduire en formules logiques les énoncés :

1. Tout nombre pair est premier.
2. Aucun nombre pair n'est premier.
3. Certains nombres premiers sont pairs.
4. Certains nombres premiers ne sont pas pairs.
5. Tout nombre premier est soit pair soit égal à 2.

Donner parmi ces formules celles qui sont valides dans \mathbb{N} .

Exercice 27

On considère un langage avec deux prédicats binaires : R et l'égalité.

- Donner une formule exprimant que la relation R est une relation d'ordre large.
- Donner une formule exprimant que l'ordre R a un plus petit élément.
- Donner une formule exprimant que l'ordre R a un élément minimal.

Exercice 28

1. On considère la phrase : *Tout nombre entier naturel x a un successeur qui est inférieur ou égal à tout entier strictement supérieur à x .* Représenter cette phrase par une formule logique en utilisant les prédicats suivants (ainsi que l'égalité) :

$\text{entier}(x)$	" x est un entier naturel"
$\text{succ}(x, y)$	" x est successeur de y "
$\text{inf}(x, y)$	" x est inférieur ou égal à y "

Solution :

$$\forall x(\text{entier}(x) \rightarrow \exists y(\text{succ}(y, x) \wedge \forall z(\text{inf}(x, z) \wedge \neg(x = z) \rightarrow \text{inf}(y, z))))$$

2. La formule $\forall x(p(x) \rightarrow \exists y(q(y, x) \wedge \forall z(r(z, x) \rightarrow s(z, y))))$ est-elle satisfaisable ? valide ?

Solution :

La formule est satisfaisable d'après ce qui précède, avec $\text{entier}(x)$ pour $p(x)$, $\text{succ}(x, y)$ pour $q(x, y)$, x est strictement inférieur à y pour $r(x, y)$ et $\text{inf}(x, y)$ pour $s(x, y)$.

Elle n'est pas valide, par exemple en prenant $\text{inf}(x, y)$ pour $r(x, y)$.

Une autre interprétation falsifiante est obtenue avec :

$\text{ville}(x)$	“ x est une ville de France” pour $p(x)$,
$\text{maire}(x, y)$	“ x est maire de y ” pour $q(x, y)$,
$\text{chien}(x, y)$	“ x est un chien de (la ville) y ” pour $r(x, y)$
$\text{a-mordu}(x, y)$	“ x a mordu y pour $s(x, y)$.”

qui donne l'énoncé : *toute ville de France a un maire qui a été mordu par tous les chiens de la ville.*

Exercice 29

On considère une structure de domaine $\mathcal{D} = \mathbb{N}$ et $X = \{x, y, z\}$.

Déterminer $v_1 = v[z \rightarrow 5]$, $v_2 = v[y \rightarrow 1]$ et $v_3 = v_1[x \rightarrow 3]$ pour la valuation v définie par :

$$v: \begin{cases} x \rightarrow 4 \\ y \rightarrow 12 \\ z \rightarrow 7 \end{cases}$$

Exercice 30

Dans une structure \mathcal{M} de domaine \mathbb{N} , munie de la multiplication et de l'égalité, on considère le prédicat p défini par :

$$p(x) \text{ si } x = x^2$$

Donner la valeur des formules suivantes :

1. $p(0)$ (qui correspond à $\hat{v}(p(x))$ pour la valuation v telle que $v(x) = 0$).
2. $p(1)$
3. $p(2)$
4. $\exists x p(x)$
5. $\forall x p(x)$

Exercice 31

Dans une structure \mathcal{M} de domaine \mathbb{N} , munie de l'addition, de la soustraction et de l'égalité, on considère le prédicat p défini par :

$$p(x, y) \text{ si } x + y = x - y$$

Les formules suivantes sont-elles valides dans \mathcal{M} ?

1. $p(1, 1)$ (qui correspond à $\hat{v}(p(x, y))$ pour la valuation v telle que $v(x) = v(y) = 1$).
2. $p(2, 0)$
3. $\forall y p(1, y)$
4. $\exists x p(x, 2)$
5. $\exists x \exists y p(x, y)$
6. $\forall x \exists y p(x, y)$
7. $\exists x \forall y p(x, y)$
8. $\forall y \exists x p(x, y)$
9. $\forall x \forall y p(x, y)$

Exercice 32

Soient les formules logiques :

1. $\forall x (\neg R(x, x) \wedge \forall y (R(x, y) \rightarrow \neg R(y, x)) \wedge \forall z ((R(x, y) \wedge R(y, z)) \rightarrow R(x, z)))$
2. $\exists x \forall y R(x, y)$
3. $\forall x \exists y R(x, y)$
4. $\exists x \forall y R(y, x)$
5. $\forall x \exists y R(y, x)$

$$6. \forall x \exists y (R(x, y) \wedge \forall z (R(x, z) \rightarrow (z = y \vee R(y, z))))$$

$$7. \forall x \forall y (R(x, y) \rightarrow \exists z (R(x, z) \wedge R(z, y)))$$

Quelles sont parmi ces formules celles qui sont valides dans les structures ci-dessous :

1. les entiers naturels (positifs) comme domaine et $<$ pour R .
2. les entiers relatifs comme domaine et $<$ pour R .
3. les rationnels comme domaine et $<$ pour R .
4. l'ensemble des parties de \mathbb{N} comme domaine et \subset (l'inclusion stricte) pour R .
5. Mêmes questions avec des ordres larges à la place des ordres stricts.

Exercice 33

Pouvez-vous trouver une structure dans laquelle la formule $\exists x A \rightarrow \forall x A$ est fausse ? est vraie ?

Exercice 34

Pouvez-vous trouver une structure dans laquelle la formule $\forall y \exists x F \rightarrow \exists x \forall y F$ est fausse ? est vraie ?

Exercice 35

Les formules :

$$(1) (\exists x F \wedge \exists x F') \rightarrow \exists x (F \wedge F')$$

$$(2) \exists x (F \wedge F') \rightarrow (\exists x F \wedge \exists x F')$$

$$(3) (\forall x F \vee \forall x F') \rightarrow \forall x (F \vee F')$$

$$(4) \forall x (F \vee F') \rightarrow (\forall x F \vee \forall x F')$$

$$(5) \forall x A \vee \exists x \neg A$$

$$(6) \exists x \forall y F \rightarrow \forall y \exists x F$$

sont-elles universellement valides ? Lorsque ce n'est pas le cas, trouver deux formules F et F' et définir une structure qui permette de falsifier la formule.

Exercice 36

Pour chacune des formules ci-dessous, pouvez-vous trouver une structure qui rend vraie la formule ? qui rend fausse la formule ?

$$(1) \exists x p(x, x) \rightarrow \exists y \exists z p(y, z)$$

$$(2) \exists x \forall y p(x, y) \rightarrow \forall y \exists x p(x, y)$$

$$(3) \forall x p(x, x) \rightarrow \forall y \forall z p(y, z)$$

$$(4) \forall y \exists x p(x, y) \rightarrow \exists x \forall y p(x, y)$$

Exercice 37

Formaliser en logique du premier ordre chacune des phrases du texte suivant emprunté à Lewis Carroll :

Aucune des choses que quelqu'un rencontre en mer et qu'il ne remarque pas n'est une sirène. Les choses qui sont rencontrées en mer et qu'on inscrit sur le livre de bord sont toujours dignes d'intérêt. Personnellement, je n'ai jamais rien rencontré, au cours de mes voyages en mer, qui soit digne d'intérêt. Les choses que quelqu'un rencontre en mer et qu'il remarque sont toujours inscrites sur le livre de bord.

Que peut-on dire des choses rencontrées en mer ? Formaliser votre réponse.

3.3 Le monde de Tarski

Exercice 38

On considère les formules logiques suivantes :

- (1) $F_1: \forall x \forall y (LeftOf(x, y) \rightarrow RightOf(y, x))$
- (2) $F_2: \forall x \forall y ((Small(x) \wedge Small(y) \wedge BackOf(x, y)) \rightarrow Dodec(x))$
- (3) $F_3: \forall x \forall y ((Cube(x) \wedge Cube(y) \wedge x \neq y) \rightarrow (Larger(x, y) \vee Smaller(x, y)))$
- (4) $F_4: \forall x \forall y \forall z ((Smaller(x, y) \wedge Smaller(y, z)) \rightarrow Smaller(x, z))$
- (5) $F_5: \forall x \forall y (Larger(x, y) \rightarrow x \neq y)$
- (6) $F_6: \forall x \forall y \forall z (Between(x, y, z) \vee \neg Between(x, y, z))$
- (7) $F_7: \forall x \forall y \forall z (Between(x, y, z) \vee \neg Between(x, z, y))$

1. Quelles sont les formules qui sont valides dans tous les mondes de Tarski ? Pour chaque formule qui n'est pas valide dans tous les mondes de Tarski, expliquer comment construire un monde qui falsifie la formule.
2. Quelle formule est une tautologie ?
3. Pour chaque formule qui est valide dans tous les mondes de Tarski mais qui n'est pas une tautologie, expliquer comment changer l'interprétation des prédicats pour falsifier la formule.

Exercice 39 – Un monde de Tarski infini.

On représente la grille du monde de Tarski par l'ensemble $\mathbb{N} \times \mathbb{N}$ des coordonnées ligne et colonne. La case $(0, 0)$ est en haut à gauche. Une figure est représentée par son nombre de face (4, 6 ou 12) et sa taille par l'entier 0, 1 ou 2.

1. Donner une définition d'un ensemble T permettant de représenter une figure dans le monde de Tarski.
2. Proposer une interprétation des prédicats Tet, Cube, Dodec, Small, Medium, Large, Smaller, Larger, LeftOf, RightOf, BackOf, FrontOf et Between dans cette représentation.
3. En utilisant les symboles de prédicats du monde de Tarski, et uniquement ceux-ci, définir les formules suivantes :
 1. $LessFig[x, y]$ pour “ x est une figure plus petite que y ” (fonction du nombre de faces).
 2. $EqFig[x, y]$ pour “ x et y sont la même figure”.
 3. $EqSize[x, y]$ pour “ x est de la même taille que y ”.
 4. $\Phi[x, y]$ qui donne l'ordre lexicographique (strict) sur l'ensemble T (on utilisera les définitions ci-dessus, et le nombre de faces est prioritaire par rapport à la taille, c'est-à-dire qu'un petit dodécaèdre est supérieur à un grand carré par exemple).

Pour aller plus loin...

Auteurs

IG, transformé en L^AT_EX par VMM

Version du 27 juillet 2012

1 Ensembles et fonctions

Exercice 1 – Niveau *

Soit A un ensemble infini. Soit $B = \{b_1, b_2, \dots\}$ un ensemble dénombrable tel que A et B sont disjoints. Démontrer qu'il existe une bijection entre $A \cup B$ et A .

Solution :

Comme A est infini il contient un sous-ensemble dénombrable d'éléments $D = \{d_1, d_2, \dots\}$. Soit $f : A \cup B \rightarrow A$ défini comme suit :

$$f(x) = \begin{cases} x & \text{si } x \in A \setminus D \\ d_{2n-1} & \text{si } x = d_n \\ d_{2n} & \text{si } x = b_n \end{cases}$$

Observons que f est une fonction bijective.

2 Relations d'ordre

Exercice 2

Soit \mathcal{A} un ensemble d'ensembles. \mathcal{A} est partiellement ordonné par inclusion et soit \mathcal{B} un sous-ensemble de \mathcal{A} .

1. Démontrer que si $A \in \mathcal{A}$ est un majorant de \mathcal{B} , alors

$$\bigcup \{B : B \in \mathcal{B}\} \subset A$$

Solution :

Soit $x \in \bigcup \{B : B \in \mathcal{B}\}$; alors $\exists B_0 \in \mathcal{B}$ t.q. $x \in B_0$. Mais A est un majorant de \mathcal{B} ; donc $B_0 \subset A$ et en particulier $x \in A$. En conséquence, $\bigcup \{B : B \in \mathcal{B}\} \subset A$.

2. Est-ce que $\bigcup \{B : B \in \mathcal{B}\}$ est un majorant de \mathcal{B} ?

Solution :

Même si \mathcal{B} est une sous-collection de \mathcal{A} , il est possible que $\bigcup \{B : B \in \mathcal{B}\}$ ne soit pas un élément de \mathcal{A} . Cela signifie que $\bigcup \{B : B \in \mathcal{B}\}$ est un majorant de \mathcal{B} ssi il appartient à \mathcal{A} . *Exemple pr soutenir l'intuition* $\mathcal{A} = \{B_n | n > 1\}$ et B_n est l'ensemble des multiples de n , alors $B_6 \cup B_8$ n'est pas dans \mathcal{A} .

Exercice 3 – Niveau **

Soit X un ensemble partiellement ordonné. Alors il existe un sous-ensemble Y totalement ordonné de X t.q. Y n'est pas un sous-ensemble propre des autres sous-ensembles totalement ordonnés de X .

Solution :

Soit \mathcal{A} l'ensemble de tous les sous-ensembles totalement ordonnés de X . Soit \mathcal{A} partiellement ordonné par inclusion. Nous voulons montrer, par le lemme de Zorn, que \mathcal{A} contient un élément maximal. Supposons que $\mathcal{B} = \{B_i : i \in I\}$ est un sous-ensemble totalement ordonné de \mathcal{A} . Soit $A = \bigcup \{B_i : i \in I\}$.

Observons que si $B_i \subset X$ pour tout $B_i \in \mathcal{B}$, alors $A \subset X$.

Ensuite, nous voulons démontrer que A est totalement ordonné. Soit $a, b \in A$; alors $\exists B_j, B_k \in \mathcal{B}$ t.q. $a \in B_j$ et $b \in B_k$. Mais \mathcal{B} est totalement ordonné par inclusion, et donc l'un de ces ensembles, disons B_j , est un sous-ensemble de l'autre. Donc $a, b \in B_k$. Rappelons que $B_k \in \mathcal{B}$ est un sous-ensemble totalement ordonné de X par définition; alors soit $a < b$ soit $b < a$. Alors A est un sous-ensemble totalement ordonné de X , et en particulier $A \in \mathcal{A}$.

Mais $B_i \subset A$, pour tout $B_i \in \mathcal{B}$; donc A est une borne supérieure de \mathcal{B} . Comme chaque sous-ensemble totalement ordonné de \mathcal{A} a une borne supérieure dans \mathcal{A} , par le lemme de Zorn, \mathcal{A} possède un élément maximale, c'est-à-dire un sous-ensemble totalement ordonné de X qui n'est pas un sous-ensemble propre des autres sous-ensembles totalement ordonnés de X .

Exercice 4 – Niveau *

Soit R une relation, $R \subset A \times B$. Supposons que le domaine de R soit A . Alors il existe un sous-ensemble f^* de R tel que f^* est une fonction de A vers B .

Solution :

Soit \mathcal{A} un ensemble de sous-ensemble de R tel que $f \in \mathcal{A}$ est une fonction de A à B . \mathcal{A} est partiellement ordonné par inclusion. Observons que si $f : A_1 \rightarrow B$ est un sous-ensemble de $g : A_2 \rightarrow B$ alors $A_1 \subset A_2$.

Supposons $\mathcal{B} = \{f_i : A_i \rightarrow B\}_{i \in I}$ est un sous-ensemble de \mathcal{A} totalement ordonné. Alors $f = \bigcup_i f_i$ est une fonction de $\bigcup_i A_i \rightarrow B$. De plus, $f \subset R$. Donc f est une borne supérieure de \mathcal{B} . Par le lemme de Zorn, \mathcal{A} possède un élément maximale $f^* : A^* \rightarrow B$. SI on démontre que $A^* = A$, alors l'énoncé est prouvé.

Supposons que $A^* \neq A$. Alors $\exists a \in A$ t.q. $a \notin A^*$. Par hypothèse, le domaine de R est A ; donc il existe une paire ordonnée $(a, b) \in R$. Alors $f^* \cup \{(a, b)\}$ est une fonction de $A^* \cup \{a\}$ dans B . Nous avons une contradiction avec le fait que f^* est un élément maximale de \mathcal{A} . On conclut que $A^* = A$, et l'énoncé est prouvé.

3 Induction structurelle**Exercice 5**

Un arbre n -aire *complet* est un arbre où :

- chaque nœud interne a exactement n fils,
- toutes les feuilles sont à la même profondeur.

L'arbre vide est un arbre n -aire de profondeur 0.

1. Donner une définition inductive des arbres n -aires complets et étiquetés sur l'alphabet $A = \{a\}$ (toutes les noeuds sont étiquetés a).

Solution :

$\emptyset \in AC$ et si $t \in AC$ alors $(a, \underbrace{t, \dots, t}_n) \in A$.

2. Donner une définition inductive des arbres n -aires complets et étiquetés sur un alphabet A non réduit à un élément.
3. Donner une définition inductive du nombre de nœuds et le nombre d'arêtes d'un arbre n -aire complet de profondeur k .

Solution :

$n_k = nn_{k-1} + 1, n_0 = 0$ et $a_k = na_{k-1} + n, a_0 = 0$ (le calcul est à faire plus tard).

Exercice 6

Les listes de lettres L de l'alphabet A sont définies inductivement par :

(B) $\varepsilon \in L$,

(I) $\forall l \in L, \forall a \in A, (al) \in L$.

Définissons $g(x, y)$ sur $L \times L$ par : $\forall a \in A, \forall l \in L, \forall y \in L$,

$$\begin{aligned} g(\varepsilon, y) &= y, \\ g(al, y) &= g(l, (ay)). \end{aligned}$$

1. Soit $Q(x)$ le prédicat ' $\forall y, g(x, y)$ est défini'. Prouver par induction sur x que $Q(x)$ est vrai sur L .

Solution :

$Q(\varepsilon)$ vrai et $Q(x)$ vrai impliquent que $Q(ax)$ est vrai.

2. Calculer $g((a_1), y)$, pour $a_1 \in A, y \in L$.

Solution :

$$g((a_1), y) = (a_1 y).$$

3. Prouver par induction sur n (pour $n \geq 1$) que

$$g((a_n(a_{n-1}(\dots(a_1)\dots))), y) = g(\varepsilon, (a_1(\dots(a_{n-1}(a_n y))\dots))).$$

Solution :

Vérifions par induction que

$$g((a_n(a_{n-1}(\dots(a_1)\dots))), y) = g(\varepsilon, (a_1(\dots(a_{n-1}(a_n y))\dots))).$$

C'est clair pour $n = 0$. Supposons que l'égalité est vraie pour n , alors

$$\begin{aligned} g((a_{n+1}(a_n(\dots(a_2(a_1))\dots))), y) &= g((a_n(\dots(a_2(a_1))\dots)), (a_{n+1}y)) \\ &= g(\varepsilon, (a_1(a_2(\dots(a_n(a_{n+1}y))\dots))))). \end{aligned}$$

4. Soit $rev(x) = g(x, \varepsilon)$.

Déduire de la question 3 que, pour $a_1, \dots, a_n \in A$,

$$rev((a_n(a_{n-1}(\dots(a_1)\dots)))) = (a_1(\dots(a_{n-1}(a_n))\dots)).$$

Solution :

Soit $rev(x) = g(x, \varepsilon)$ avec

$$\begin{aligned} g(\varepsilon, y) &= y \\ g(al, y) &= g(l, (ay)). \end{aligned}$$

Donc,

$$\begin{aligned} rev((a_1(a_2(\dots(a_n)\dots)))) &= g((a_1(a_2(\dots(a_n)\dots))), \varepsilon) \\ &= g(\varepsilon, (a_n(\dots(a_2(a_1\varepsilon))\dots))) = (a_n(\dots(a_2(a_1))\dots)). \end{aligned}$$

Dans l'??, alors même que les listes de $X = A^*$ étaient engendrées sous la forme $((\dots(a_1(a_2))\dots a_{n-1})a_n)$, où nous avons ajouté les parenthèses pour souligner la structure implicite dans l'?? 2), A^* est muni d'une concaténation associative et le miroir est défini comme une application de X dans A^* . Ici, les listes de L sont engendrées sous la forme $(a_1(a_2(\dots(a_n(a_{n+1}))\dots)))$ en préfixant par les lettres de A , et, comme rev est une application $L \rightarrow L$, les seules opérations dont nous disposons dans L sont les préfixages par une lettre de A . rev doit donc être défini en termes de ces préfixages exclusivement, d'où la définition plus complexe.

Les notations qui suivent sont communes aux exercices 7 à 1.25. Un ensemble E muni d'une opération $*$ associative est un *semi-groupe*. Si de plus E possède un élément neutre e pour $*$ on dit que $(E, *, e)$ est un *monoïde*. Étant donné un ensemble A , le *monoïde libre* sur A , noté A^* , est l'ensemble des suites finies d'éléments de A (dont la suite vide, notée ε). Dans ce cas, l'opération qui fait de A^* un monoïde est la concaténation, notée $.$ ("point"), qui admet la suite vide ε , pour élément neutre.

Exercice 7

On considère le monoïde libre A^* sur un alphabet fini A . On note u^p le mot $u \dots u$, où p exemplaires de u ont été concaténés. On dit que mot u est un *facteur gauche* d'un mot v , s'il existe un mot w tel que $v = uw$, (de même, u est facteur droit de v si on peut écrire $v = wu$). Soit α et β deux mots.

1. Montrer qu'une condition nécessaire et suffisante pour que l'un d'entre eux soit facteur gauche de l'autre est qu'il existe deux mots u et v tels que $\alpha u = \beta v$.
2. On suppose que $\alpha\beta = \beta\alpha$. Montrer que cela équivaut à : $\exists \gamma \in A^*, \exists (p, q) \in \mathbb{N} \times \mathbb{N}, \alpha = \gamma^p, \beta = \gamma^q$.

Solution :

Par récurrence sur $|\alpha\beta|$. D'après la question précédente, l'un des deux mots α ou β (le plus court) est facteur gauche de l'autre. Supposons que ce soit α , si α est le mot vide, le résultat est immédiat avec $\gamma = \beta, p = 0$ et $q = 1$, sinon on peut écrire $\beta = \alpha\beta_1$ avec $|\alpha\beta_1| < |\alpha\beta|$, en remplaçant, on obtient après simplification à gauche par $\alpha : \alpha\beta_1 = \beta_1\alpha$. On applique l'hypothèse de récurrence : $\alpha = \gamma^p$ et $\beta_1 = \gamma^q$ et finalement $\beta = \gamma^{p+q}$.

Exercice 8

Les notations sont les mêmes que dans l'exercice 7. On appelle *langage* (sur l'alphabet A) un sous-ensemble de A^* . On définit inductivement les langages *rationnels* (on dit aussi *réguliers*) par :

- (i) Un langage fini est rationnel,
- (ii) Si L_1 et L_2 sont rationnels alors $L_1 \cup L_2$ est rationnel,
- (iii) Si L_1 et L_2 sont rationnels alors $L_1 L_2 = \{uv, u \in L_1, v \in L_2\}$ est rationnel,
- (iv) Si L est rationnel alors $L^* = \bigcup_{n \geq 0} L^n$ est rationnel ($L^0 = \{\epsilon\}$).

On appelle *miroir* du mot $u = a_1 \dots a_n$, le mot $\tilde{u} = a_n \dots a_1$. Si L est un langage, on définit $\tilde{L} = \{\tilde{u}, u \in L\}$. Montrer que si L est un langage rationnel, alors \tilde{L} est également rationnel.

Solution :

Par récurrence sur le nombre d'opérations permettant d'obtenir le langage rationnel L , en séparant suivant la nature de la dernière opération :

- Si L est fini, \tilde{L} est également fini,
- Si $L = L_1 \cup L_2$ avec L_1 et L_2 rationnels, alors \tilde{L}_1 et \tilde{L}_2 sont rationnels par hypothèse de récurrence, or $\tilde{L} = \widetilde{L_1 \cup L_2} = \tilde{L}_1 \cup \tilde{L}_2$, donc \tilde{L} est rationnel,
- Si $L = L_1 L_2$ c'est la même chose (on a : $\tilde{L} = \widetilde{L_1 L_2} = \tilde{L}_2 \tilde{L}_1$),
- Si $L = L_0^*$, L_0 rationnel, donc \tilde{L}_0 également, on montre ensuite par une récurrence (triviale) sur n que $\tilde{L}^n = \tilde{L}^n$, ce qui termine la démonstration.

Exercice 9

Soit X un ensemble de symboles de variable et F un ensemble de symboles de fonction, on définit inductivement l'ensemble $T(F, X)$ des termes sur F et X de la façon suivante :

- (B) Si $x \in X$ alors $x \in T(F, X)$; si $f \in F_0$, alors $f \in T(F, X)$;
- (I) Si $t_1, \dots, t_n \in T(F, X)$, avec $n \geq 1$ et $f \in F_n$, alors $f(t_1, \dots, t_n) \in T(F, X)$. (F_n est l'ensemble des symboles de fonction d'arité n .)
1. Une application φ de $T(F, X)$ dans $T(F, X)$ est un morphisme si elle vérifie $\varphi(f_0) = f_0$ pour $f_0 \in F_0$ et $\varphi(f(t_1, \dots, t_n)) = f(\varphi(t_1), \dots, \varphi(t_n))$ pour $f \in F_n, n > 0$. Soit une application h de X dans $T(F, X)$, montrer qu'il existe un morphisme unique h^* de $T(F, X)$ dans $T(F, X)$ qui prolonge h (c'est-à-dire, tel que $\forall x \in X, h^*(x) = h(x)$).

Solution :

Existence : on définit inductivement le prolongement h^* en posant $h^*(t) = h(t)$ si $t \in X$ et $h^*(f(t_1, \dots, t_n)) = f(h^*(t_1), \dots, h^*(t_n))$ sinon. Il est clair que h^* prolonge h , et que c'est un morphisme.

Unicité : soit deux morphismes h_1^* et h_2^* prolongeant h , montrons par induction structurelle qu'ils sont égaux :

- (i) si $t \in X$ alors $h_1^*(t) = h_2^*(t) = h(t)$,
- (ii) si $t = f(t_1, \dots, t_n)$ alors $h_1^*(t) = f(h_1^*(t_1), \dots, h_1^*(t_n))$ et $h_2^*(t) = f(h_2^*(t_1), \dots, h_2^*(t_n))$, par hypothèse d'induction, $h_1^*(t_1) = h_2^*(t_1), \dots, h_1^*(t_n) = h_2^*(t_n)$, d'où l'égalité $h_1^*(t) = h_2^*(t)$.

2. Une substitution est une opération sur les termes permettant de remplacer certaines variables d'un terme par un terme. Plus formellement, l'ensemble des substitutions est l'ensemble des fonctions de X dans $T(F, X)$:

$$\Theta = \{\theta : X \longrightarrow T(F, X)\}$$

On notera s_{id} la substitution identité, c'est-à-dire la substitution telle que pour toute variable $x \in X$, on a $s_{id}(x) = x$. Une substitution θ se prolonge de façon unique en une fonction θ^* de $T(F, X)$ dans $T(F, X)$ définie par :

$$\forall x \in X, \theta^*(x) = \theta(x) \text{ et } \forall f_0 \in F_0, \theta^*(f_0) = f_0$$

$$\forall f \in F_n, \forall t_1, t_2, \dots, t_n \in T(X, F), \theta^*(f(t_1, t_2, \dots, t_n)) = f(\theta^*(t_1), \theta^*(t_2), \dots, \theta^*(t_n))$$

Appliquer une substitution à un terme c'est donc instancier certaines de ses variables par des termes et donc faire augmenter sa taille.

Montrer que :

$$\forall \theta \in \Theta, \forall t \in T(X, F), \tau(\theta^*(t)) \geq \tau(t)$$

où la taille d'un terme est donnée par l'application $\tau : T(X, F) \longrightarrow \mathbb{N}$ définie par :

$$\tau(t) = \begin{cases} 0 & \text{si } t = x, \\ 1 & \text{si } t = f_0 \in F_0 \\ 1 + \sum_{i=1}^n \tau(t_i) & \text{si } t = f(t_1, \dots, t_n). \end{cases}$$

Solution :

On procède par induction sur t . Si $t = x$, alors $\tau(\theta^*(x)) \geq \tau(x) = 0$, sinon on a :

$$\tau(\theta^*(f(t_1, t_2, \dots, t_n))) = 1 + \sum_{i=1}^n \tau(\theta^*(t_i)) \underset{(hyp.dinduction)}{\geq} 1 + \sum_{i=1}^n \tau(t_i) = \tau(f(t_1, t_2, \dots, t_n))$$

3. On peut définir un préordre (i.e. une relation réflexive et transitive) sur $T(X, F)$ comme suit : $t_1 \preceq t_2$ si et seulement si il existe une substitution θ telle que $\theta^*(t_1) = t_2$. Montrer que cette relation définit bien un préordre (ce préordre est appelé préordre de filtrage).

Solution :

Il s'agit de montrer que \preceq est une relation réflexive et transitive.

- (i) \preceq est une relation réflexive puisque pour tout terme t , $t = s_{id}^*(t)$ et donc $t \preceq t$
- (ii) \preceq est une relation transitive puisque étant donnés trois termes t_1, t_2 et t_3 , si $t_1 \preceq t_2$, c'est-à-dire s'il existe une substitution θ_1 telle que $\theta_1^*(t_1) = t_2$, et si $t_2 \preceq t_3$, c'est-à-dire s'il existe une substitution θ_2 telle que $\theta_2^*(t_2) = t_3$, alors $\theta_2^*(\theta_1^*(t_1)) = t_3$ et on a donc $t_1 \preceq t_3$.

4. Pourquoi le préordre de filtrage ne définit-il pas une relation d'ordre (i.e. une relation réflexive, antisymétrique et transitive) ?

Solution :

\preceq ne définit pas un ordre, puisque $t_1 \preceq t_2$ et $t_2 \preceq t_1$ n'implique pas forcément $t_1 = t_2$. Par exemple, si on considère les deux termes $g(x)$ et $g(y)$, on a d'une part, $g(x) \preceq g(y)$ puisque la substitution θ_1 qui associe y à x et laisse inchangées les autres variables vérifie bien $\theta_1(g(x)) = g(y)$ et d'autre part, on a $g(y) \preceq g(x)$ puisque la substitution θ_2 qui associe x à y et laisse inchangées les autres variables vérifie bien $\theta_2(g(y)) = g(x)$.

5. Dans cette question nous allons voir qu'il est possible de définir une relation d'ordre à partir d'un préordre.

Soit E un ensemble muni d'un préordre \preceq .

- a) Montrer que la relation \approx définie par $x \approx y$ ssi soit $x = y$ soit $x \preceq y$ et $y \preceq x$ est une relation d'équivalence (i.e., une relation réflexive, symétrique et transitive).
- b) On note $[x]_{\approx} = \{y \mid y \approx x\}$ la classe d'équivalence de x et E/\approx l'ensemble des classes d'équivalence de E pour \approx (ensemble quotient). Montrer que la relation \succeq définie sur E/\approx par $[x]_{\approx} \succeq [y]_{\approx}$ ssi $x \preceq y$ est une relation d'ordre (cet ordre est appelé ordre quotient du préordre).

Solution :

Trivial par définition.

6. D'après la question précédente, on peut définir une relation d'équivalence sur les termes comme suit :

$$t_1 \equiv t_2 \text{ ssi } t_1 = t_2 \text{ ou } (t_1 \preceq t_2 \text{ et } t_2 \preceq t_1)$$

En fait, la relation \equiv identifie les termes équivalents à un renommage près des variables.

À partir de cette relation d'équivalence entre termes, il est alors possible de construire une relation d'ordre \preceq sur l'ensemble quotient $T(X, F)/\equiv$ définie par :

$$[t_1]/\equiv \preceq [t_2]/\equiv \text{ ssi } t_1 \preceq t_2$$

On remarquera que tous les termes d'une même classe d'équivalence ont la même taille.

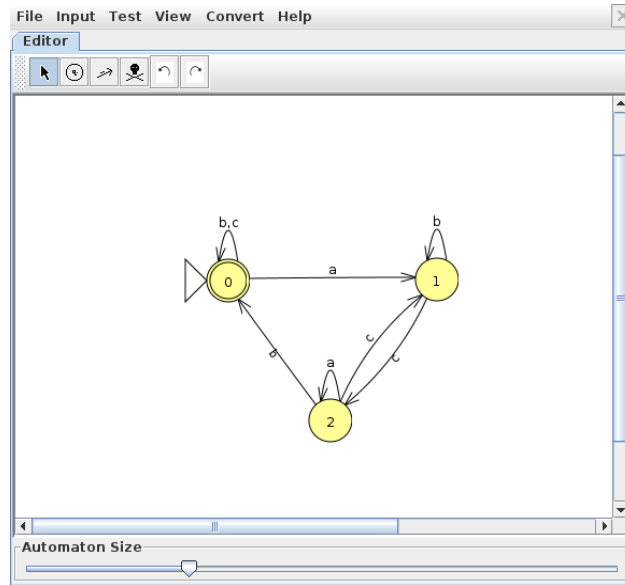
- (i) Soit $\Lambda(t)$ le nombre de variables distinctes apparaissant dans le terme t . Montrer que $t \prec t' \text{ et } \tau(t) = \tau(t') \implies \Lambda(t') < \Lambda(t)$.
- (ii) Montrer que \preceq est un ordre bien fondé sur $T(X, F)/\equiv$.

Solution :

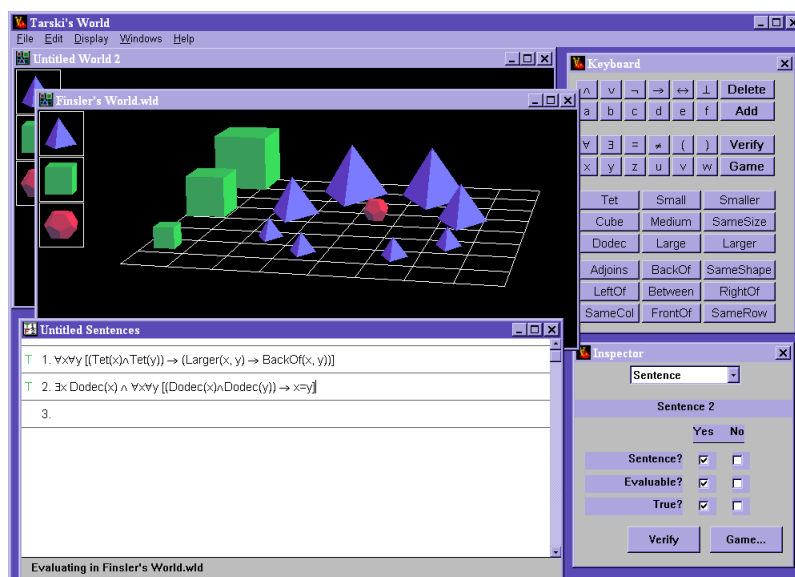
- (i) d'abord un exemple : $g(x, y) \prec g(x, x) = \theta^*(g(x, y))$ où $\theta(y) = x = \theta(x)$.

Ensuite on remarque que si $t \prec t'$ et $\tau(t) = \tau(t')$ alors $t' = \theta^*(t)$ avec une substitution θ qui ne peut que remplacer des variables par d'autres variables (sinon la taille du terme augmente) et qui identifie forcément au moins 2 variables (sinon ce serait un renommage et les termes seraient équivalents), d'où ce qu'on veut.

- (ii) Montrons qu'il n'existe pas de suite infinie $([t_i]/\equiv)_{i \in \mathbb{N}}$ strictement décroissante. Pour cela, on note $\Lambda(t)$ le nombre de variables distinctes apparaissant dans le terme t . On vérifie facilement que si $t_1 \equiv t_2$ alors $\Lambda(t_1) = \Lambda(t_2)$ et on peut donc étendre Λ à $T(X, F)/\equiv$ en posant $\Lambda([t_i]/\equiv) = \Lambda(t)$. Soit $([t_i]/\equiv)_{i \in \mathbb{N}}$ une suite strictement décroissante. Pour tout i , on a $t_{i+1} \prec t_i$ et donc il existe une suite de substitutions $(\theta_i)_{i \in \mathbb{N}}$ telle que $\theta_i^*(t_i) = t_{i+1}$ et puisqu'on a montré que $\tau(t_{i+1}) = \tau(\theta_i^*(t_i)) \geq \tau(t_i)$, on en déduit que $(\tau(t_i))_{i \in \mathbb{N}}$ est une suite décroissante. \leq étant un ordre bien fondé sur \mathbb{N} , cette suite est stationnaire à partir d'un certain rang k et donc pour tout $i \geq k$, on a $\tau(t_i) = \tau(t_{i+1})$. Or puisque $\theta_{i+1}^*(t_{i+1}) = t_i$, l'image par θ_{i+1}^* de chaque variable x apparaissant dans t_{i+1} est une variable. De plus puisque la suite $([t_i]/\equiv)_{i \in \mathbb{N}}$ est strictement décroissante, θ_{i+1} n'est pas un simple renommage (i.e. θ_i n'est pas injective) et il existe (au moins) deux variables distinctes y_1 et y_2 apparaissant dans t_{i+1} telles que $\theta_{i+1}(y_1) = \theta_{i+1}(y_2)$ et donc $\Lambda(t_{i+1}) = \Lambda(\theta_i^*(t_i)) > \Lambda(t_i)$. On en déduit donc que la suite $(\Lambda(t_i))_{i \geq k}$ est strictement croissante. Or, $(\tau(t_i))_{i \geq k}$ étant stationnaire, la suite $(\Lambda(t_i))_{i \geq k}$ est bornée par le nombre d'occurrences de variables dans t_k ce qui est contradictoire.



TME LI214



TME Langages et automates

1 Création d'automates

Le logiciel JFLAP se trouve sous Unix ; si vous le souhaitez, vous pouvez le télécharger à cette adresse :

<http://www.cs.duke.edu/csed/jflap/>

où il y a des versions pour Unix, Windows et MacOS.

Lancer le logiciel JFLAP avec la commande

```
java -jar /usr/local/jflap/JFLAP.jar
```

et choisir **Finite Automaton**. Il apparaît alors une fenêtre d'édition, dans laquelle on peut dessiner un automate. La barre d'outils située sous l'onglet **Editor** contient quatre boutons ; de gauche à droite :

- Attribute Tool que l'on nommera ici bouton A
- State Tool (bouton S)
- Transition Tool (bouton T)
- Deleter (bouton D).

Exercice 1

Soit l'automate \mathcal{A} défini sur l'alphabet $\{a, b\}$, d'états 0, 1, 2, d'état initial 0, d'état final 2 et de transitions $0.a = 0$, $0.b = 1$, $1.a = 2$, $1.b = 2$, $2.a = 0$ et $2.b = 1$.

Création de l'automate \mathcal{A}

Pour dessiner l'automate \mathcal{A} :

- créer les états : cliquer sur le bouton S ; cliquer en trois endroits différents de la fenêtre d'édition, les états apparaissent avec les noms q_0 , q_1 , q_2
- choisir la nature des états (initial, final) : cliquer sur le bouton A ; faire un clic droit sur l'état q_0 et choisir *Initial* ; faire un clic droit sur l'état q_2 et choisir *Final*
- créer les transitions : cliquer sur le bouton T
 - transition $0.a = 0$: cliquer sur l'état q_0 et taper a dans le cadre qui apparaît
 - transition $0.b = 1$: promener la souris, bouton gauche enfoncé, de l'état q_0 à l'état q_1 , lâcher le bouton et taper b dans le cadre qui apparaît
 - autres transitions : sur le modèle de $0.b = 1$
 - attention : il faut créer deux transitions pour $1.a = 2$ et $1.b = 2$ (ne pas taper les deux étiquettes a et b dans le même cadre).
- sauver le fichier en le nommant `ex01-tme`.

Vous découvrirez tout seul les autres utilisations du bouton A et l'utilisation du bouton D.

Reconnaissance de mots par \mathcal{A}

Pour vérifier si des mots sont acceptés, on utilise l'un des menus `Input/Step by State` ou `Input/Fast Run` ou `Input/Multiple Run`.

- Menu `Input/Step by State` : sous la fenêtre contenant l'automate apparaissent une fenêtre montrant les différentes configurations et une barre contenant six boutons.

En cliquant sur le bouton `Step`, le mot est lu lettre à lettre, la progression dans le mot est visible dans la fenêtre des configurations et la progression dans l'automate est visible dans la fenêtre de l'automate.

Après le dernier `Step`, la configuration est soit verte (le mot est accepté) soit rouge (le mot est rejeté).

En cliquant sur une configuration puis sur le bouton `Trace`, on voit apparaître, dans une autre fenêtre, la suite des transitions qui ont amené à cette configuration.

- Menus `Input/Fast Run` et `Input/Multiple Run` : utilisation évidente.

Remarque : il y a aussi un menu `Input/Step by Closure` mais il n'a d'intérêt que pour les automates avec λ -transitions (transitions étiquetées par le mot vide).

1. En utilisant chacun des trois menus précédents, vérifier si les mots suivants sont acceptés ou non par l'automate \mathcal{A} : ba , $babb$, $aaabab$, $aaababa$, $abaabb$, $abaabaaab$.

Exercice 2

On considère l'automate \mathcal{B} obtenu en ajoutant la transition $2.b = 2$ à l'automate \mathcal{A} défini dans l'exercice 1.

1. Dessiner l'automate \mathcal{B} et sauvegarder le fichier en le nommant `exo2-tme`.
2. Choisir le menu `Test/Compare Equivalence` et tester si les automates \mathcal{A} et \mathcal{B} sont équivalents.
3. Utiliser le menu `Test/Highlight Nondeterminism` pour vérifier que l'automate \mathcal{B} n'est pas déterministe.
4. Utiliser le menu `Input/Step by State` pour vérifier que le mot *ababbba* est accepté.

Dans la fenêtre des configurations, il apparaît cinq configurations correspondant aux cinq lectures possibles du mot *ababbba* ; on remarque que certaines configurations sont acceptées et d'autres pas. Faire afficher la trace de chacune des cinq configurations.

5. Tester d'autres mots, en utilisant `Input/Step by State` ou `Input/Fast Run`.

Exercice 3

Soit $A = \{a, b, c\}$. Dessiner des automates, non nécessairement déterministes, reconnaissant les langages suivants :

1. $\{a, ab, ca, cab, acc\}$
2. $[a(b+c)^*abc]^*$
3. l'ensemble des mots contenant un nombre impair de a
4. l'ensemble des mots contenant le facteur ab
5. l'ensemble des mots contenant le facteur ab et se terminant par b

2 Déterminisation d'automates

Exercice 4

Dessiner l'automate \mathcal{A} sur l'alphabet $\{a, b, c\}$ d'états 0, 1, 2, 3, d'état initial 0, d'état terminal 3 et de transitions $0.a = 1$, $0.a = 0$, $0.b = 0$, $0.c = 0$, $1.a = 2$, $1.b = 2$, $1.c = 2$, $2.a = 3$, $2.b = 3$, $2.c = 3$ (cf. ex. 4 du TD), en faisant en sorte que l'état i soit nommé q_i dans le dessin. Vérifier que \mathcal{A} n'est pas déterministe.

Pour déterminer l'automate \mathcal{A} , on choisit le menu

`Convert/Convert to DFA`.

La fenêtre se partage en deux :

- la partie gauche contient l'automate \mathcal{A}
- la partie droite est une sous-fenêtre de travail, dans laquelle on construit le déterminisé \mathcal{A}' de \mathcal{A} , et dont la barre d'outils contient cinq boutons ; de gauche à droite :
 - `Attribute Editor` (bouton A)
 - `Expand Group on Terminal` (bouton T)
 - `State Expander` (bouton S)
 - `Complete`
 - `Done?`.

Initialement, la sous-fenêtre de travail contient un seul état, l'état initial, nommé q_0 , auquel est attaché un petit cadre contenant tous les indices des états initiaux de \mathcal{A} (ici, le cadre contient 0). Il faut construire toutes les transitions de \mathcal{A}' , c'est-à-dire les transitions : $\{0\}.a = \{0, 1\}$, $\{0\}.b = \{0\}$, $\{0\}.c = \{0\}$, $\{0, 1\}.a = \{0, 1, 2\}$, $\{0, 1\}.b = \{0, 2\}$, etc...

Il y a plusieurs façons de construire les transitions :

- on peut les construire une à une : cliquer sur le bouton T
 - transition $\{0\}.a = \{0, 1\}$: promener la souris, bouton gauche enfoncé, de l'état q_0 de \mathcal{A}' vers un endroit quelconque de la sous-fenêtre de travail ; lâcher le bouton ; une boîte de dialogue apparaît, demandant l'étiquette de la transition que l'on veut construire ("Expand on what terminal ?") ; taper a ; une nouvelle boîte de dialogue apparaît, demandant l'état but de la transition que l'on veut construire ; taper 0 et 1 en les séparant par un espace ; apparaissent alors l'état q_1 avec un petit cadre contenant 0,1 et la transition d'étiquette a qui va de q_0 à q_1 .
 - si l'état but de la transition existe déjà dans \mathcal{A}' , procéder comme dans la création d'automates ; par exemple, pour construire la transition $\{0\}.b = \{0\}$: cliquer sur l'état q_0 de \mathcal{A}' puis taper b dans la boîte de dialogue
 - on peut construire en une seule fois toutes les transitions issues d'un état q de \mathcal{A}' : cliquer sur le bouton S puis sur l'état q
 - on peut construire toutes les transitions en une seule fois : cliquer sur le bouton `Complete`.
- 1. Construire l'automate \mathcal{A}' , déterminisé de \mathcal{A} , en utilisant uniquement le bouton T.

Exercice 5

Déterminer l'automate \mathcal{B} défini dans l'exercice 2 (sans utiliser le bouton `Complete`).

Exercice 6

Tester si les automates construits dans l'exercice 3 sont déterministes et déterminer ceux qui ne le sont pas (sans utiliser le bouton `Complete`).

3 Minimisation d'automates

Pour minimiser un automate \mathcal{A} , on utilise l'algorithme suivant :

- initialement l'ensemble des états est partagé en deux sous-ensembles : l'ensemble des états terminaux et l'ensemble des états non terminaux
- on réitère le processus suivant :
 - parmi les ensembles d'états déjà construits $Q_1, Q_2 \dots Q_m$, on choisit un ensemble Q_i
 - on choisit une lettre x
 - on partage l'ensemble Q_i en plusieurs sous-ensembles : deux états p et q de Q_i appartiennent au même sous-ensemble ssi les états $p.x$ et $q.x$ de \mathcal{A} appartiennent à un même ensemble Q_j
- et ce jusqu'à ce qu'aucune lettre ne puisse plus partager l'un des ensembles d'états.

Exercice 7

Soit l'automate \mathcal{A} défini sur l'alphabet $\{a, b\}$, d'états 0, 1, 2, 3, 4, 5, d'état initial 0, d'état final 4 et de transitions $0.a = 1$, $0.b = 2$, $1.a = 3$, $1.b = 3$, $2.a = 3$, $2.b = 3$, $3.a = 4$, $3.b = 5$, $4.a = 4$, $4.b = 4$, $5.a = 5$ et $5.b = 4$.

Pour minimiser l'automate \mathcal{A} , on peut dérouler l'algorithme de plusieurs manières.

Une première manière :

- initialement : $\{0, 1, 2, 3, 5\}$ et $\{4\}$
- on partage $\{0, 1, 2, 3, 5\}$ en utilisant a : $\{0, 1, 2, 5\}$, $\{3\}$ et $\{4\}$
- on partage $\{0, 1, 2, 5\}$ en utilisant b : $\{0\}$, $\{1, 2\}$, $\{5\}$ et $\{3\}$, $\{4\}$
- on ne peut pas partager $\{1, 2\}$ (ni en utilisant a , ni en utilisant b).

Une deuxième manière :

- initialement : $\{0, 1, 2, 3, 5\}$ et $\{4\}$
- on partage $\{0, 1, 2, 3, 5\}$ en utilisant a : $\{0, 1, 2, 5\}$, $\{3\}$ et $\{4\}$
- on partage $\{0, 1, 2, 5\}$ en utilisant a : $\{0, 5\}$, $\{1, 2\}$ et $\{3\}$, $\{4\}$
- on partage $\{0, 5\}$ en utilisant a : $\{0\}$, $\{5\}$ et $\{1, 2\}$, $\{3\}$, $\{4\}$
- on ne peut pas partager $\{1, 2\}$ (ni en utilisant a , ni en utilisant b).

Et il y a encore d'autres manières...

Minimisation de l'automate \mathcal{A}

Dessiner l'automate \mathcal{A} , en faisant en sorte que l'état i soit nommé q_i dans le dessin.

Pour construire l'automate \mathcal{A}' , minimisé de l'automate \mathcal{A} , on choisit le menu `Convert/Minimize DFA`. La fenêtre se partage en deux :

- la partie gauche contient l'automate \mathcal{A} (ou le complété de \mathcal{A} si \mathcal{A} n'est pas complet)
- la partie droite est une sous-fenêtre de travail qui contient un arbre dont les feuilles représentent des ensembles d'états.

Initialement, l'arbre a seulement deux feuilles : l'une représente les états non terminaux (ici 0, 1, 2, 3, 5) et l'autre les états terminaux (ici 4). Pour compléter cet arbre, on partage les ensembles d'états comme il est dit dans l'algorithme de minimisation.

En suivant la première manière :

- pour partager $\{0, 1, 2, 3, 5\}$ en utilisant a : cliquer sur le cadre contenant 0, 1, 2, 3, 5 puis cliquer sur le bouton `Set Terminal` taper a dans la boîte de dialogue ; sous le cadre contenant 0, 1, 2, 3, 5, apparaissent deux sous-arbres, dont les feuilles sont vides ; on remplit l'une des deux feuilles avec 0, 1, 2, 5 et l'autre feuille avec 3 ; pour cela, cliquer sur l'une des deux feuilles puis, successivement, sur les états q_0, q_1, q_2, q_5 de \mathcal{A} ; ensuite, cliquer sur l'autre feuille puis sur l'état q_3 de \mathcal{A} .
Remarque : la même démarche (cliquer sur la feuille de l'arbre puis sur l'état de l'automate \mathcal{A}) permet d'enlever un état déjà présent dans une feuille ; cliquer sur le bouton `Check Node` pour soumettre la partition ;

- pour partager $\{0, 1, 2, 5\}$ en utilisant b : cliquer sur le cadre contenant 0, 1, 2, 5 puis cliquer sur le bouton `Set Terminal` ; taper b dans la boîte de dialogue ; sous le cadre contenant 0, 1, 2, 5, apparaissent deux sous-arbres ; comme la lettre b partage $\{0, 1, 2, 5\}$ en trois sous-ensembles ($\{0\}$, $\{1, 2\}$ et $\{5\}$), il faut ajouter un sous-arbre (bouton `Add Child`) ; remplir les trois feuilles ; cliquer sur le bouton `Check Node` pour soumettre la partition ; si elle n'est pas correcte, un message d'erreur s'affiche et il faut alors modifier la composition des feuilles ; si elle est correcte, le message "The expansion is correct !" s'affiche
- comme il n'y a plus d'ensemble d'états à partager, le seul bouton accessible est le bouton `Finish` ; cliquer dessus ; dans la sous-fenêtre de travail, l'arbre est remplacé par l'ensemble des états de l'automate minimal ; construire les transitions de l'automate minimal
- cliquer sur le bouton `Done` ; s'il manque des transitions, un message le signale.

Refaire la minimisation de l'automate \mathcal{A} en suivant la deuxième manière de dérouler l'algorithme.

Exercice 8

Soit l'automate \mathcal{A} d'états 0, 1, 2, 3, 4, 5, d'état initial 0, d'état terminal 5 et de transitions : $0.a = 1$, $1.a = 2$, $2.a = 3$, $3.a = 4$, $4.a = 5$, $0.b = 3$, $1.b = 2$, $2.b = 3$, $3.b = 4$, $4.b = 5$, $5.b = 5$. Dessiner l'automate \mathcal{A} . Minimiser \mathcal{A} .

Exercice 9

1. Soit l'automate \mathcal{A} d'états 0, 1, 2, 3, 4, 5, d'état initial 0, d'état terminal 3 et de transitions : $0.a = 1$, $1.a = 1$, $2.a = 4$, $3.a = 5$, $4.a = 4$, $5.a = 5$, $0.b = 2$, $1.b = 3$, $2.b = 2$, $3.b = 3$, $4.b = 5$, $5.b = 5$. Dessiner l'automate \mathcal{A} . Minimiser \mathcal{A} .
2. Soit l'automate \mathcal{B} d'états 0, 1, 2, 3, 4, 5 d'état initial 0, d'états terminaux 3, 4, 5 et de même transitions que \mathcal{A} . Dessiner l'automate \mathcal{B} . Minimiser \mathcal{B} .

Exercice 10

1. Soit l'automate \mathcal{A} d'états 0, 1, 2, 3 d'état initial 0, d'état terminal 2 et de transitions : $0.a = 0$, $0.a = 1$, $1.a = 1$, $3.a = 0$, $1.b = 1$, $1.b = 2$, $3.b = 0$, $3.b = 2$, $1.c = 1$, $1.c = 3$, $3.c = 0$.
2. Dessiner l'automate \mathcal{A} .
3. Déterminer \mathcal{A} .
4. Minimiser \mathcal{A} .

4 À découvrir seul

- 1) Le menu `Convert/Convert FA to RE` de la fenêtre `Editor` permet de calculer une expression rationnelle pour le langage reconnu par un automate.

Pour chacun des exercices 22, 24, 25, 26 de la feuille de TD, on pourra comparer l'expression rationnelle calculée par le logiciel JFLAP à l'expression rationnelle calculée en TD. On pourra au préalable traiter l'exercice suivant :

Exercice 11

Soient X et Y des langages sur un alphabet A . Montrer que

$$(X + Y)^* = (X^*Y)^*X^* = X^*(YX^*)^*$$

- 2) Le bouton `Regular Expression` de la fenêtre

`New Document`. Il permet de dessiner l'automate reconnaissant un langage donné par une expression rationnelle (ne contenant que des $+$, $.$ et $*$). Pour chacune des expressions rationnelles de l'exercice 21 de la feuille de TD, on pourra comparer l'automate construit par le logiciel JFLAP à l'automate construit en TD.

TME Logique

Machine : sous WINDOWS, ENSEIGNEMENT

Tarski se trouve dans *tout le réseau / Enseignement / C/ Program Files*

Faire une copie du fichier d'exercices dans *votre directory personnel h* : vous ne pourrez travailler que sur cette copie.

Tarski's world est un petit logiciel illustrant de façon simple et récréative les notions de modèle et d'interprétation des formules du calcul des prédicats.

1 Présentation

Domaine d'interprétation

Le *monde* servant de support au domaine d'interprétation de formules est une grille carrée de 8 sur 8 sur laquelle on peut disposer trois sortes de figures géométriques de tailles différentes. Les figures sont : la pyramide, le cube et le dodécaèdre. Les tailles sont : petit, moyen et grand.

Langage

Pour décrire ce monde géométrique, le logiciel propose un langage réduit (ensemble de symboles de variables, constantes et prédicats).

- 6 symboles de constantes : `a b c d e f`
- 6 symboles de variables : `u v w x y z`
- 13 symboles de prédicats dont
 - 6 symboles unaires : `Tet Cube Dodec Small Medium Large`
 - 6 symboles binaires : `Smaller Larger LeftOf RightOf BackOf FrontOf`
 - 1 symbole ternaire : `Between`

La sémantique attendue des symboles est indiquée par la signification du mot anglais utilisé pour les désigner. Pour en être sûr, vous pouvez déterminer la sémantique des symboles de prédicat en saisissant pour chacun d'eux une formule atomique et en construisant un monde où elle est valide et un monde où elle est fausse. A-t-on par exemple `Smaller(a, b)` si `a, b` sont tous deux *small*, ou bien `FrontOf(a, b)` si `a, b` sont sur une même horizontale ?

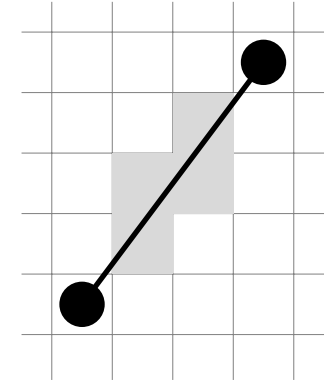
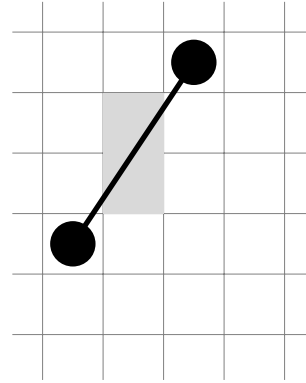
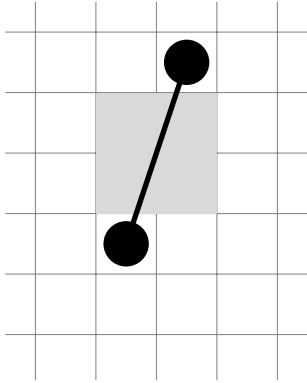
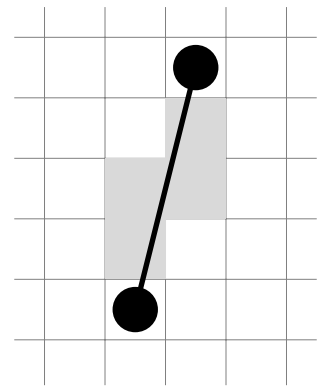
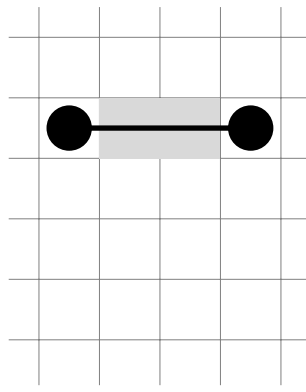
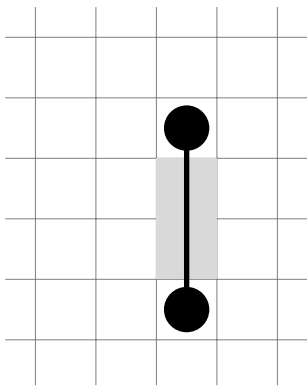
Ergonomie

Le logiciel offre une interface graphique aux fonctionnalités usuelles (usage de la souris, menus, etc.). Il présente à l'utilisateur quatre fenêtres :

1. une représentation graphique de la grille offrant, sur la gauche, les trois figures possibles. Cette fenêtre sert à construire un domaine d'interprétation. Au lancement du programme, cette fenêtre est intitulée `NEW0.WLD`.
2. un *clavier* dont chaque touche permet l'édition des symboles logiques (quantificateurs, connecteurs) et des symboles du langage (variables, constantes, prédicats).
3. une zone de saisie des formules séparées l'une de l'autre par un trait horizontal. L'utilisateur peut éditer une formule à l'aide du clavier du logiciel ou directement (clavier de sa console). Au lancement du programme, cette fenêtre est intitulée `NEW0.SEN`.
4. la dernière fenêtre, intitulée `Inspector` permet deux catégories d'opérations : l'édition des caractéristiques d'une figure ; la vérification de la validité d'une formule (fenêtre 3) dans le monde représenté dans la fenêtre 1. Le choix de l'une ou l'autre de ces opérations est accessible par un petit menu en haut de la fenêtre.

L'option `Block` correspond à l'édition des figures ; l'option `Sentence` correspond à la vérification des formules.

COMPLÉMENT : la figure suivante illustre l'interprétation du prédicat `Between`, les cases grisées sont celles qui satisfont `Between(z, x, y)` si `x` et `y` sont représentés par les points noirs.



Construction d'un monde

- En cliquant sur l'une des figures à gauche de la fenêtre, vous créez une nouvelle instance de cette figure sur la grille.
- Par *glisser-déposer* vous pouvez déplacer les figures d'une case à l'autre. Deux figures ne peuvent cohabiter sur une même case.
- En mode **Block** de la fenêtre **Inspector**, vous pouvez modifier les attributs d'une figure préalablement sélectionnée par un clic. On peut modifier la nature des figures, leur taille ou nommer une figure en utilisant un des symboles de constante (a, b, c, d, e ou f)

Remarque Une même figure peut recevoir plusieurs noms (symboles de constantes).

Validation d'une formule

Il y a trois niveaux de vérifications :

- correction syntaxique : *wff* i.e. *Well-Formed-Formula* ou encore formule bien formée selon les règles d'écriture des formules ;
- énoncé clos (ou non) : *sentence* (ou non) ;
- vérité : la formule est vraie dans le modèle considéré.

Pour chacun de ces niveaux, deux réponses sont possibles (*yes* ou *no*) que l'on sélectionne en cochant la réponse attendue. Un bouton **Verify** permet de demander au logiciel de contrôler la validité de vos choix. Une croix est affichée sur la ligne correspondante lorsque la réponse proposée est erronée et un signe ressemblant à un V est affiché sur la ligne correspondante lorsque la réponse proposée est correcte.

SYNTAXE les expressions sont complètement parenthésées – sauf lorsque le connecteur est associatif (conjonction ou disjonction)

VALIDITÉ seuls les énoncés clos peuvent être vrais ou faux.

Le jeu de la vérité

Le logiciel propose un mode *pas-à-pas* pour vérifier la validité d'une formule : le bouton **Game** de la fenêtre **Inspector**. À chaque étape de validation (ou d'invalidation) on peut vous demander de sélectionner une sous-formule, de sélectionner ou de nommer une figure du monde. La succession des étapes constitue une preuve de la vérité ou la fausseté de l'énoncé traité. En fait, cette preuve est partielle : par exemple, tous les cas d'une conjonction ne seront pas examinés.

Remarque : avant de pouvoir utiliser cette fonctionnalité, il faut avoir fait vérifier (bouton **Verify**) la vérité ou la fausseté de la formule visée.

Mondes prédéfinis

Le logiciel propose un certain nombre de mondes ou d'ensemble de formules prédéfinis accessibles par le menu **File**, option **Open**, puis **World** ou **Sentence**. Les mondes prédéfinis sont dans le sous répertoire **exercice**.

2 Exercices – Calcul propositionnel

Les 3 exercices suivants sont tirés du livre *Tarski's World* de Barwise et Etchemendy

Exercice 1

Chargez le fichier `DeMorg .sen`. Réaliser un monde qui rende toutes les formules vraies.

Exercice 2

1. Chargez le fichier `Boole .wld`.

Créez un fichier `formules .sen` où vous mettrez des formules exprimant que :

1. f (le grand dodécaèdre au fond) n'est pas devant a
 2. f est à droite de a et à gauche de b
 3. f est soit derrière a soit plus petit que a .
 4. e et d sont entre c et a .
 5. Ni e ni d ne sont plus grands que c .
 6. e n'est ni plus grand ni plus petit que d .
 7. c est plus grand que e et plus petit que a .
 8. c est plus petit que f et de plus il est devant f .
2. Changez le monde de manière à ce que toutes vos formules 1 à 8 deviennent fausses dans ce nouveau monde. Vous pourrez commencer par déplacer f dans le coin avant de la grille. Ensuite déplacez e dans le coin arrière gauche de la grille et faites-en un grand objet. Vérifiez toutes vos formules 1 à 8 dans ce monde. Si vous les avez formulées correctement elles devraient être toutes fausses. Si l'une d'elles est encore vraie, pouvez vous trouver votre erreur.

Exercice 3

1. Créez un nouveau fichier de formules et inscrivez-y la traduction en logique du premier ordre des phrases suivantes :
 1. a est petit ou c et d sont grands.
 2. d et e sont tous deux derrière b .
 3. d et e sont tous deux derrière b et sont tous deux plus grands que b .
 4. c et d sont tous deux des cubes et de plus aucun des deux n'est petit.
 5. Ni e ni a n'est à droite de c et à gauche de b .
 6. Soit e n'est pas grand, soit e est derrière a .
 7. c n'est ni entre a et b , ni en avant de a , ni en avant de b .
 8. Soit a et e sont des tétraèdres, soit a et f sont des tétraèdres.
 9. Ni c ni d n'est devant soit c , soit b .
 10. c est entre d et f ou plus petit que d et plus petit que f .

Enregistrez cet ensemble de propositions sous le nom `exo3 .sentences`.

2. Chargez le fichier `Wittgenstein .wld`. Vérifiez que toutes les propositions écrites en français dans la question 1. sont vraies dans ce monde. Vérifiez que vos traductions sont toutes valides ; si ce n'est pas le cas corrigez-les.
3. Modifiez le monde de `Wittgenstein .wld` comme suit : réduisez à la taille *small* tous les objets moyens ou grands, et agrandissez à la taille maximale tous les petits objets. Après ces changements, les formules 1, 3, 4 et 10 deviennent fausses, et toutes les autres restent vraies. Vérifiez qu'il en est bien ainsi dans votre fichier `exo3 .sentences` (si ce n'était pas le cas, corrigez vos formules). Ensuite faites pivoter de 90 degrés vers la droite le monde ainsi modifié : les formules 5, 6, 8 et 9 devraient être les seules à être vraies. Corrigez vos formules si ce n'était pas le cas.
Chargez le fichier `Boole .wld` : seule la formule 6 doit être vraie dans ce monde. Corrigez vos formules si ce n'était pas le cas. Modifiez `Boole .wld` en échangeant les positions de b et c . Les formules 2, 5, 6 et 7 deviennent vraies et les autres restent fausses. Corrigez vos formules si ce n'était pas le cas.

3 Exercices – Calcul des prédicats

Les 12 exercices suivants sont tirés du livre *Tarski's World* de Barwise et Etchemendy

Exercice 4

1. Ouvrir `Edgar.sen`. Évaluer les formules dans `Edgar.wld`.
2. Quelle formule traduit *il y a un grand tétraèdre* ?
3. Quelle formule traduit *il y a un cube entre a et b* ?
4. Exprimer en français les assertions faites par les formules 5 et 6.

Exercice 5

1. Ouvrir `Allan.sen`.
2. Quelle formule traduit *un dodécaèdre est grand* ? Construisez des mondes pour vérifier votre choix.
3. Quelle formule traduit *tous les tétraèdres sont petits* ? Construisez des mondes pour vérifier votre choix.
4. Construisez un monde où la formule 3 soit fausse et la formule 4 soit vraie.
5. Construisez un monde où les formules 2 et 4 soient vraies et les formules 1 et 3 soient fausses.
6. Peut-on construire un monde où la formule 1 soit vraie et la formule 2 soit fausse. Si oui construisez-le, si non expliquez pourquoi.

Exercice 6

Ouvrir un nouveau fichier de suffixe `.sen` et y mettre les formules traduisant les assertions suivantes (chaque formule contiendra un seul quantificateur qui sera universel) :

1. Tous les cubes sont petits.
2. Chaque petit cube est à droite de a.
3. Tous les dodécaèdres sont grands.
4. a est à gauche de tout dodécaèdre.
5. Tout tétraèdre moyen est devant b.
6. Chaque cube est devant b ou derrière a.
7. Tout cube est à gauche de b et à droite de a.
8. Tout ce qui se trouve entre b et a est un cube.
9. Tout ce qui est plus petit que a est un cube.
10. Tous les dodécaèdres ne sont pas petits. Remarquer que cette assertion est ambiguë : on peut la traduire par une formule commençant par \forall ou bien par une formule commençant par \neg . Vous choisirez la première formule, celle qui affirme que tous les dodécaèdres sont soit moyens soit grands.
11. Aucun dodécaèdre n'est petit.
12. a n'est pas à droite de tout objet. Remarquer que cette assertion est ambiguë : la traduire comme la négation de *a est à droite de tout*.
13. a n'est à droite d'aucun objet. Remarquer la différence avec l'assertion précédente.
14. a n'est pas à droite d'un cube.
15. Si un objet est un cube, alors cet objet est à gauche de b et à droite de a.
16. Un objet est un cube, si et seulement si cet objet est à gauche de b et à droite de a.

Sauvegardez votre fichier dans `exo6.sen`.

1. Ouvrez `Claire.wld`. Vérifiez que toutes vos formules sont vraies (sinon il y a une erreur dans vos formules que vous corrigerez).
2. Changez le fichier `Claire.wld` en déplaçant a dans l'angle droit au premier rang. Vérifiez que les formules 2, 4, 7, 13, 14, 15, 16 sont fausses et les autres vraies (sinon il y a une erreur dans vos formules que vous corrigerez).
3. Chargez le fichier `Wittgenstein.wld`. Remarquez que les assertions 2, 4, 8, 9, 12, 14 sont vraies et les autres fausses. S'il n'en est pas de même de vos formules, vous les corrigerez.
4. Chargez le fichier `Leibniz.wld`. Remarquez que les assertions 3, 4, 8, 9, 10, 11, 12, 13, 14 sont vraies et les autres fausses. S'il n'en est pas de même de vos formules, vous les corrigerez.

Exercice 7

Chargez le fichier `Leibniz.wld`. Ouvrez un nouveau fichier `exo7.sen` où vous mettrez des formules traduisant les assertions 1 à 5 suivantes. Chaque assertion que vous allez traduire est vraie dans ce monde. Vérifiez qu'il en est de même des formules que vous écrirez.

1. `b` est un tétraèdre plus petit que `e`.
 2. Il n'y a pas de cube moyen.
 3. Rien n'est devant `b`.
 4. Tout cube est soit devant soit derrière `e`.
 5. Aucun cube n'est entre deux objets (Il faut 3 quantificateurs pour traduire cette assertion).
1. Changez le monde de sorte que toutes les assertions deviennent fausses. Pour ce faire, transformez `b` en cube moyen, supprimez le tétraèdre le plus à gauche et mettez `b` à sa place, puis mettez un petit cube à l'ancienne place de `b`. Toutes vos formules doivent être fausses dans ce modèle ; si ce n'est pas le cas, vous les corrigerez.
 2. Faites divers changements au monde pour que certaines assertions soient vraies et d'autres fausses, et vérifiez que vos formules prennent la même valeur de vérité que les assertions.

Exercice 8

1. Chargez le fichier `Ramsey.sen`. Les formules de 1 à 10 sont soit existentielles soit sans quantificateur : on peut donc les rendre vraies en ajoutant des objets nouveaux pour chaque formule, mais un des buts de l'exercice est d'ajouter le moins possible d'objets. Réaliser un monde qui rende les formules 1 à 10 vraies. On essaiera de faire un tout petit monde : 6 objets doivent suffire (il faudra que l'un des objets ait 2 noms).
2. Si le monde réalisé à la question 1 ne satisfait pas les formules 11 à 20 (formules universelles), modifiez-le sans ajouter d'objet pour que toutes les formules 1 à 20 soient vraies.
3. Changez le monde de la question 2, uniquement en ajoutant des objets pour falsifier le plus grand nombre possible de formules de `Ramsey.sen`. Quelles sont les formules que vous arrivez à falsifier, et quelles sont celles que vous n'arriverez pas à falsifier ?

Exercice 9

1. Chargez le fichier `Peano.wld`.
Créez un fichier `formules.sen` où vous mettrez des formules exprimant que :
 1. Tout dodécaèdre est petit.
 2. Il y a un cube moyen.
 3. Il y a au moins deux cubes.
 4. Il y a un tétraèdre entre deux cubes.
 5. Tout cube n'est pas devant un dodécaèdre. Remarquer que cette assertion est ambiguë : la traduire comme la négation de *Tout cube est devant un dodécaèdre*.
2. Changez le monde en remplaçant le cube moyen par un dodécaèdre moyen. Toutes vos formules doivent devenir fausses dans ce nouveau monde.

Exercice 10

1. Chargez le fichier `Buridan.sen`. Construire un monde où toutes les formules sont vraies et sauvegardez-le dans `exo10.wld`.
2. Les assertions suivantes sont des conséquences des formules de `Buridan.sen`, elles doivent donc être vraies dans `exo10.wld`.
 1. Il n'y a pas de cube.
 2. Un tétraèdre n'est pas grand. remarquer que cette assertion est ambiguë pour certains, vous choisirez de l'interpréter comme *il y a tétraèdre qui n'est pas grand*
 3. Rien n'est derrière `a`.
 4. Il n'y que des grands objets derrière `b`.

Traduisez les assertions 1 à 4 en formules, ajoutez-les à `Buridan.sen` et sauvegardez le tout dans `exo10.sen` et vérifiez que toutes les formules de `exo10.sen` sont vraies dans `exo10.wld`.

3. Pour chaque formule traduisant une des assertions 1 à 4, modifiez le modèle `exo10.wld` pour qu'elle devienne fausse, et cherchez celle(s) des formules initiales de `Buridan.sen` qui sont devenues fausses.
4. Montrez que la formule 9 n'est pas conséquence des 13 autres formules : il suffit de construire un monde (que vous appellerez `exo10b.wld`) où 9 est fausse et où toutes les autres sont vraies.
5. Montrez que la formule F suivante (affirmant qu'il y a au moins 2 tétrèdres moyens) est indépendante des formules de `Buridan.sen` : i.e. F n'est pas conséquence de formules de `Buridan.sen` et $\neg F$ non plus.
$$\exists x \exists y ((x \neq y) \wedge Tet(x) \wedge Tet(y) \wedge Medium(x) \wedge Medium(y))$$

Il suffit de construire deux mondes, l'un où F est fausse et où toutes les formules de `Buridan.sen` sont vraies, et l'autre où $\neg F$ est fausse et où toutes les formules de `Buridan.sen` sont vraies.

Exercice 11

Chargez le fichier `Whitehead.sen`.

1. Testez les formules 1 à 7 dans des mondes à 1, 2, 3, ou 4 objets. Quelles formules affirment qu'il y a au moins (au plus exactement) 2 objets ? Quelles formules affirment qu'il y a au moins (au plus, exactement) 3 objets ?
2. Quelle est la différence entre ce qu'affirment les formules 8 et 9 ?
3. Construire un monde où la formule 10 est vraie.
4. Construire un monde où les formules 11 et 12 sont vraies.
5. À quelle formule est équivalente la formule 13 ? Traduisez en français. Vérifiez en construisant des modèles.
6. Qu'affirme la formule 14 ? Vérifiez en construisant des modèles.

Exercice 12

1. Chargez le fichier `Montague.sen`. Complétez les formules. Les assertions faites par les formules sont les suivantes :
 1. Tout cube est à gauche de tout tétraèdre : il faut donc compléter par $\forall y (Tet(y) \longrightarrow LeftOf(x, y))$
 2. Tout petit cube est derrière un grand cube.
 3. Un cube est devant tous les tétraèdres (assertion ambiguë).
 4. Un grand cube est devant un petit cube (assertion ambiguë).
 5. Rien n'est plus grand que tout.
 6. Tout cube qui est devant tous les tétraèdres est grand.
 7. Tout ce qui est à droite d'un grand cube est petit.
 8. Rien de ce qui est derrière un cube et devant un cube n'est grand.
 9. Tout objet qui n'a rien derrière lui est un cube.
 10. Tout dodécaèdre est plus petit qu'un tétraèdre.
2. Chargez le fichier `Pierce.wld`. Vérifiez que toutes vos formules sont vraies.
3. Chargez le fichier `Claire.wld`. Vérifiez que seules les formules 3, 5, 7, 9 sont vraies (sinon il y a une erreur dans vos formules que vous corrigerez).
4. Chargez le fichier `Leibniz.wld`. Vérifiez que seules les formules 5, 6, 8, 10 sont vraies (sinon il y a une erreur dans vos formules que vous corrigerez).
5. Chargez le fichier `Ron.wld`. Vérifiez que seules les formules 2, 3, 4, 5, 8 sont vraies.

Exercice 13

1. Créez un fichier `exo13.sen` où vous mettrez des formules traduisant les assertions suivantes :
 1. Tout tétraèdre est devant tout dodécaèdre.
 2. Aucun dodécaèdre n'a quelque chose derrière lui.
 3. Aucun tétraèdre n'a la même taille qu'un cube.
 4. Tout dodécaèdre a la même taille qu'un cube.

5. 5. Tout objet entre deux tétraèdres est un petit cube.
 6. 6. Tout cube est entre deux objets.
 7. 7. Tout cube qui a quelque chose derrière lui est petit.
 8. 8. Tout dodécaèdre qui n'a rien à sa droite est petit.
 9. 9. Tout dodécaèdre qui n'a rien à sa droite a quelque chose à sa gauche.
 10. 10. Tout dodécaèdre à gauche d'un cube est grand.
2. Chargez le fichier `Bolzano.wld`. Vérifiez que toutes vos formules sont vraies.
 3. Chargez le fichier `Ron.wld`. Vérifiez que seules les formules 4, 5, 8, 9, 10 sont vraies (sinon il y a une erreur dans vos formules que vous corrigerez).
 4. Chargez le fichier `Claire.wld`. Vérifiez que seules les formules 1, 3, 5, 7, 9, 10 sont vraies (sinon il y a une erreur dans vos formules que vous corrigerez).
 5. Chargez le fichier `Peano.wld`. Vérifiez que seules les formules 8 et 9 sont vraies (sinon il y a une erreur dans vos formules que vous corrigerez).

Exercice 14

1. Créer un fichier `exo14.sen` qui contienne des formules traduisant les propriétés suivantes :
 1. Seuls les grands objets n'ont rien devant eux.
 2. Si un cube a quelque chose devant lui, alors il est petit
 3. Tout cube qui est derrière un dodécaèdre est aussi plus petit que ce dodécaèdre.
 4. Si e est entre deux objets, alors ils sont tous les deux petits.
 5. Si un tétraèdre est entre deux objets, alors ils sont tous les deux petits.
 6. Tout dodécaèdre est au moins aussi grand que tout cube.
 7. Si un cube est à droite d'un dodécaèdre mais pas derrière lui, alors il est aussi grand que le dodécaèdre.
 8. Aucun cube n'ayant rien à sa gauche n'est entre deux cubes.
 9. Les seuls grands cubes sont b et c .
 10. Au plus b et c sont des grands cubes.
2. Chargez le fichier `Ron.wld`. Vérifiez que toutes vos formules sont vraies (attention, il y a des objets cachés).
3. Chargez le fichier `Bolzano.wld`. Vérifiez que seules les formules 3, 8, 10 sont vraies (sinon il y a une erreur dans vos formules que vous corrigerez).
4. Chargez le fichier `Wittgenstein.wld`. Vérifiez que seules les formules 5, 7, 8 sont vraies.

Exercice 15

Ouvrez `Bozo2.sen` et `Leibniz.wld`. La plupart des expressions de ce fichier ne sont pas des formules bien formées ; quelques-unes sont des formules bien formées mais ne sont pas des formules closes (sentence). Lisez et examinez chaque expression, si ce n'est pas une formule bien formée, corrigez-la. Si ce n'est pas une sentence faites-en une sentence vraie, et ce en apportant le moins possible de changements. Si c'est une sentence fautive, essayez de la rendre vraie, ici aussi en apportant le moins possible de changements. Voyez si vous pouvez reconstituer ce qu'on cherchait initialement à dire. Enregistrez votre ensemble de sentences sous `exo15.sen`.

Exercice 16

1. Créer un fichier `quant1.sen` qui contienne des formules traduisant les propriétés suivantes : (on interprète la phrase le cube est grand par : il y a exactement un cube, et il est grand ; cette phrase sera supposée fautive si par exemple il n'y a pas de cube, ou s'il y en a deux)
 1. Il y a au moins deux dodécaèdres.
 2. Il y a au plus deux tétraèdres.
 3. Il y a exactement deux cubes.
 4. Il y a seulement trois objets qui ne sont pas petits.
 5. Le petit tétraèdre n'a rien devant lui.

6. Le tétraèdre qui a quelque chose devant lui est grand.
 7. Aucun dodécaèdre n'est derrière le grand cube.
 8. Le cube moyen est à droite du grand cube.
 9. Le seul objet n'ayant rien à sa droite est le cube moyen.
 10. Le plus petit cube est moyen.
2. Chargez le fichier `Peano.wld`. Vérifiez que toutes vos formules sont vraies.
 3. Chargez le fichier `Bolzano.wld`. Vérifiez que seules les formules 1, 3, 7 sont vraies (sinon il y a une erreur dans vos formules que vous corrigerez).
 4. Chargez le fichier `Skolem.wld`. Vérifiez que seules les formules 5, 7 sont vraies.
 5. Chargez le fichier `Montague.wld`. Vérifiez que seules les formules 2, 3, 5, 7, 10 sont vraies.

Dans les deux exercices suivants, nous dirons qu'une formule est T-valide si toute interprétation où les prédicats `Cube`, ..., `Between` sont fixés comme dans Tarski's world est un modèle de cette formule. Noter qu'on ne fixe pas l'interprétation des six constantes.

Exercice 17

1. Chargez les fichiers `Carnap.sen` et `Bolzano.wld`. Traduisez les formules en français et vérifiez qu'elles sont vraies dans `Bolzano.wld`.
2. Quelles sont les formules qui sont T-valides ? (Il y en a cinq). Pour chaque formule qui n'est pas T-valide construire un monde qui falsifie la formule.
3. Quelle formule est une tautologie ?
4. Pour chaque formule qui est T-valide mais qui n'est pas une tautologie, dites comment changer l'interprétation des prédicats pour falsifier la formule.

Exercice 18

Chargez le fichier `Post.sen`.

1. Quelles sont les formules qui sont T-valides ?
2. Pour chaque formule qui n'est pas T-valide essayez de construire un monde qui falsifie la formule.
3. Y a-t-il des formules non T-valides mais pour lesquelles vous ne pouvez pas construire de contre-exemple à l'aide de Tarski's world à cause des contraintes sur les mondes de Tarski's world ?

Exercice 19

Traduire l'ensemble d'énoncés suivants en formules et vérifiez que l'ensemble de formules obtenu est satisfaisable en construisant un monde `monde.wld` qui en est un modèle.

1. Tout cube est à gauche de tout tétraèdre.
2. Il n'y a pas de dodécaèdre.
3. Il y a exactement quatre cubes.
4. Il y a exactement quatre tétraèdres.
5. Aucun tétraèdre n'est grand.
6. Aucun objet n'est plus grand que tous les objets à sa droite.
7. Un objet est à gauche d'un autre objet uniquement si le second est derrière le premier.

Exercice 20

1. Chargez le fichier `Padoa.sen`. Il contient quatre formules, trois quelconques d'entre elles sont satisfaisables, mais pas les quatre. Construire quatre mondes `123.wld`, `124.wld`, `134.wld`, `234.wld` qui sont modèles des sous-ensembles de trois formules.
2. Réinterprétez les prédicats `Tet` et `Dodec` pour que la formule 3 devienne vraie dans `124.wld`.
3. Réinterprétez le prédicat `Between` pour que la formule 4 devienne vraie dans `123.wld`.

Exercice 21

Cet exercice explore les limites posées par le nombre de variables qui est de six dans Tarski's world.

1. Traduire *Il y a au moins deux objets*. Combien de variables avez-vous utilisées ?
2. Traduire *Il y a au moins trois objets*. Combien de variables avez-vous utilisées ? On peut montrer qu'on ne peut pas exprimer *Il y a au moins sept objets* en n'utilisant que les six variables et le symbole $=$.
3. Chargez les fichiers `Robinson.sen` et `Robinson.wld`. Il y a six cubes, vérifiez la validité de la formule. Ajoutez un petit cube à droite des autres et revérifiez. Jouez le jeu en prétendant (faussetment) que la formule est fausse. Observez la stratégie de Tarski pour choisir un objet pour la variable x . Pouvez-vous en déduire comment on a réutilisé les variables ?
4. Supprimez un des cubes, et rejouez en prétendant que la formule est vraie. Voyez-vous pourquoi on ne peut pas gagner ?
5. Dans la question 2, on a dit qu'il était impossible d'exprimer l'existence de 7 objets avec 6 variables ; ici on exprime l'existence de 7 objets avec 2 variables. Pouvez-vous expliquer ce paradoxe apparent ? (Indication : faites tourner la figure de 90° et réévaluez la formule de Robinson. Reste-t-elle vraie ?)
6. Écrivez une formule qui dit qu'il y a au moins 4 objets, les uns devant les autres, en utilisant uniquement les variables x et y .

Exercice 22

On définit un ordre de complexité sur les figures en combinant lexicographiquement les deux ordres suivants :

- une figure a est plus complexe qu'une figure b si a contient plus de faces que b ;
- une figure a est plus complexe qu'une figure b si a est plus grande que b .

Ainsi une pyramide est moins complexe qu'un cube, un petit cube est moins complexe qu'un grand cube et une grande pyramide est moins complexe qu'un cube moyen, qui est lui-même moins complexe qu'un petit dodécaèdre.

1. Construire un monde contenant 6 figures rangées de gauche à droite par ordre croissant de complexité.
2. Saisir un ensemble de formules décrivant la propriété de rangement de ce monde.
3. Vérifier.

Exercice 23

1. Charger l'ensemble de formules du fichier `austin.sen`
2. Réaliser un monde qui satisfasse toutes les formules

Exercice 24

1. Charger le fichier `montagne.sen`
2. Compléter les formules
3. Réaliser un monde pour les formules obtenues

Exercice 25

1. Créer un monde contenant au moins une figure de chaque espèce.
2. Saisir les deux formules

$$\forall x(Tet(x) \vee Cube(x) \vee Dodec(x))$$
$$\forall x(Tet(x) \longrightarrow (Cube(x) \longrightarrow Dodec(x)))$$

3. Vérifier automatiquement puis interactivement (bouton `Game`) la validité de ces formules.

Variante : construire un monde avec une seule figure qui valide la formule $\forall x(Tet(x) \longrightarrow (Cube(x) \longrightarrow Dodec(x)))$. Est-ce le seul monde validant cette formule ?

Exercice 26

1. Saisir la formule

$$\forall x((Tet(x) \vee Cube(x)) \longrightarrow Dodec(x))$$

2. Construire un monde à une seule figure validant cette formule.
3. Vérifier interactivement la validité.

Exercice 27

1. Charger le fichier `morecnf.sen`
2. Donner les formes normales conjonctives de chacune des formules
3. Réaliser un monde pour les formules obtenues

Exercice 28 – Existence et unicité

Charger le fichier `russel.sen`. Réaliser un monde pour les formules obtenues.

Exercice 29

Charger le fichier `skolem.sen`. Réaliser un monde pour les formules obtenues.

Exercice 30

Charger le fichier `post.sen`. Réaliser un monde pour les formules obtenues.

Exercice 31 – Paradoxe apparent

Charger le fichier `dodgson.sen`. Réaliser un monde pour les formules obtenues.

Exercice 32

Charger le fichier `bozo.sen`.

1. Dire (et vérifier) quelles formules sont correctes et lesquelles ne le sont pas.
2. Corriger les formules incorrectes.
3. Réaliser (si possible) un monde pour les formules obtenues.

Outils mathématiques pour l'informatique

Merci à ma collègue Irène Guessarian pour ses documents

Merci à Benjamin Baron, étudiant 2009-2010 S1, pour ses notes de cours

Table des matières

1	Introduction : terminaison	3
1.1	Conjecture de Syracuse	3
1.2	Un modèle de calcul très général : les machines à deux compteurs	4
2	Ensembles et relations	6
2.1	Notions de base	6
2.2	Relations	6
2.3	Relations d'équivalences	7
2.4	Fonctions	7
2.5	Monoïdes	8
3	Induction sur \mathbb{N}	8
3.1	Premier principe d'induction sur \mathbb{N}	8
3.2	Deuxième principe d'induction : récurrence complète sur \mathbb{N}	9
3.3	Fonctions définies par récurrence	10
4	Induction structurelle	11
4.1	Ensembles définis par induction	11
4.2	Fonctions définies par induction structurelle	12
4.3	Preuves par induction structurelle	14
5	Relations d'ordre	15
5.1	Définitions	15
5.2	Ordres bien fondés	16
5.3	Exemples et contre-exemples	17
6	Systèmes de transitions et automates finis	18
6.1	Motivations et exemples	18
6.2	Définitions	18
6.3	Langages non reconnaissables	19
7	Langages reconnaissables	20
7.1	Union, intersection	20
7.2	Complémentaire et déterminisation	20
7.3	Produit, étoile	21
7.4	Langages rationnels et théorème de Kleene	22
7.5	Terminaison	23
7.6	Minimisation	24

8 Fonctions booléennes	27
8.1 Algèbre de Boole	27
8.2 Fonctions booléennes	27
8.3 Formes normales	28
9 Calcul propositionnel	29
9.1 Syntaxe	29
9.2 Sémantique	29
9.3 Equivalence sémantique	31
9.4 Conséquence sémantique	31
9.5 Conséquence logique (ou déduction)	33
10 Logique du premier ordre	34
10.1 Syntaxe	34
10.2 Variables libres et liées	35
10.3 Sémantique	36
10.4 Propriétés sémantiques	38

1 Introduction : terminaison

1.1 Conjecture de Syracuse

Cette conjecture, appelée aussi conjecture de Collatz, a été proposée en 1928, par Lothar Collatz.

Version programme. On considère le programme suivant :

```
while (k > 1) {  
    if (k % 2 == 1)  
        k = 3 * k + 1; // Si k impair  
    else  
        k = k / 2; // Si k pair  
}
```

qui peut aussi être écrit sous la forme d'une fonction récursive :

```
collatz(k) { //prend un entier en paramètre et retourne un entier  
    if (k > 1) {  
        if (k % 2 == 1)  
            return collatz(3 * k + 1); // Si k impair  
        else  
            return collatz(k / 2); // Si k pair  
    }  
    else return 1;  
}
```

On a par exemple :

pour $k = 4$: $4 \rightarrow 2 \rightarrow 1$
pour $k = 33$: $33 \rightarrow 100 \rightarrow 50 \rightarrow 25 \rightarrow 76 \rightarrow 38 \rightarrow 19 \rightarrow 58 \rightarrow 29 \rightarrow 88 \rightarrow 44 \rightarrow 22 \rightarrow 11 \rightarrow 34 \rightarrow 17 \rightarrow 52 \rightarrow 26 \rightarrow 13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

La conjecture dit que ces programmes s'arrêtent pour toute valeur de $k \in \mathbb{N}$, $k > 1$. Cette conjecture a été vérifiée pour tout $k < 2^{62}$, mais il n'y a pas de démonstration pour le cas général.

En fait, l'arrêt d'un programme est indécidable : il n'existe pas d'algorithme qui, étant donné un programme, retourne **true** si ce programme s'arrête et **false** sinon.

Version suite. Soit $(u_n)_{n \in \mathbb{N}}$ la suite définie inductivement par :

$$\begin{aligned} u_0 &= k \text{ pour } k \in \mathbb{N}, k > 1 \\ u_{n+1} &= \begin{cases} u_n/2 & \text{si } u_n \text{ pair} \\ 3u_n + 1 & \text{si } u_n \text{ impair} \end{cases} \end{aligned}$$

Dans ce cas, la question peut être reformulée de la façon suivante : pour tout $k > 1$, existe-t-il $n \in \mathbb{N}^*$ tel que $u_n = 1$?

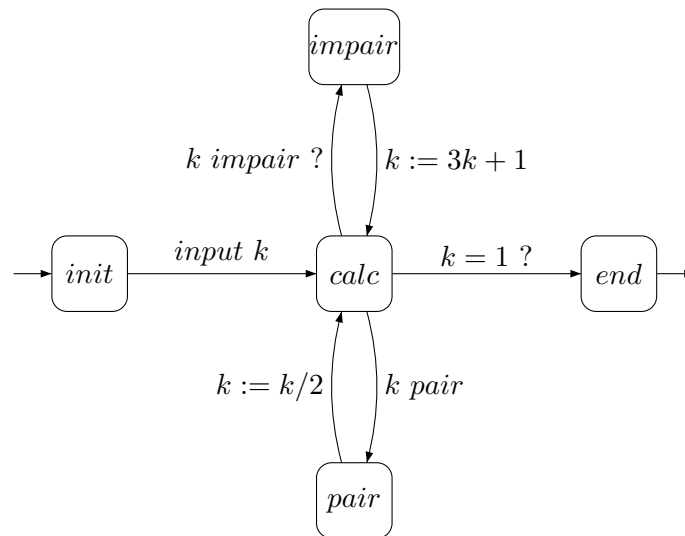
Version fonction. Soit $f: \mathbb{N} \setminus \{0\} \longrightarrow \mathbb{N}$ une fonction définie par :

$$f(k) = \begin{cases} k/2 & \text{si } k \text{ pair} \\ 3k + 1 & \text{si } k \text{ impair} \\ 1 & \text{si } k = 1 \end{cases}$$

La question est alors de savoir si pour tout $k > 1$, il existe $n \in \mathbb{N}$ tel que

$$f^n(k) = \underbrace{f \circ f \circ \dots \circ f}_n(k) = 1.$$

Version machine à un registre.



Dans ce modèle, la question est de savoir si pour tout $k > 1$ fourni en entrée, l'état *end* est atteint.

Conclusion. Pour montrer la terminaison d'un programme avec une boucle, on peut associer au corps de cette boucle une valeur dans un ensemble muni d'une relation d'ordre et montrer que la suite de ces valeurs est stationnaire (constante à partir d'un certain rang). Plusieurs outils mathématiques sont utiles pour cela :

- l'induction (dont un cas particulier est la récurrence), qui permet de déduire les valeurs successives,
- les théorèmes de points fixes, qui donnent des conditions pour que des suites soient stationnaires,
- les ordres bien fondés, qui donnent aussi de telles conditions,
- les automates finis, qui sont un modèle de calcul suffisamment simple pour qu'on puisse décider plusieurs questions.

1.2 Un modèle de calcul très général : les machines à deux compteurs

Les machines à registres sont des modèles de calcul qui comportent un ensemble fini de registres manipulés par des opérations. Un compteur est un registre particulier qui peut

contenir seulement une valeur entière et, à part l'initialisation, admet comme seules opérations l'incrémement et la décrémement avec test à 0.

Les machines à deux compteurs peuvent être vues comme des programmes d'un langage de programmation de bas niveau.

Définition 1.1 (Machines à deux compteurs).

Une machine \mathcal{M} à deux compteurs C et D est une suite finie d'instructions étiquetées. Chaque instruction peut être soit l'instruction particulière $STOP$, soit une opération sur un des deux compteurs $X \in \{C, D\}$:

1. $\ell : X := X + 1; \text{ goto } \ell'$;
2. $\ell : \text{ si } X > 0 \text{ alors } X := X - 1; \text{ goto } \ell';$
sinon goto ℓ'' ;

Exemple. Concevoir puis dessiner une machine à deux compteurs réalisant l'addition $2 + 3$.

Pour une machine \mathcal{M} , une configuration est un triplet (ℓ, n, m) où ℓ est une étiquette, et n et m sont les valeurs respectives des compteurs C et D juste avant l'instruction ℓ . Une exécution de la machine part de la configuration initiale $(\ell_0, 0, 0)$, où ℓ_0 est l'étiquette de la première instruction et les deux compteurs sont à 0. On passe ensuite d'une configuration à la suivante par une transition. Depuis (ℓ, n, m) , on obtient :

- $(\ell', n + 1, m)$ si l'instruction d'étiquette ℓ est de type 1. sur le compteur C ,
- $(\ell', n - 1, m)$ si l'instruction d'étiquette ℓ est de type 2. sur le compteur C et que $n > 0$,
ou $(\ell'', 0, m)$ si $n = 0$,
- ou des configurations similaires pour des opérations sur le compteur D .

C'est ce que nous appellerons dans la suite un système de transitions, qui définit la sémantique d'une machine à deux compteurs. Ces machines peuvent aussi être vues comme une généralisation des automates finis, vus plus loin dans ce cours, un automate fini étant une machine sans compteur.

Le problème de l'arrêt d'une machine à deux compteurs est défini par :

Donnée : une machine à deux compteurs

Question : existe-t-il une exécution qui atteint l'instruction $STOP$?

Théorème 1.2 (Minski, 1967). *Le problème de l'arrêt d'une machine à deux compteurs est indécidable.*

Cela signifie qu'il n'existe pas d'algorithme qui, prenant en entrée une machine à deux compteurs, répond à la question par oui ou par non. On retrouve dans ce cadre l'indécidabilité de la terminaison d'un programme.

2 Ensembles et relations

2.1 Notions de base

Pour un élément x d'un ensemble E , on note $x \in E$, et pour un sous-ensemble ou partie A de E , on note $A \subseteq E$. L'ensemble de toutes les parties de E est noté $\mathcal{P}(E)$.

Si E est fini de cardinal n alors $\text{card}(\mathcal{P}(E)) = 2^n$.

Différence. Si A et B sont deux parties de E , on définit $A \setminus B = \{x \in E \mid x \in A \text{ et } x \notin B\}$.

Produit cartésien. Le produit cartésien de deux ensembles E et F est l'ensemble :

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

Si E et F sont finis, le cardinal de $E \times F$ est $\text{card}(E) \times \text{card}(F)$.

Ce produit se généralise à n ensembles E_1, \dots, E_n :

$$E_1 \times \dots \times E_n = \prod_{i=1}^n E_i = \{(x_1, \dots, x_n) \mid x_1 \in E_1, \dots, x_n \in E_n\}.$$

2.2 Relations

Définition 2.1 (Relation). Une relation binaire R d'un ensemble E vers un ensemble F est un sous-ensemble de $E \times F$.

Si $(x, y) \in R$, on dit que x est en relation avec y et on note aussi xRy ou $R(x, y)$.

- Dans le cas particulier où $E = F$, on dit que R est une relation binaire sur E .
- Une relation n -aire sur un ensemble E est un sous-ensemble de $E^n = \underbrace{E \times E \times \dots \times E}_{n \text{ fois}}$.

Si R est une relation binaire de E vers F , on définit la relation inverse de R par :

$$R^{-1} = \{(y, x) \in F \times E \mid (x, y) \in R\}.$$

On note également Id_E la relation d'égalité sur un ensemble E : $\text{Id}_E = \{(x, x) \mid x \in E\}$.

Etant données deux relations $R \subseteq E \times F$ et $R' \subseteq F \times G$, leur composition est la relation RR' (parfois aussi notée $R' \circ R$) de E vers G définie par :

$$RR' = \{(x, z) \in E \times G \mid \text{il existe } y \in F \text{ tel que } (x, y) \in R \text{ et } (y, z) \in R'\}.$$

Pour une relation binaire R sur un ensemble E , on définit :

$$R^0 = \text{Id}$$

$$R^{n+1} = R R^n$$

La fermeture transitive de R est définie par $R^+ = \cup_{n \geq 1} R^n$ et la fermeture réflexive et transitive de R est $R^* = \cup_{n \geq 0} R^n$.

Définition 2.2 (Propriétés d'une relation binaire sur un ensemble E). Une relation binaire R sur un ensemble E est :

- réflexive si pour tout x de E , xRx ,
- symétrique si pour deux éléments quelconques x et y de E , si xRy alors yRx ,
- antisymétrique si pour deux éléments quelconques x et y de E , si xRy et yRx alors $x = y$.
- transitive si pour trois éléments quelconques x, y et z de E , si xRy et yRz alors xRz .

2.3 Relations d'équivalences

Définition 2.3 (Relation d'équivalence). *Une relation d'équivalence sur un ensemble E est une relation binaire réflexive, symétrique et transitive sur E .*

Si \equiv est une relation d'équivalence sur un ensemble E , on définit la classe d'un élément x de E comme l'ensemble de tous les éléments reliés à x , c'est-à-dire : $[x] = \{y \in E \mid x \equiv y\}$. Les classes d'équivalences forment une partition de E , c'est-à-dire un ensemble de parties non vides $(A_i)_{i \in I}$ deux à deux disjointes telles que $E = \cup_{i \in I} A_i$. L'ensemble des classes d'équivalences s'appelle l'ensemble quotient, noté E/\equiv . L'application de E dans E/\equiv qui à un élément x associe sa classe $[x]$ est surjective, on l'appelle la surjection canonique.

Réciproquement, à toute partition $(A_i)_{i \in I}$ d'un ensemble E , on peut associer une relation d'équivalence R définie par : xRy s'il existe un i tel que x et y sont tous deux dans A_i .

2.4 Fonctions

Définition 2.4 (Fonction). *Une fonction f d'un ensemble E dans un ensemble F est une relation binaire de E vers F telle que pour tout élément x de E , il existe au plus un élément y de F tel que $(x, y) \in f$. On dit alors dans ce cas que y est l'image de x par f et on note $f(x) = y$.*

Une application de E dans F est une fonction pour laquelle tout élément x de E a une image.

Une application f est :

- *injective si deux éléments distincts ont des images distinctes :*
 $\forall x, y \in E, f(x) = f(y) \Rightarrow x = y$
- *surjective si tout élément de F est l'image d'un élément de E :*
 $\forall y \in F, \exists x \in E \text{ tq } y = f(x)$
- *bijjective si elle est injective et surjective : tout élément y de F est l'image d'un unique élément x de E .*

On note F^E l'ensemble des applications de E dans F . En effet, une application $f : E \mapsto F$ peut être décrite par la famille $(f(x))_{x \in E}$ des images des éléments de E .

Si $E = \{x_1, \dots, x_n\}$ est fini de cardinal n , cette famille est le n -uplet $(f(x_1), \dots, f(x_n)) \in F^n$. Lorsque E et F sont finis, le cardinal de F^E est donc $\text{card}(F)^{\text{card}(E)}$.

Pour la composition des applications, on utilise souvent aussi la notation $g \circ f$ au lieu de fg , qui est une application de E dans G lorsque f est une application de E dans F et g une application de F dans G . Ainsi, $y = g \circ f(x)$ signifie $y = g[f(x)]$.

Remarque. Si A est une partie d'un ensemble E , la fonction caractéristique de A est l'application notée $\mathbf{1}_A : E \mapsto \{0, 1\}$ définie par $\mathbf{1}_A(x) = 1$ si $x \in A$ et 0 sinon.

Ainsi, si R est une relation n -aire sur un ensemble E , on peut aussi décrire R par sa fonction caractéristique $\mathbf{1}_R : E^n \mapsto \{0, 1\}$

$$\mathbf{1}_R(x_1, \dots, x_n) = \begin{cases} 1 & \text{si } (x_1, \dots, x_n) \in R \\ 0 & \text{sinon} \end{cases}$$

C'est pourquoi, pour une relation binaire R , l'écriture $R(x, y)$ au lieu de xRy ou $(x, y) \in R$ revient à identifier R avec sa fonction caractéristique : $R(x, y)$ signifie $\mathbf{1}_R(x, y) = 1$.

2.5 Monoïdes

Définition 2.5 (Monoïde). *Un monoïde est une paire (E, \star) où E est un ensemble et $\star : E \times E \mapsto E$ est une opération associative qui possède un élément neutre :*

- \star est associative : $\forall (x, y, z) \in E^3, x \star (y \star z) = (x \star y) \star z$
- il existe $e \in E$ élément neutre : $\forall x \in E, x \star e = e \star x = x$

Exemple à retenir. Soit A un ensemble fini. L'ensemble des suites d'éléments de A est un monoïde pour la concaténation, notée \cdot ou sans aucun symbole. Cet ensemble est noté A^* (c'est le monoïde libre engendré par A) et ses éléments sont appelés des *mots* sur l'alphabet A . L'élément neutre est le mot vide, qui correspond à une suite vide (aucune lettre) et qui est noté ε .

Par exemple, pour $A = \{a, b, c, \dots, z\}$, les mots de A^* sont tous les mots écrits en minuscule (ce ne sont pas forcément des mots de la langue française). La concaténation des mots $u = \text{bon}$ et $v = \text{jour}$ donne le mot $uv = \text{bonjour}$, la concaténation de $u = \text{jron}$ et $v = \text{obu}$ donne le mot $uv = \text{jronobu}$.

Pour $A = \{a, b\}$, on a : $A^* = \{\varepsilon, a, b, aa, ab, ba, bb, \dots\}$.

Pour $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, l'ensemble A^* contient toutes les écritures en base 10 des nombres entiers. Le mot 2010 est un élément de A^* .

3 Induction sur \mathbb{N}

L'induction sur \mathbb{N} est simplement la récurrence. On peut définir une suite ou une fonction sur \mathbb{N} par une relation de récurrence, par exemple :

$$\begin{cases} u_0 = 3 \\ u_{n+1} = u_n + 2 \end{cases} \quad \text{ou} \quad \begin{cases} f(0) = 5 \\ f(n+1) = 3f(n) \end{cases}$$

Pour calculer la valeur de u_n ou de $f(n)$ pour tout n , on utilise une démonstration par récurrence. Les résultats présentés dans la suite de ce chapitre fournissent diverses méthodes pour montrer qu'une propriété \mathcal{P} est vraie sur \mathbb{N} , c'est-à-dire : $\mathcal{P}(n)$ est vraie pour tout $n \in \mathbb{N}$.

3.1 Premier principe d'induction sur \mathbb{N}

Proposition 3.1. *Si $\mathcal{P}(0)$ est vraie et si pour tout $n \in \mathbb{N}, n \geq 0$, en supposant que $\mathcal{P}(n)$ est vraie, on peut montrer que $\mathcal{P}(n+1)$ est vraie, alors $\mathcal{P}(n)$ est vraie pour tout $n \in \mathbb{N}$.*

$\begin{cases} \text{Base :} & \mathcal{P}(0) \text{ vraie} \\ \text{Induction :} & \text{pour tout } n \in \mathbb{N}, \mathcal{P}(n) \text{ vraie implique } \mathcal{P}(n+1) \text{ vraie} \end{cases}$
alors pour tout $n \in \mathbb{N}, \mathcal{P}(n)$ vraie

Exemple. Soit $S_n = 1 + 2 + \dots + n$ pour tout $n \geq 0$.

$$\mathcal{P}(n) : S_n = \frac{n(n+1)}{2} \text{ pour tout } n \in \mathbb{N}$$

Montrons ce résultat par récurrence :

Base. $S_0 = 0$, donc $\mathcal{P}(0)$ est vraie.

Induction. Supposons $\mathcal{P}(n)$ vraie à un certain rang $n \in \mathbb{N}$ (ie. $2 \cdot S_n = n(n+1)$)
 Montrons alors que $\mathcal{P}(n+1)$ est vraie.

$$\begin{aligned} 2 \cdot S_{n+1} &= 2(\underbrace{1+2+\cdots+n}_{S_n} + (n+1)) \\ &= 2 \cdot S_n + 2(n+1) \\ &= n(n+1) + 2(n+1) \text{ par hypothèse de récurrence} \\ 2 \cdot S_{n+1} &= (n+1)(n+2) \end{aligned}$$

Conclusion. On a montré :

Base : $\mathcal{P}(0)$ vraie.

Induction : Pour $n \in \mathbb{N}$, en supposant $\mathcal{P}(n)$, on en a déduit $\mathcal{P}(n+1)$.

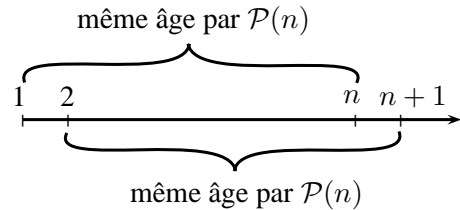
Donc \mathcal{P} est vraie sur \mathbb{N} .

Proposition 3.2 (Variante). *Une variante du résultat précédent :*

$\begin{cases} \text{Base :} & \mathcal{P}(n_0) \text{ vraie} \\ \text{Induction :} & \text{Pour tout } n \geq n_0, \mathcal{P}(n) \text{ vraie implique } \mathcal{P}(n+1) \text{ vraie} \end{cases}$
 alors pour tout $n \geq n_0$, $\mathcal{P}(n)$ est vraie.

Exemple. Soit $\mathcal{P}(n)$: les n personnes de cette salle ont le même âge.

Base $n_0 = 1$ $\mathcal{P}(1)$ vraie
 Induction pour tout $n \geq 2$ $\mathcal{P}(n)$ vraie implique $\mathcal{P}(n+1)$ vraie



L'erreur vient du fait que $\mathcal{P}(1)$ n'implique pas $\mathcal{P}(2)$

3.2 Deuxième principe d'induction : récurrence complète sur \mathbb{N}

Proposition 3.3. *Soit \mathcal{P} une propriété sur \mathbb{N} . Si*

(I') : pour tout $n \in \mathbb{N}$, [si (pour tout $k < n$, $\mathcal{P}(k)$ vraie) implique $\mathcal{P}(n)$ vraie]

alors $\mathcal{P}(n)$ est vraie pour tout $n \in \mathbb{N}$.

Proposition 3.4 (Variante). *Variante du théorème précédent pour montrer $\mathcal{P}(n)$ vraie pour tout $n \geq n_0$:*

(I') : pour tout $n \geq n_0$, [si (pour tout $k \in \mathbb{N}$, $n_0 \leq k < n$, $\mathcal{P}(k)$ vraie) implique $\mathcal{P}(n)$ vraie]

alors $\mathcal{P}(n)$ est vraie pour tout $n \geq n_0$.

Exemple. Soit $\mathcal{P}(n)$: n est décomposable en produit de facteurs premiers. Montrons que $\mathcal{P}(n)$ est vraie pour tout $n \geq 2$.

$\mathcal{P}(2)$ est vraie car 2 est premier, donc produit d'un seul facteur premier.

Soit maintenant $n > 2$. On veut montrer (I') : Si $\forall k \in \mathbb{N}, 2 \leq k < n, k$ est décomposable en produit de facteurs premiers, alors n est décomposable en produit de facteurs premiers. Il y a deux cas :

1. si n est premier, n est bien un produit d'un seul facteur premier.
2. si n n'est pas premier, alors $n = n_1 \cdot n_2$ où $n_1, n_2 > 1$ et $n_1, n_2 < n$.

Par hypothèse d'induction, on a :

- $2 \leq n_1 < n$ donc $\mathcal{P}(n_1)$ est vraie
- $2 \leq n_2 < n$ donc $\mathcal{P}(n_2)$ est vraie

Or d'après la propriété \mathcal{P} , on a :

- $\mathcal{P}(n_1)$ vraie : $n_1 = p_1 \cdots p_{i_1}$ où les p_j sont premiers
- $\mathcal{P}(n_2)$ vraie : $n_2 = q_1 \cdots q_{i_2}$ où les q_j sont premiers

De ce fait, $n = p_1 \cdots p_{i_1} \cdot q_1 \cdots q_{i_2}$ est décomposable en produit de facteurs premiers. Ainsi, $\mathcal{P}(n_1)$ et $\mathcal{P}(n_2)$ vraies $\Rightarrow \mathcal{P}(n)$ vraie. On en conclut que $\mathcal{P}(n)$ est vraie pour tout $n \geq 2$.

3.3 Fonctions définies par récurrence

Définition 3.5 (Fonction définie par récurrence). Une fonction sur \mathbb{N} est définie par récurrence par :

- Base : une valeur initiale $f(0) = a \in \mathbb{N}$
- Induction : $f(n+1) = h(n, f(n))$ pour tout $n \geq 0$

Exemple 1. Soit la fonction f définie par récurrence par :

- $$\begin{cases} \text{Base :} & f(0) = 1 \\ \text{Induction :} & f(n+1) = 2 \cdot f(n) \end{cases}$$

Prouvons par récurrence sur n que $f(n) = 2^n$.

- Base : $f(0) = 1 = 2^0$
- Induction : Supposons $f(n) = 2^n$.
Alors $f(n+1) = 2 \cdot f(n) = 2 \cdot 2^n = 2^{n+1}$ d'où le résultat.

Exemple 2. Donner l'écriture explicite de la fonction f définie par récurrence par :

- $$\begin{cases} \text{Base :} & f(0) = 1 \\ \text{Induction :} & f(n+1) = f(n) + 3 \end{cases}$$

Montrons à l'aide d'une récurrence sur n que $f(n) = 3n + 1$.

- Base : $f(0) = 1 = 3 \cdot 0 + 1$
- Induction : Supposons $f(n) = 3n + 1$.
Alors $f(n+1) = f(n) + 3 = (3n + 1) + 3 = 3(n+1) + 1$ d'où le résultat.

4 Induction structurelle

L'induction structurelle généralise la récurrence sur \mathbb{N} .

4.1 Ensembles définis par induction

Un ensemble X est défini inductivement par la donnée de :

(B) un ensemble initial X_0 qui doit être contenu dans X

(I) un ensemble de règles, sous forme d'opérations, pour construire de nouveaux éléments de X .

L'ensemble X est alors le plus petit ensemble (pour l'inclusion) satisfaisant ces conditions.

Exemple 1. Déterminer le sous-ensemble X de \mathbb{N} défini inductivement par :

(B) $1 \in X$

(I) si $x \in X$ alors $x + 2 \in X$

Exemple 2. On se donne un symbole a et une opération unaire s . Déterminer l'ensemble \mathcal{T} défini inductivement par :

(B) $a \in \mathcal{T}$

(I) si $t \in \mathcal{T}$ alors $s(t) \in \mathcal{T}$

Remarquons que l'exemple 1 est un cas particulier de l'exemple 2 avec $a = 1$ et l'opération $s : \mathbb{N} \mapsto \mathbb{N}$ définie par $s(x) = x + 2$.

Exemple 3. On considère l'alphabet $A = \{a, b\}$. Déterminer le sous-ensemble L de A^* défini inductivement par :

(B) $\varepsilon \in L$

(I) si $u \in L$ alors $aub \in L$

Exemple 4. On se donne deux symboles a et b , et une opération binaire \odot . Décrire graphiquement l'ensemble \mathcal{T} défini inductivement par :

(B) a et b sont dans \mathcal{T}

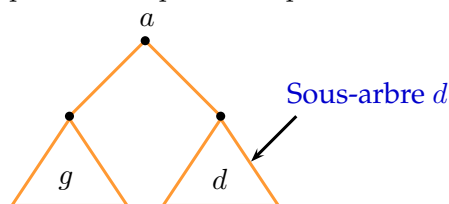
(I) si t_1 et t_2 sont dans \mathcal{T} alors $\odot(t_1, t_2) \in \mathcal{T}$

Arbres binaires. On se donne un alphabet A auquel on ajoute les 4 symboles $(,) \emptyset$ et on définit inductivement l'ensemble AB des arbres binaires sur A par :

(B) $\emptyset \in AB$

(I) si g et d sont dans AB alors pour toute lettre $a \in A$, (a, g, d) appartient à AB .

La forme générale d'un arbre binaire construit à partir d'une lettre a et de sous-arbres g et d peut être représentée par le dessin suivant :



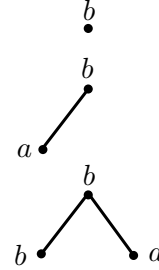
Pour $A = \{a, b\}$, on a par exemple :

\emptyset

$(b, \emptyset, \emptyset)$

$(b, (a, \emptyset, \emptyset), \emptyset)$

$(b, (b, \emptyset, \emptyset), (a, \emptyset, \emptyset))$



Termes. Les termes sont un cas particulier d'arbres, mais plus nécessairement binaires. On considère des symboles de fonctions rangés selon leur nombre d'arguments : F_0 est l'ensemble des symboles de constantes, correspondant aux fonctions sans arguments, F_1 contient les opérations unaires (à un seul argument), F_2 les opérations binaires, etc. L'ensemble des termes sur $\cup_{i \geq 0} F_i$ est l'ensemble \mathcal{T} défini inductivement par :

(B) $F_0 \subseteq \mathcal{T}$, donc toutes les constantes sont des termes

(I) si f est une fonction de F_n (à n arguments) et si t_1, \dots, t_n sont dans \mathcal{T} alors $f(t_1, \dots, t_n)$ appartient aussi à \mathcal{T} .

Remarquer que les exemples 2 et 4 ci-dessus sont directement des cas particuliers de cette définition. De plus, un terme de la forme $f(t_1, \dots, t_n)$ peut être représenté par un arbre ayant pour racine f et avec n fils t_1, \dots, t_n .

4.2 Fonctions définies par induction structurelle

Sur un ensemble X défini inductivement, on peut définir une fonction g comme suit :

- Base : $g(x)$ est donné explicitement pour tout $x \in X_0$.
- Induction : un procédé pour calculer l'image par g d'un élément construit inductivement.

Exemple 1. Soit la fonction g définie par :

(B) $g(0) = 1$

(I) $g(n+1) = (n+1) \times g(n)$

La fonction f est alors définie explicitement par $g: \mathbb{N} \mapsto \mathbb{N}^*$
 $n \mapsto n!$

Exemple 2. Pour l'exemple 1.4, donner une définition inductive de la fonction nba qui associe à tout élément $t \in \mathcal{T}$ le nombre de a apparaissant dans t .

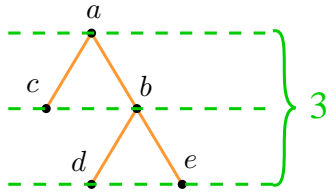
Fonctions associées à un arbre binaire.

- La **hauteur** d'un arbre binaire est définie inductivement par :

(B) $h(\emptyset) = 0$

(I) $h((a, g, d)) = 1 + \max(h(g), h(d))$

Sur l'alphabet $A = \{a, b, c, d, e\}$:



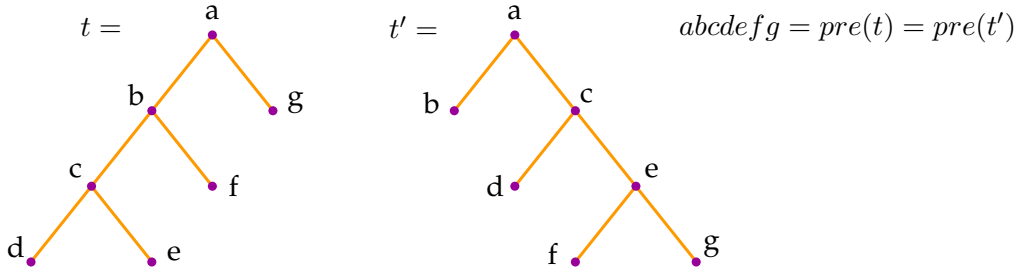
- Le **parcours préfixe** d'un arbre binaire « De haut en bas et de gauche à droite » est défini inductivement par :

$$(B) \text{ } pre(\emptyset) = \varepsilon$$

$$(I) \text{ } pre((a, g, d)) = a \text{ } pre(g)pre(d)$$

Pour l'arbre ci-dessus $pre(t) = a \text{ } c \text{ } b \text{ } d \text{ } e$.

Remarquons que deux arbres binaires différents peuvent avoir le même parcours préfixe.



Fonctions d'évaluation (ou d'interprétation) des termes.

Dans la définition générale des termes donnée précédemment, les constantes de F_0 ainsi que les autres fonctions sont seulement des symboles. Pour interpréter (ou évaluer) un ensemble de termes, il faut d'abord définir les opérations et les valeurs des constantes.

Dans l'exemple 1.4, on peut se placer dans le domaine $D = \mathbb{N}$, avec la valeur 2 pour a et la valeur 3 pour b . Ceci correspond à associer à a l'élément $a_D = 2$ de D et à b l'élément $b_D = 3$, c'est-à-dire en général à associer aux constantes de F_0 des valeurs dans D .

Il faut aussi dire ce que fait l'opération \odot , c'est le rôle d'une autre fonction d'interprétation $\odot_D : D \times D \mapsto D$. Prenons par exemple ici l'addition binaire $+$ pour \odot_D .

On peut alors calculer l'interprétation (ou la valeur) de chaque terme de $t \in \mathcal{T}$, notée $h^*(t)$, grâce à l'induction. La notation h^* provient du fait qu'on étend l'application $h : F_0 \mapsto D$ définie par $h(a) = a_D$ pour tout $a \in F_0$.

Par exemple, la valeur du terme $t_1 = \odot(a, b)$ est obtenue en appliquant \odot_D (donc $+$) aux valeurs de a et b , donc $h^*(t_1) = 2 + 3 = 5$. Ceci s'exprime formellement par : $h^*(\odot(a, b)) = \odot_D(h^*(a), h^*(b))$. De même, la valeur de $t_2 = \odot(b, b)$ est $h^*(t_2) = 3 + 3 = 6$. Pour calculer maintenant la valeur de $t = \odot(\odot(a, b), \odot(b, b))$, il suffit de remplacer les deux termes par leur valeur :

$$h^*(t) = h^*(\odot(t_1, t_2)) = h^*(t_1) + h^*(t_2) = 11.$$

La proposition suivante décrit ce procédé dans le cas général.

Proposition 4.1. *Soit \mathcal{T} l'ensemble des termes construits sur $\cup_{i \geq 0} F_i$. Soit D un ensemble, h une application de F_0 dans D qui associe à toute constante a la valeur a_D et, pour chaque fonction f de $\cup_{i \geq 0} F_i$ avec n arguments, une application $f_D : D^n \mapsto D$.*

Alors il existe une unique application $h^* : \mathcal{T} \mapsto D$ définie inductivement par :

(B) si $a \in F_0$ alors $h^*(a) = a_D$,

(I) si $t = f(t_1, \dots, t_n)$ pour une fonction f de F_n (à n arguments) et n termes t_1, \dots, t_n , alors $h^*(t) = f_D(h^*(t_1), \dots, h^*(t_n))$.

4.3 Preuves par induction structurelle

Définition 4.2. Preuve par induction d'une propriété \mathcal{P} sur un ensemble X défini par induction.

Base (B) : si $\mathcal{P}(x)$ est vraie pour tout $x \in X_0$

Induction (I) : si \mathcal{P} est préservée par les opérations de construction.

Alors $\mathcal{P}(x)$ est vraie pour tout $x \in X$.

Exemple 1.3. Montrer par induction que dans le langage L de l'exemple 1.3, tout mot possède autant de a que de b .

Exemple 1.4. Dans l'exemple 1.4, On peut montrer par induction que $h^*(t) = 2nba(t) + 3nbb(t)$ où $nba(t)$ est le nombre de a dans t (définie dans la section précédente) et de même $nbb(t)$ est le nombre de b dans t .

Induction sur \mathbb{N} . L'ensemble \mathbb{N} est défini inductivement par :

(B) : $0 \in \mathbb{N}$

(I) : si $n \in \mathbb{N}$ alors $n + 1 \in \mathbb{N}$

Pour montrer qu'une proposition $\mathcal{P}(n)$ est vraie pour tout $n \in \mathbb{N}$, on doit montrer :

(B) : $\mathcal{P}(0)$ vraie

(I) : Si $\mathcal{P}(n)$ est vraie, alors $\mathcal{P}(n + 1)$ est vraie

On retrouve donc le principe de récurrence comme un cas particulier.

5 Relations d'ordre

5.1 Définitions

Définition 5.1 (Relation d'ordre). *Une relation d'ordre (large) sur un ensemble E est une relation binaire R sur E réflexive, antisymétrique et transitive.*

A une relation d'ordre \preceq , on associe la relation notée \prec définie par : $x \prec y$ si $x \preceq y$ et $x \neq y$. Donc $\preceq = \prec \cup Id$, et aussi $\prec = \preceq \setminus Id$. La relation \prec est appelée un ordre strict.

Définition 5.2 (Ordre total et ordre partiel). *Une relation d'ordre \preceq sur un ensemble E est dite totale si deux éléments quelconques de E sont comparables par cette relation.*

Une relation d'ordre qui n'est pas totale est appelée un ordre partiel : il existe des éléments non comparables.

Définition 5.3 (Applications monotones et isomorphismes d'ensembles ordonnés). *Soient E_1 et E_2 deux ensembles ordonnés respectivement par \preceq_1 et \preceq_2 . Une application f de E_1 dans E_2 est monotone si pour toute paire (x, y) d'éléments de E_1 , si $x \preceq_1 y$ alors $f(x) \preceq_2 f(y)$. Les ensembles E_1 et E_2 sont isomorphes s'il existe une bijection f de E_1 sur E_2 telle que f et f^{-1} sont monotones.*

Exemples d'ordres à retenir.

- Sur \mathbb{N} , l'ordre naturel \leq est un ordre (large) total.
- Sur \mathbb{N} , la relation de divisibilité $|$ est définie par : $n | m$ si n divise m c'est-à-dire s'il existe un entier k tel que $m = kn$. C'est un ordre partiel.
- Sur A^* , la relation *préfixe* est définie par : $u \leq v$ si v commence par u c'est-à-dire s'il existe un mot w de A^* tel que $v = uw$. C'est un ordre partiel.
- Sur A^* , pour un ordre total donné sur l'alphabet A , l'ordre lexicographique (ou alphabétique) est un ordre total.
- Sur $\mathbb{N} \times \mathbb{N} = \mathbb{N}^2$, l'ordre produit de l'ordre naturel défini par $(a_1, b_1) \preceq (a_2, b_2)$ si $a_1 \leq a_2$ et $b_1 \leq b_2$ est un ordre partiel.
- Si E est un ensemble, l'ensemble $\mathcal{P}(E)$ des parties de E est ordonné par inclusion. C'est un ordre partiel.

Majorants, borne supérieure, plus grand élément, éléments maximaux. Les minorants, borne inférieure, plus petit élément, éléments minimaux sont obtenus en considérant les ordres opposés.

Définition 5.4. *Soit (E, \preceq) un ensemble muni d'une relation d'ordre et A une partie de E .*

- *Un élément M de E est un majorant de A si $\forall a \in A, a \preceq M$.
On note $Maj(A) = \{M \in E \mid \forall a \in A, a \preceq M\}$ l'ensemble des majorants de A dans E .*
- *Le plus grand élément de A , s'il existe, est l'unique élément M de $A \cap Maj(A)$ (c'est-à-dire un majorant de A dans A).*
- *La borne supérieure S de A dans E , si elle existe, est le plus petit élément de $Maj(A)$.*
- *Un élément N de A est un élément maximal de A si aucun élément de A n'est plus grand que lui : il appartient à A et $\forall a \in A, N \preceq a \Rightarrow N = a$.*

5.2 Ordres bien fondés

Soit (E, \preceq) un ensemble ordonné. Une suite $(a_n)_{n \in \mathbb{N}}$ d'éléments de E est décroissante si pour tout $n \in \mathbb{N}$, $a_{n+1} \preceq a_n$. Elle est strictement décroissante si pour tout $n \in \mathbb{N}$, $a_{n+1} \prec a_n$, c'est-à-dire $a_{n+1} \preceq a_n$ et $a_{n+1} \neq a_n$.

Définition 5.5 (Ordre bien fondé). *Un ordre \preceq sur un ensemble E est bien fondé s'il n'existe pas dans E de suite infinie strictement décroissante pour \preceq .*

Ainsi, si l'on peut associer au corps d'une boucle une valeur qui décroît strictement à chaque tour de la boucle (ou suffisamment régulièrement), on construit une suite strictement décroissante. Si cette suite est à valeurs dans un ensemble ordonné (E, \preceq) tel que l'ordre \preceq est bien fondé, on obtient une preuve de terminaison.

Exemples. L'ordre naturel est bien fondé sur l'ensemble \mathbb{N} , mais pas sur l'ensemble \mathbb{Z} .

Proposition 5.6. *Un ordre \preceq est bien fondé sur un ensemble E si et seulement si toute partie non vide de E a (au moins) un élément minimal pour \preceq .*

Démonstration. On montre l'équivalence des négations.

- Supposons d'abord qu'il existe une suite $(a_n)_{n \in \mathbb{N}}$ strictement décroissante, alors la partie $A = \{a_n \mid n \in \mathbb{N}\}$ n'a pas d'élément minimal.

- Réciproquement, soit $A \subseteq E$ une partie non vide de E sans élément minimal. Alors pour tout $x \in A$, x n'est pas minimal, donc il existe $y \in A$ tel que $y \prec x$.

Soit $f : A \mapsto A$

$$x \mapsto y \text{ tel que } y \prec x$$

On définit la suite $(a_n)_{n \in \mathbb{N}}$ par : $a_0 \in A$ est un élément quelconque de A (possible car A non vide) et $a_{n+1} = f(a_n)$ pour tout $n \in \mathbb{N}$. Cette suite est strictement décroissante, par construction. \square

Proposition 5.7 (Corollaire : **principe d'induction**). *Soit (E, \preceq) un ensemble muni d'un ordre bien fondé, et soit \mathcal{P} une propriété des éléments de E . On a alors la propriété suivante d'induction :*

Si (I) : pour tout élément x de E , on a : si (pour tout $y \prec x$, $\mathcal{P}(y)$ vraie) alors $\mathcal{P}(x)$ vraie, alors \mathcal{P} est vraie pour tout élément de E .

Démonstration. Soit $A = \{x \in E \mid \mathcal{P}(x) \text{ fautive}\}$ et supposons (I) vraie pour \mathcal{P} et E . Nous allons montrer que $A = \emptyset$.

Par l'absurde, supposons que A est non vide. D'après la proposition précédente, il existe un élément minimal $a_0 \in A$ donc :

- $\mathcal{P}(a_0)$ est fautive et
- a_0 minimal dans A , donc pour tout $y \prec a_0$, $y \notin A$, d'où $\mathcal{P}(y)$ est vraie.

Mais par (I), $\mathcal{P}(a_0)$ est vraie ce qui est une contradiction. De ce fait, $A = \emptyset$. \square

5.3 Exemples et contre-exemples

Ordres usuels.

1. L'ordre naturel sur \mathbb{N} est bien fondé (avec 0 comme plus petit élément).
2. Remarquons que la présence d'un plus petit élément ne suffit pas en général : pour \mathbb{R}_+ , l'ensemble des nombres réels positifs ou nuls, muni également de l'ordre usuel, 0 est aussi un plus petit élément mais il existe une suite (infinie) strictement décroissante : $u_n = \frac{1}{n}$ pour $n \in \mathbb{N}$, $n \geq 1$.
3. L'ensemble \mathbb{Z} n'est pas un ensemble bien fondé pour l'ordre usuel. En effet, il existe une suite strictement décroissante $(x_n)_{n \in \mathbb{N}}$ définie par :
 - $x_0 = k$, $k \in \mathbb{Z}$
 - $x_{n+1} = x_n - 1$

Divisibilité. La divisibilité est un ordre bien fondé sur \mathbb{N} .

Ordres sur le monoïde libre A^* .

1. L'ordre lexicographique n'est pas un ordre bien fondé car $(a^n b)_{n \in \mathbb{N}}$ est une suite strictement décroissante : $b \succ ab \succ a^2 b \dots a^n b \succ a^{n+1} b \succ \dots$. Remarquons que pourtant cet ordre admet un plus petit élément ε .
2. L'ordre préfixe est un ordre bien fondé car une suite strictement décroissante est formée de mots dont la longueur décroît strictement (dans \mathbb{N}).

Ordres produits. Considérons $\mathbb{N} \times \mathbb{N}$. Il y a (entre autres) deux possibilités :

1. L'ordre produit de l'ordre naturel est un ordre partiel bien fondé.
2. L'ordre lexicographique (strict) sur $\mathbb{N} \times \mathbb{N}$ est défini par :
 $(a_1, b_1) < (a_2, b_2)$ si $a_1 < a_2$ ou $(a_1 = a_2 \text{ et } b_1 < b_2)$.
L'ordre large associé est obtenu naturellement par :
 $(a_1, b_1) \leq (a_2, b_2)$ si $(a_1, b_1) < (a_2, b_2)$ ou $(a_1 = b_1 \text{ et } a_2 = b_2)$.
C'est aussi un ordre bien fondé.

Pour deux ensembles ordonnés E et F , on peut de manière analogue définir différents ordres sur le produit $E \times F$.

6 Systèmes de transitions et automates finis

6.1 Motivations et exemples

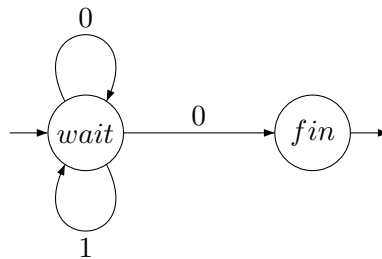
Les automates finis décrivent une classe restreinte de « programmes » dans laquelle les questions sont faciles à traiter.

D'une façon générale, un des objectifs de l'informatique est de formaliser des problèmes et de construire, lorsque c'est possible, des programmes pour les résoudre. Un problème est formalisé comme un langage sur un alphabet (*i.e.* un ensemble de mots).

A un problème P , on associe un langage L_P sur un alphabet A , c'est-à-dire un sous-ensemble de A^* , avec la propriété suivante : un mot w de A^* est solution de P ssi il appartient à L_P .

Exemple. Sur l'alphabet $A = \{0, 1\}$, un mot représente l'écriture binaire d'un entier. Cet entier est pair si le mot qui lui correspond se termine par 0. Ainsi, le problème « Pair » peut être associé au langage $L_{Pair} = A^*0$ qui contient exactement tous les mots se terminant par un 0.

Pour décrire ces solutions de manière opérationnelle (par un programme), l'étape suivante est de chercher s'il est possible de construire un automate fini acceptant le langage L_P . Pour l'exemple des écritures en binaire des entiers pairs :



Autres exemples : digicode, traversée de la rivière par un loup, une chèvre et une salade, barman aveugle.

6.2 Définitions

Définition 6.1. Un système de transitions sur un alphabet A est un triplet $\mathcal{T} = (S, T, I)$ où S est l'ensemble des configurations, $T \subseteq S \times A \times S$ est l'ensemble des transitions et I est un sous-ensemble de S contenant les configurations initiales.

On note $s \xrightarrow{a} s'$ une transition (s, a, s') de T . La lettre a est l'*étiquette* de la transition. Une exécution de \mathcal{T} est un chemin dans ce graphe étiqueté, c'est-à-dire une suite de transitions $(s_0, a_1, s_1)(s_1, a_2, s_2) \dots$ qui s'écrit aussi $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \dots$, commençant dans une configuration initiale ($s_0 \in I$). Par extension, le mot $w = a_1 a_2 \dots$ est l'*étiquette* (ou *trace*) du chemin (ou de l'exécution).

Exemple. Dessiner un système de transitions \mathcal{T} représentant le fonctionnement d'un tampon de taille non bornée. L'alphabet $A = \{p, c\}$ décrit les deux opérations possibles sur ce tampon : p est une opération de production qui ajoute un élément au tampon, tandis que c est une opération de consommation qui retire un élément du tampon.

Décrire le langage L_{pc} des mots sur l'alphabet A qui étiquettent des exécutions de \mathcal{T} .

Donner aussi une machine à un compteur acceptant L_{pc} .

Définition 6.2. Un automate fini est un système de transitions

- dont l'ensemble des configurations est fini (les éléments sont alors appelés des états),
- auquel on adjoit un sous-ensemble $F \subseteq S$ d'états finals.

Si $\mathcal{A} = (S, T, I, F)$ est un automate fini sur A , on dit qu'un mot w de A^* est accepté par \mathcal{A} s'il existe une exécution finie $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \dots \xrightarrow{a_n} s_n$ commençant dans un état initial ($s_0 \in I$), se terminant dans un état final ($s_n \in F$) et ayant w comme étiquette. Le langage de \mathcal{A} , noté $\mathcal{L}(\mathcal{A})$, est l'ensemble des mots acceptés par \mathcal{A} .

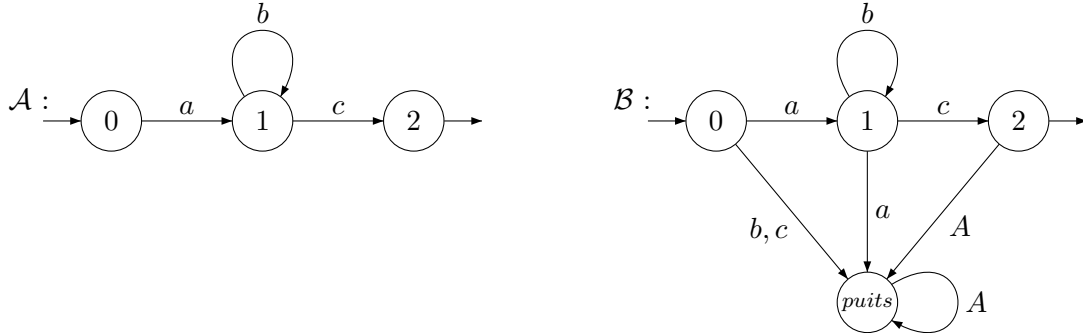
Définition 6.3. Soit A un alphabet. Un langage L de A^* est reconnaissable s'il existe un automate fini \mathcal{A} tel que $L = \mathcal{L}(\mathcal{A})$. Deux automates finis sont équivalents s'ils acceptent le même langage.

Définition 6.4. Un automate fini $\mathcal{A} = (S, T, I, F)$ est complet si pour tout état $s \in S$ et pour toute lettre $a \in A$, il existe (au moins) une transition partant de s avec l'étiquette a .

Partant d'un automate fini $\mathcal{A} = (S, T, I, F)$, on construit facilement un automate complet $\mathcal{B} = (S \cup \{p\}, I, T \cup T', F)$ équivalent à \mathcal{A} (donc acceptant le même langage). L'état p appelé *puits* est un nouvel état ajouté à S et T' contient les transitions suivantes :

- pour tout état s et toute lettre a tels qu'il n'existe pas de transition d'étiquette a sortant de s , la transition $s \xrightarrow{a} p$ est dans T' ,
- les transitions $p \xrightarrow{a} p$, pour toute lettre $a \in A$, sont dans T' .

Exemple. Soit l'automate \mathcal{A} suivant sur l'alphabet $A = \{a, b, c\}$ qui accepte le langage $L = ab^*c$. On le complète en un automate équivalent \mathcal{B} , c'est-à-dire **sans changer le langage accepté** :



6.3 Langages non reconnaissables

Il ne faut pas croire que tout langage est reconnaissable. Par exemple :

Proposition 6.5. Il n'existe pas d'automate fini acceptant le langage $L = \{a^n b^p, n \geq p\}$ sur l'alphabet $A = \{a, b\}$.

Démonstration. Par l'absurde, on suppose qu'il existe un automate fini $\mathcal{A} = (S, T, I, F)$ acceptant L et on note N le nombre d'états de cet automate. On considère alors le mot $w = a^{N+1}b^{N+1}$ de L , et un chemin associé dans l'automate : $s_0 \xrightarrow{a^{N+1}} s_1 \xrightarrow{b} s_2 \dots \xrightarrow{b} s_{N+2}$

où s_0 est un état initial et s_{N+2} un état final. Parmi les $N + 2$ états s_1, \dots, s_{N+2} , il y en a deux égaux : $s_i = s_j$ pour $i < j$, ce qui correspond à une boucle dans l'automate. Donc on peut réécrire le chemin acceptant w en : $s_0 \xrightarrow{a^{N+1}} s_1 \xrightarrow{b^{p_1}} s_i \xrightarrow{b^{p_2}} s_i \xrightarrow{b^{p_3}} s_{N+2}$. Le mot b^{p_2} , qui est l'étiquette de la boucle, peut être répété un nombre arbitraire de fois, ce qui produit des mots acceptés par l'automate, mais qui n'appartiennent pas à L puisqu'ils ont un nombre de b strictement supérieur au nombre de a . Par exemple $w' = a^{N+1}b^{p_1+2p_2+p_3} = a^{N+1}b^{N+1+p_2}$. On obtient donc une contradiction. \square

Remarque 1. En transformant a en p et b en c , le langage L est un sous-ensemble de L_{pc} , l'ensemble des mots associés au système des productions/consommations avec un tampon non borné. On se rappelle que ce langage pouvait être décrit par un système de transition avec un nombre infini d'états (ou par un automate avec un compteur). En utilisant le résultat ci-dessus, ainsi que des propriétés établies dans le chapitre suivant, on peut aussi montrer qu'il n'existe pas d'automate fini acceptant L_{pc} .

Remarque 2. Ce raisonnement se généralise en une condition nécessaire pour qu'un langage soit reconnaissable, appelée lemme d'itération ou lemme de l'étoile.

7 Langages reconnaissables

Rappelons qu'un langage sur un alphabet A est un sous-ensemble de A^* et qu'un langage L est reconnaissable s'il est accepté par un automate fini. Ce chapitre examine les propriétés de clôture des langages reconnaissables et donne une condition nécessaire et suffisante pour qu'un langage soit reconnaissable.

7.1 Union, intersection

Proposition 7.1. Si L_1 et L_2 sont deux langages reconnaissables alors $L_1 \cup L_2$ et $L_1 \cap L_2$ sont aussi reconnaissables.

On considère deux automates finis $\mathcal{A}_1 = (S_1, T_1, I_1, F_1)$ et $\mathcal{A}_2 = (S_2, T_2, I_2, F_2)$ et on note $L_1 = \mathcal{L}(\mathcal{A}_1)$ et $L_2 = \mathcal{L}(\mathcal{A}_2)$ les langages acceptés respectivement par ces deux automates.

- **En supposant les ensembles d'états S_1 et S_2 disjoints**, un automate fini acceptant $L_1 \cup L_2$ est défini par $\mathcal{A} = (S_1 \cup S_2, T_1 \cup T_2, I_1 \cup I_2, F_1 \cup F_2)$. Autrement dit, il suffit de considérer ces deux automates comme un seul, et il n'y a pas d'ambiguïté sur les transitions du fait que $S_1 \cap S_2 = \emptyset$.

- Pour l'intersection, on réalise une sorte de synchronisation entre les deux automates, en simulant des exécutions parallèles dans \mathcal{A}_1 et \mathcal{A}_2 . Un automate fini acceptant $L_1 \cap L_2$ est défini par $\mathcal{B} = (S_1 \times S_2, T, I_1 \times I_2, F_1 \times F_2)$, les transitions de T étant obtenues pour toute lettre $a \in A$ par :

$$(s_1, s_2) \xrightarrow{a} (s'_1, s'_2) \text{ est dans } T \text{ ssi } s_1 \xrightarrow{a} s'_1 \text{ est dans } T_1 \text{ et } s_2 \xrightarrow{a} s'_2 \text{ est dans } T_2$$

7.2 Complémentaire et déterminisation

Définition 7.2. Un automate fini $\mathcal{A} = (S, T, I, F)$ sur A est déterministe si :

- il a un unique état initial : $I = \{s_0\}$,

- pour tout état s et toute lettre a , il existe au plus une transition d'étiquette a sortant de s , c'est-à-dire : si $s \xrightarrow{a} s_1$ et $s \xrightarrow{a} s_2$ alors $s_1 = s_2$.

On a alors :

Proposition 7.3. *Pour tout automate fini \mathcal{A} , on peut construire un automate fini déterministe \mathcal{D} qui accepte le même langage que \mathcal{A} (donc les deux automates \mathcal{A} et \mathcal{D} sont équivalents).*

- Soit $\mathcal{A} = (S, T, I, F)$, l'automate \mathcal{D} est défini par $\mathcal{D} = (\mathcal{P}(S), \hat{T}, \{I\}, \hat{F})$ avec :
- l'ensemble des états de \mathcal{D} est l'ensemble $\mathcal{P}(S)$ des parties de S ,
 - l'unique état initial de \mathcal{D} est la partie I ,
 - $\hat{F} = \{P \subseteq S \mid P \cap F \neq \emptyset\}$ (une partie est finale si elle contient au moins un état final de \mathcal{A}),
 - \hat{T} contient les transitions de la forme $P \xrightarrow{a} P'$, pour deux parties P et P' de S , avec

$$P' = \{s' \in S \mid \text{il existe un état } s \in P \text{ tel que } s \xrightarrow{a} s' \text{ est une transition dans } T\}.$$

Proposition 7.4 (Corollaire). *Si L est reconnaissable alors $A^* \setminus L$ est aussi reconnaissable.*

Démonstration. En effet, on considère un automate $\mathcal{D} = (S, T, \{s_0\}, F)$ **déterministe et complet** $\mathcal{D} = (S, T, \{s_0\}, F)$ acceptant le langage L (au besoin en utilisant la construction précédente à partir d'un automate quelconque acceptant L). L'automate $\mathcal{C} = (S, T, \{s_0\}, S \setminus F)$ accepte alors $A^* \setminus L$, puisque tous les états qui étaient finaux ne le sont plus et réciproquement. Attention, cette construction n'est valable que parce que l'automate de départ est déterministe (un chemin au plus pour tout mot en entrée) et complet (un chemin au moins pour tout mot en entrée). \square

7.3 Produit, étoile

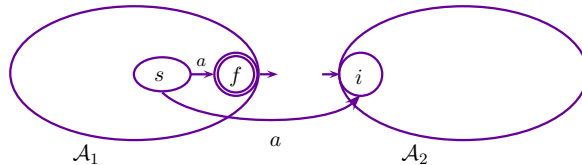
Proposition 7.5. *Si L_1 et L_2 sont deux langages reconnaissables alors $L_1 L_2$ est aussi reconnaissable. Si L est un langage reconnaissable alors L^* est aussi reconnaissable.*

• Soient $\mathcal{A}_1 = (S_1, T_1, I_1, F_1)$ et $\mathcal{A}_2 = (S_2, T_2, I_2, F_2)$ deux automates finis avec $L_1 = \mathcal{L}(\mathcal{A}_1)$ et $L_2 = \mathcal{L}(\mathcal{A}_2)$. Un automate fini acceptant $L_1 L_2$ est défini par $\mathcal{B} = (S_1 \cup S_2, T_1 \cup T_2 \cup J, I, F_2)$

$$\text{avec : } I = \begin{cases} I_1 & \text{si } I_1 \cap F_1 = \emptyset \\ I_1 \cup I_2 & \text{sinon (cas où } \varepsilon \in L_1) \end{cases}$$

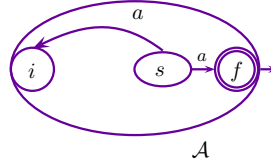
et $J = \{s \xrightarrow{a} i \mid i \in I_2 \text{ s'il existe } f \in F_1 \text{ tq } s \xrightarrow{a} f \in T_1\}$.

Ainsi, pour faire la « jonction » de \mathcal{A}_1 avec \mathcal{A}_2 , on procède en ajoutant de nouvelles transitions : pour toute transition allant vers un état final de \mathcal{A}_1 , on ajoute une transition de même étiquette vers un état initial de \mathcal{A}_2 .



- Pour $L = M^*$, on remarque que $M^* = M^+ \cup \{\varepsilon\}$, où $M^+ = \bigcup_{n \geq 1} M^n$.

A partir d'un automate $\mathcal{A} = (S, T, I, F)$ acceptant M , on construit un automate \mathcal{C} qui accepte M^+ : $\mathcal{C} = (S, T \cup J, I, F)$ avec $J = \{s \xrightarrow{a} i \mid i \in I \text{ et } \exists f \in F \text{ tq } s \xrightarrow{a} f \text{ dans } T\}$.



Il reste ensuite à réaliser l'opération pour l'union (comme au paragraphe précédent) avec l'automate de M^+ et un automate acceptant $\{\varepsilon\}$.

7.4 Langages rationnels et théorème de Kleene



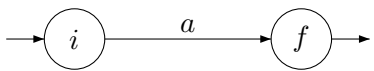
Définition 7.6. Soit A un alphabet. L'ensemble des langages rationnels sur A est défini inductivement par :

- (B) \emptyset , $\{\varepsilon\}$ et $\{a\}$, pour tout $a \in A$ sont des langages rationnels,
- (I) Si L_1 et L_2 sont des langages rationnels, alors $L_1 \cup L_2$ et $L_1 L_2$ sont aussi des langages rationnels, et si M est un langage rationnel, alors M^* est aussi un langage rationnel.

Remarque : on note souvent $+$ à la place de \cup .

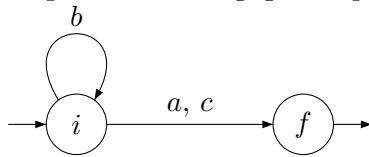
Théorème 7.7 (Théorème de Kleene). Un langage est rationnel ssi il est reconnaissable.

Démonstration. • Montrons d'abord par induction structurelle que tout langage rationnel est reconnaissable.

- Base. $L = \emptyset$ est accepté par l'automate suivant (qui n'a pas d'état final) : 
- $L = \{\varepsilon\}$ est accepté par : 
- et $L = \{a\}$ est accepté par l'automate : 

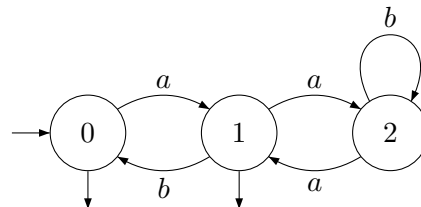
- Induction. $L_1 \cup L_2$, $L_1 L_2$ et M^* : voir les paragraphes précédents.
- Réciproquement, montrons que tout langage reconnaissable est rationnel.

Exemples. 1. Le langage accepté par l'automate \mathcal{A} suivant :



est $L(\mathcal{A}) = b^* \{a, c\}$ qui peut s'écrire aussi $b^*(a + c)$. Il s'exprime donc bien à partir des éléments de base et des opérations spécifiées.

2. Moins facile, on voudrait calculer (sans se tromper) une expression rationnelle du langage accepté par l'automate fini ci-dessous.



Dans le cas général, partant de $\mathcal{A} = (S, T, I, F)$, on considère la famille d'automates obtenus en prenant un état quelconque $s \in S$ comme état initial : $\mathcal{A}_s = (S, \{s\}, T, F)$ et on

note $L_s = \mathcal{L}(\mathcal{A}_s)$ le langage accepté par \mathcal{A}_s (c'est-à-dire les mots acceptés par \mathcal{A} en partant de s). Si les transitions sortant de s sont $s \xrightarrow{a_1} s_1, s \xrightarrow{a_2} s_2, \dots, s \xrightarrow{a_p} s_p$, on a l'égalité suivante :

$$L_s = \begin{cases} a_1 L_{s_1} + \dots + a_p L_{s_p} + \varepsilon & \text{si } s \text{ est un état final} \\ a_1 L_{s_1} + \dots + a_p L_{s_p} & \text{sinon} \end{cases}$$

qui exprime que tout chemin de s vers un état final passe par un des s_i (c'est-à-dire commence par un des a_i).

À partir de là, on résout le système d'équations en utilisant la proposition 7.8 suivante :

Proposition 7.8 (Lemme d'Arden). *Soient X et M deux langages et K un langage ne contenant pas le mot vide ε . L'équation $X = KX + M$ a pour unique solution $X = K^*M$. Autrement dit, K^*M est l'unique point fixe de $g: \mathcal{P}(A^*) \rightarrow \mathcal{P}(A^*)$.*

$$\begin{array}{ccc} g: & \mathcal{P}(A^*) & \longrightarrow & \mathcal{P}(A^*) \\ & X & \longmapsto & KX + M \end{array}$$

On remarque que le langage accepté par \mathcal{A} est : $\mathcal{L}(\mathcal{A}) = \bigcup_{s \in I} L_s$. Donc si E_s est l'expression rationnelle calculée par cette méthode pour chaque langage L_s , $s \in S$, alors l'expression finale cherchée pour le langage $L = \mathcal{L}(\mathcal{A})$ est : $\sum_{s \in I} E_s$. \square

Suite de l'exemple 2. Pour cet automate, notons :

- $L_0 = L(\mathcal{A})$,
- $L_1 = L(\mathcal{A}_1)$ en prenant 1 comme état initial,
- $L_2 = L(\mathcal{A}_2)$, l'ensemble des mots acceptés par \mathcal{A} en partant de l'état 2.

On obtient :

$$\begin{cases} L_0 &= aL_1 + \varepsilon & (1) \\ L_1 &= aL_2 + bL_0 + \varepsilon & (2) \\ L_2 &= bL_2 + aL_1 & (3) \end{cases}$$

- D'après (3), on a $L_2 = KL_2 + M$ avec $K = \{b\}$ et $M = aL_1$, donc $L_2 = b^*aL_1$.

- On remplace dans (2) :

$$L_1 = ab^*aL_1 + bL_0 + \varepsilon = KL_1 + M \text{ avec } K = ab^*a \text{ et } M = bL_0 + \varepsilon$$

$$\text{Donc } L_1 = (ab^*a)^*(bL_0 + \varepsilon)$$

- On remplace dans (1) :

$$\begin{aligned} L_0 &= a(ab^*a)^*(bL_0 + \varepsilon) \\ &= \underbrace{a(ab^*a)^*}_{K} \cdot bL_0 + \underbrace{a(ab^*a)^* + \varepsilon}_{M} \end{aligned}$$

Ce qui donne finalement pour le langage cherché, qui est justement L_0 :

$$L_0 = (a(ab^*a)^*b^*)^*[a(ab^*a)^* + \varepsilon]$$

7.5 Terminaison

Rappelons qu'un des problèmes examinés au début du cours était celui de l'arrêt d'un programme. Dans le cas de programmes décrits par des automates finis, ce problème est facile à résoudre :

Théorème 7.9. *Le problème de l'arrêt est décidable (en temps polynomial) pour les automates finis.*

Démonstration. Une exécution d'un automate fini s'arrête dans un état final de cet automate. On cherche donc un algorithme qui prend en entrée un automate fini $\mathcal{A} = (S, T, I, F)$ sur un alphabet A et décide s'il existe une exécution de cet automate partant d'un état de I et atteignant un état de F .

Soit donc $\mathcal{A} = (S, T, I, F)$ un automate fini sur un alphabet A . On définit, pour une partie P de S , l'ensemble des successeurs des états de P . Ce sont tous les états qu'on peut atteindre à partir d'un état de P en une transition :

$$Post(P) = \{s' \in S \mid \text{il existe un état } s \in P \text{ et une lettre } x \in A \text{ tels que } s \xrightarrow{x} s'\}.$$

L'algorithme consiste alors à calculer une suite $(P_n)_{n \in \mathbb{N}}$ de parties de S définie par :

$$P_0 = I \text{ (on part de l'ensemble des états initiaux),}$$

$$P_{n+1} = P_n \cup Post(P_n) \text{ pour tout } n \geq 0.$$

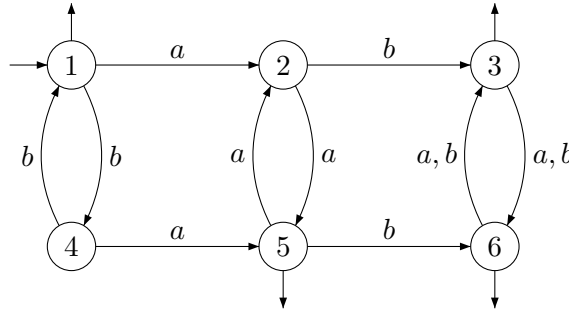
La suite $(P_n)_{n \in \mathbb{N}}$ est une suite croissante de sous-ensembles de S , qui est fini. Donc la suite est stationnaire à partir d'un certain rang N , inférieur au nombre d'états de l'automate \mathcal{A} . L'ensemble P_N , qui satisfait $P_N = P_{N+1}$, contient exactement tous les états accessibles depuis un état initial. Il vérifie donc :

il existe une exécution de \mathcal{A} partant d'un état de I et atteignant un état de F si et seulement si $P_N \cap F \neq \emptyset$. \square

7.6 Minimisation

Objectif : pour un langage L reconnaissable, trouver l'automate déterministe qui l'accepte avec un nombre minimal d'états.

Exemple. Sur $A = \{a, b\}$, en considérant l'automate ci-dessous, on a l'impression qu'on pourrait fusionner les états 3 et 6.



On considère dans ce paragraphe des automates finis déterministes complets et monogènes (un automate $\mathcal{A} = (S, T, I, F)$ est monogène si tout état $s \in S$ est accessible depuis l'état initial).

Dans un automate déterministe et complet $\mathcal{A} = (S, T, I, F)$, on a la propriété suivante : pour tout état $s \in S$, pour tout mot $u \in A^*$, il existe un unique s' tel que $s \xrightarrow{u} s'$ qui sera noté $s' = s \cdot u$

Définition 7.10. Soit $\mathcal{A} = (S, T, \{i\}, F)$ un automate fini déterministe complet et monogène. Deux états s_1 et s_2 sont dits *inséparables* si $\forall u \in A^*, s_1 \cdot u \in F \text{ ssi } s_2 \cdot u \in F$.

Autrement dit, si $s_1 \xrightarrow{u} s'_1$ et $s_2 \xrightarrow{u} s'_2$ alors $\begin{cases} \text{ou bien } s'_1 \text{ et } s'_2 \text{ sont tous deux dans } F \\ \text{ou bien ni l'un ni l'autre : } s'_1 \notin F \text{ et } s'_2 \notin F \end{cases}$

Ils sont *séparables* dans le cas contraire.

Proposition 7.11. *La relation sur S définie par $s_1 \sim s_2$ si s_1 et s_2 sont inséparables est une relation d'équivalence (appelée équivalence de Nérode).*

De plus, si $s_1 \sim s_2$, alors $\forall u \in A^, s_1.u \sim s_2.u$*

Exemple (suite). On vérifie que les classes pour \sim sont : $\{1\}, \{2\}, \{5\}, \{4\}, \{3, 6\}$.

Remarque. En particulier, si $s_1 \sim s_2$ ou bien $s_1 \in F$ et $s_2 \in F$ ou bien aucun des deux. De plus, pour toute lettre $a \in A$, $s_1 \cdot a \sim s \cdot a$.

Etant donné $\mathcal{A} = (S, T, \{i\}, F)$ et \sim l'équivalence de Nérode associée, on note $[s]$ la classe d'un état s pour \sim et $\tilde{S} = S/\sim$ l'ensemble quotient.

En utilisant la remarque ci-dessus, on peut étendre l'opération de changement d'état aux classes, en posant $[s] \cdot a = [s \cdot a]$.

On définit alors l'automate quotient $\tilde{\mathcal{A}} = (\tilde{S}, \tilde{T}, [i], \tilde{F})$ avec :

$$\begin{aligned}\tilde{F} &= \{[s] \mid s \in F\} \\ \tilde{T} &= \{[s] \xrightarrow{a} [s'] \mid s \xrightarrow{a} s' \text{ dans } \mathcal{A}\}\end{aligned}$$

Théorème 7.12. *Soit \mathcal{A} un automate fini déterministe complet et monogène et $L = L(\mathcal{A})$. Alors $\tilde{\mathcal{A}}$ est l'unique (à isomorphisme près) automate minimal de \mathcal{A} .*

Méthode de Moore : algorithme de calcul de \sim . On définit une suite de relations d'équivalences :

\sim_0 est la relation initiale à deux classes F et $S \setminus F$
pour $k \geq 0$, la relation \sim_{k+1} est définie par : $s_1 \sim_{k+1} s_2$ si $s_1 \sim_k s_2$ et $\forall a \in A, s_1 \cdot a \sim_k s_2 \cdot a$

Proposition 7.13. *L'algorithme se termine : il existe un entier p tel que $\sim_p = \sim_{p+1}$, et l'équivalence de Nérode est $\sim_p = \sim$.*

Exemple (suite et fin). On part des deux classes $I = \{1, 3, 5, 6\} = F$ et $II = \{2, 4\} = S \setminus F$ qui définissent l'équivalence \sim_0 .

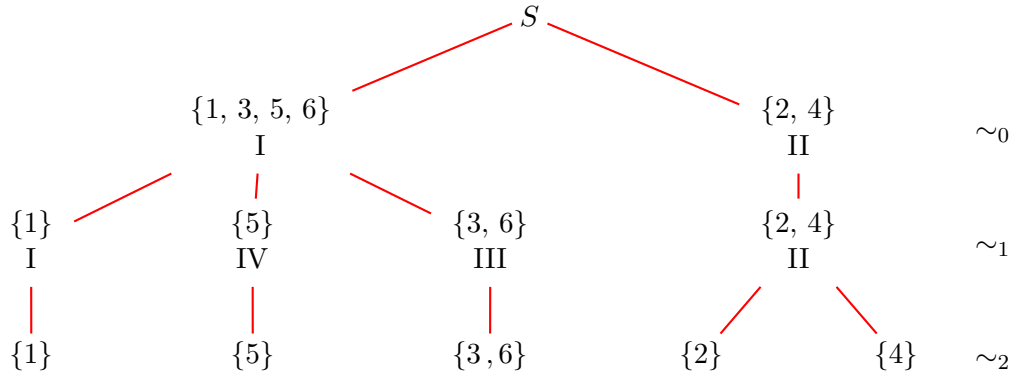
On construit l'équivalence \sim_1 en examinant les changements de classes (tableau de gauche) : 2 et 4 restent dans la même classe ainsi que 3 et 6, mais 1 et 5 se comportent différemment par b , de même que 1 et 3 par a , et 5 et 3 par a aussi.

Ceci nous amène à créer quatre classes pour \sim_1 : $I = \{1\}$, $II = \{2, 4\}$, $III = \{3, 6\}$, $IV = \{5\}$, qui sont reportées dans le tableau suivant (à droite). En réitérant l'opération, on construit \sim_2 qui va séparer 2 et 4 (à cause de b).

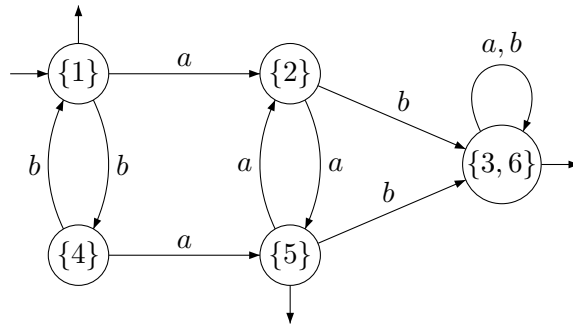
s	1	2	3	4	5	6
$\text{cl}(s)$	I	II	I	II	I	I
$\text{cl}(s \cdot a)$	II	I	I	I	II	I
$\text{cl}(s \cdot b)$	II	I	I	I	I	I

s	1	2	3	4	5	6
$\text{cl}(s)$	I	II	III	II	IV	III
$\text{cl}(s \cdot a)$	II	IV	III	IV	IV	III
$\text{cl}(s \cdot b)$	II	III	III	I	II	III

On a l'arbre suivant :



Conclusion. On voit qu'ajouter un étage de plus ne changera rien (3 et 6 sont clairement inséparables). Donc on en déduit que $\sim = \sim_2$. L'automate minimal $\tilde{\mathcal{A}}$ est construit à partir des classes de \sim_2 , en appliquant les transitions comme si elles s'appliquaient à un des états de la classe :



8 Fonctions booléennes

8.1 Algèbre de Boole

Définition 8.1 (Algèbre de Boole). Une algèbre de Boole est un tuple $\mathcal{B} = (E, \perp, \top, \vee, \wedge, \neg)$ où E est un ensemble, \perp et \top sont deux éléments distincts de E , \vee et \wedge sont deux opérations binaires, \neg est une opération unaire, satisfaisant les propriétés suivantes :

- Associativité : pour tous $a, b, c \in E$, $(a \vee b) \vee c = a \vee (b \vee c)$ et $(a \wedge b) \wedge c = a \wedge (b \wedge c)$
- Commutativité : pour tous $a, b \in E$, $a \vee b = b \vee a$ et $a \wedge b = b \wedge a$
- Distributivité d'une loi par rapport à l'autre : pour tous $a, b, c \in E$,
 $(a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c)$ et $(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c)$
- Absorption : pour tous $a, b \in E$, $a \wedge (a \vee b) = a$ et $a \vee (a \wedge b) = a$
- Idempotence : pour tout $a \in E$, $a \vee a = a$ et $a \wedge a = a$
- Bornes : pour tout $a \in E$, $a \wedge \perp = \perp$, $a \vee \perp = a$ et $a \wedge \top = a$, $a \vee \top = \top$
- Complémentarité : pour tout $a \in E$, $a \wedge \bar{a} = \perp$ et $a \vee \bar{a} = \top$

Exemples.

1. Soit $E = \mathcal{P}(A)$ pour un ensemble A non vide. On définit alors une algèbre de Boole avec :

\perp	\top	\vee	\wedge	\neg
\emptyset	A	\cup	\cap	complémentaire

2. Soit $\mathbb{B} = \{0, 1\}$. On définit alors une algèbre de Boole avec :

\perp	\top	\vee	\wedge	\neg
0 (faux)	1 (vrai)	ou (disjonction)	et (conjonction)	négation

Pour les calculs dans \mathbb{B} , on note généralement $+$ pour \vee et \cdot ou rien pour \wedge .

8.2 Fonctions booléennes

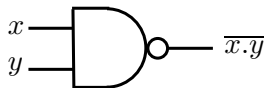
Définition 8.2 (Fonction booléenne). Soit $n \in \mathbb{N}$. Une fonction booléenne à n arguments est une application $f: \mathbb{B}^n \longrightarrow \mathbb{B}$.

Remarque. Si $n = 0$, il y a deux fonctions constantes : 0, 1.

Si $n = 1$, il y a quatre fonctions : $x \rightarrow 0$, $x \rightarrow 1$, $x \rightarrow x$ et $x \rightarrow \bar{x}$.

Il y a 2^{2^n} fonctions booléennes à n arguments.

Exemple. Table de vérité de la fonction *NAND* (à deux arguments), définie par :
 $NAND(x, y) = \overline{x \cdot y}$.



x	y	$NAND(x, y)$
0	0	1
0	1	1
1	0	1
1	1	0

Théorème 8.3. *Toute fonction booléenne f à n arguments, ($n \geq 1$), s'écrit comme combinaison de ses arguments ou de leurs complémentaires avec somme et produit.*

Exemple. $NAND(x, y) = \bar{x} \cdot \bar{y} + \bar{x} \cdot y + x \cdot \bar{y}$

Ce théorème se démontre par récurrence sur n , en utilisant le lemme suivant :

Lemme 8.4. *Soit f une fonction booléenne à n arguments.*

Alors $f(x_1, \dots, x_n) = x_1 f(1, x_2, \dots, x_n) + \bar{x}_1 f(0, x_2, \dots, x_n)$

Démonstration. Posons $g(x_1, \dots, x_n) = x_1 f(1, x_2, \dots, x_n) + \bar{x}_1 f(0, x_2, \dots, x_n)$ et montrons que $f = g$, c'est-à-dire que pour tout n -uplet (x_1, \dots, x_n) , $g(x_1, \dots, x_n) = f(x_1, \dots, x_n)$.

- Si $x_1 = 1$, $g(x_1, \dots, x_n) = f(1, x_2, \dots, x_n) = f(x_1, \dots, x_n)$
- Si $x_1 = 0$, $g(x_1, \dots, x_n) = f(0, x_2, \dots, x_n) = f(x_1, \dots, x_n)$

Donc $f = g$. □

8.3 Formes normales

Pour une fonction booléenne f à n arguments, on note $b = (b_1, \dots, b_n)$ un élément de \mathbb{B}^n et $\mathcal{D}_f = \{b \in \mathbb{B}^n / f(b) = 1\}$.

Définition 8.5 (Formes disjonctives et conjonctives).

- Une fonction est sous forme normale disjonctive (FND) si elle s'écrit comme une somme de produits de x_i ou \bar{x}_i .
- Une fonction est sous forme normale conjonctive (FNC) si elle s'écrit comme produit de sommes de x_i ou \bar{x}_i .

Pour une fonction booléenne f à n arguments, une forme normale disjonctive pour f est obtenue par :

$$f(x_1, \dots, x_n) = \sum_{b \in \mathcal{D}_f} M_b(x_1, \dots, x_n) \text{ où } M_b(x_1, \dots, x_n) = x'_1 \cdots x'_n$$

$$\text{avec } x'_i = \begin{cases} x_i & \text{si } b_i = 1 \\ \bar{x}_i & \text{si } b_i = 0 \end{cases}$$

Exemple. Pour la fonction $NAND$, on a : $NAND(x, y) = \underbrace{\bar{x} \cdot \bar{y}}_{M_{(0,0)}(x,y)} + \underbrace{\bar{x} \cdot y}_{M_{(0,1)}(x,y)} + \underbrace{x \cdot \bar{y}}_{M_{(1,0)}(x,y)}$

Soit f une fonction booléenne à n arguments, une forme normale conjonctive pour f est obtenue par :

$$f(x_1, \dots, x_n) = \prod_{b \notin \mathcal{D}_f} S_b(x_1, \dots, x_n) \text{ où } S_b(x_1, \dots, x_n) = x'_1 + \cdots + x'_n$$

$$\text{avec } x'_i = \begin{cases} x_i & \text{si } b_i = 0 \\ \bar{x}_i & \text{si } b_i = 1 \end{cases}$$

Exemple. Pour la fonction $NAND$, on a : $NAND(x, y) = \underbrace{\bar{x} + \bar{y}}_{S_{(1,1)}(x,y)}$

9 Calcul propositionnel

9.1 Syntaxe

Définition 9.1. Soit \mathcal{P} un ensemble de symboles propositionnels (ou variables propositionnelles). Les formules du calcul propositionnel sont définies inductivement par :

- (B) Si $p \in \mathcal{P}$, alors p est une formule.
- (I) Si F est une formule, alors $\neg F$ est une formule,
si F_1 et F_2 sont deux formules, alors $(F_1 \vee F_2)$ et $(F_1 \wedge F_2)$ sont aussi des formules.

Exemple 1. $F = ((q \wedge r) \vee \neg p)$ est une formule utilisant les symboles p , q et r .

Définition 9.2. On définit deux nouvelles opérations \rightarrow et \leftrightarrow par :

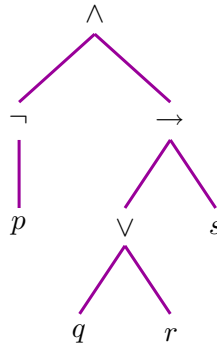
- $F_1 \rightarrow F_2 = F_2 \vee \neg F_1$
- $F_1 \leftrightarrow F_2 = (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$

Exemple 2. $G = (p \rightarrow (p \rightarrow q))$ est une formule utilisant les deux symboles p et q .

Remarque 1. Le symbole \supset est parfois utilisé au lieu de \rightarrow pour l'implication et parfois aussi \equiv au lieu de \leftrightarrow .

Remarque 2. Les formules du calcul propositionnel peuvent être vues comme les termes construits avec $F_0 = \mathcal{P}$, $F_1 = \{\neg\}$ et $F_2 = \{\vee, \wedge, \rightarrow, \leftrightarrow\}$, avec une notation infixée pour les opérateurs binaires, c'est-à-dire par exemple $p \vee q$ au lieu de $\vee(p, q)$.

Exemple 3. $F = \neg p \wedge ((q \vee r) \rightarrow s)$ est une formule sur le sous-ensemble $\{p, q, r, s\}$ de \mathcal{P} , qui peut être représentée (comme un terme) par un arbre :



9.2 Sémantique : interprétation des formules

Exemple. On reprend l'exemple ci-dessus. En associant des valeurs dans $\mathbb{B} = \{0, 1\}$ aux propositions p , q , r et s , on peut obtenir une valeur (également dans \mathbb{B}) pour la formule F , et plus généralement pour toute formule portant sur des propositions de $\{p, q, r, s\}$. Par exemple, l'interprétation $p \mapsto 0$, $q \mapsto 0$, $r \mapsto 1$, $s \mapsto 1$ produit la valeur 1 pour F .

Dans le cas général, une interprétation est une application $I: \mathcal{P} \longrightarrow \mathbb{B}$. À partir d'une telle application, qui associe à chaque proposition de \mathcal{P} une valeur dans $\mathbb{B} = \{0, 1\}$, il est possible de déduire une interprétation de toutes les formules de CP.

Définition 9.3. Soit $I: \mathcal{P} \longrightarrow \{0, 1\}$ une interprétation des symboles de \mathcal{P} . Le prolongement de I aux formules du calcul propositionnel est l'application encore notée I (au lieu de I^*) de $CP \longrightarrow \{0, 1\}$ définie inductivement par :

- (B) Si $F = p$, alors $I(F) = I(p)$.
(I) Si $F = \neg G$, alors $I(F) = \overline{I(G)}$,
si $F = F_1 \vee F_2$ alors $I(F) = I(F_1) + I(F_2)$
et si $F = F_1 \wedge F_2$ alors $I(F) = I(F_1)I(F_2)$.

Ceci correspond bien à interpréter les symboles d'opérations de la façon usuelle : \neg comme la négation, \wedge comme la conjonction et \vee comme la disjonction.

Proposition 9.4. Soit I une interprétation. Alors :

1. $I(F_1 \rightarrow F_2) = I(F_2) + \overline{I(F_1)}$
2. $I(F_1 \leftrightarrow F_2) = I(F_1).I(F_2) + \overline{I(F_1)}.\overline{I(F_2)}$

Démonstration. Montrons le premier point de la proposition précédente, le point 2. est laissé en exercice.

$$\begin{aligned}
I(F_1 \rightarrow F_2) &= I(F_2 \vee \neg F_1) \text{ par définition de } F_1 \rightarrow F_2 \\
&= I(F_2) + I(\neg F_1) \text{ par définition de } I(X \vee Y) \\
&= I(F_2) + \overline{I(F_1)} \text{ par définition de } I(\neg X) \\
\text{donc } I(F_1 \vee F_2) &= I(F_2) + \overline{I(F_1)}
\end{aligned}$$

□

Exemple. Soit $F = \neg p \wedge ((q \vee r) \rightarrow s)$

$$\begin{aligned}
I(F) &= \overline{I(p)}.I((q \vee r) \rightarrow s) \\
&= \overline{I(p)}.\overline{I(q \vee r)} + I(s) \\
&= \overline{I(p)}.\overline{I(q) + I(r)} + I(s) \\
I(F) &= \overline{I(p)}.\overline{I(q).I(r)} + I(s) \text{ d'après les lois de De Morgan}
\end{aligned}$$

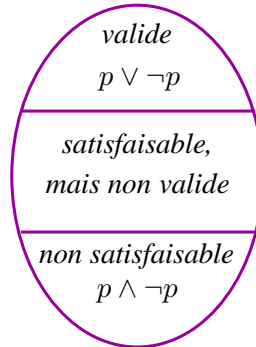
On retrouve le fait que si $I(p) = 0$, $I(r) = 1$, $I(q) = 0$ et $I(s) = 1$, alors $I(F) = 1$.

Définition 9.5. Soit F une formule.

- F est valide (ou une tautologie) si pour toute interprétation I , $I(F) = 1$.
- F est satisfaisable s'il existe une interprétation I telle que $I(F) = 1$.
- F est non satisfaisable si pour toute interprétation I , $I(F) = 0$.

Remarque 1. Lorsqu'une interprétation I est telle que $I(F) = 1$, on note parfois $I \models F$ qui se lit : « I satisfait F » ou « F est vraie pour I ».

Remarque 2. Une formule F est non satisfaisable si et seulement si $\neg F$ est valide.



Exemple. Pour le problème de Kerstin, Pollet et Anne, on considère la formule : $F = A \wedge B \wedge C$ (en omettant les parenthèses puisque l'interprétation de \wedge est associative), avec $A = p \rightarrow (q \wedge r)$, $B = \neg p \rightarrow q$ et $C = \neg p \rightarrow r$. Donc pour toute interprétation I , on a : $I(A) = I(q)I(r) + \overline{I(p)}$, $I(B) = I(q) + I(p)$ et $I(C) = I(r) + I(p)$. Le problème posé revient à chercher les interprétations I pour lesquelles F est vraie, c'est-à-dire $I(F) = 1$.

En posant $x = I(p)$, $y = I(q)$, $z = I(r)$, on obtient $I(F) = (yz + \overline{x})(x + y)(x + z)$, on retrouve donc la fonction booléenne : $f(x, y, z) = (yz + \overline{x})(x + y)(x + z)$, pour laquelle on avait vu que $f(x, y, z) = yz$. Donc $I(F) = 1$ si et seulement si l'interprétation I est telle que $y = I(q) = 1$ et $z = I(r) = 1$. Ainsi, on en déduit que Anne et Pollet iront à la conférence mais qu'on ne sait pas pour Kerstin.

9.3 Equivalence sémantique

Définition 9.6. Deux formules F et G sont équivalentes, noté $F \sim G$, si pour toute interprétation I , on a : $I(F) = I(G)$.

Remarque 1. La relation \sim sur l'ensemble CP des formules du calcul propositionnel est une relation d'équivalence.

Exemples. Les formules p et $\neg(\neg p)$ sont équivalentes, et en général :

- $F \sim \neg(\neg F)$
- $F \vee G \sim G \vee F$ (car $+$ est commutatif dans \mathbb{B})
- $\neg(F \vee G) \sim (\neg F \wedge \neg G)$

Remarque 2. On obtient des propriétés similaires à l'associativité, l'idempotence, l'absorption, la distributivité, etc. mais avec \sim au lieu de l'égalité. Ainsi, l'ensemble quotient CP / \sim est une algèbre de Boole.

Remarque 3. Sur $\mathcal{P} = \{p_1, \dots, p_n\}$, une interprétation $I : \mathcal{P} \rightarrow \mathbb{B}$ peut être identifiée au n -uplet de ses valeurs $I = (I(p_1), \dots, I(p_n)) \in \mathbb{B}^n$. On peut donc associer à toute formule F sur \mathcal{P} une fonction booléenne $g_F : \mathbb{B}^n \rightarrow \mathbb{B}$ définie par $g_F(I) = I(F)$. Ainsi :

Proposition 9.7. L'ensemble des fonctions booléennes est en bijection avec l'ensemble CP / \sim (qui contient les formules du calcul propositionnel à équivalence près).

Exemple. Soit $f : \mathbb{B}^2 \rightarrow \mathbb{B}$
 $(x, y) \rightarrow \overline{x + y}$

La formule F de CP qui lui est associée est $\neg(p \vee q)$ sur $\{p, q\}$.

Cette correspondance permet d'associer à toute formule F une formule F' sous forme normale conjonctive (FNC) et une formule F'' sous forme normale disjonctive (FND), qui sont équivalentes à F .

9.4 Conséquence sémantique

Définition 9.8. Pour deux formules F et G , on dit que G est conséquence de F , ou que F satisfait G , noté $F \models G$ si pour toute interprétation I , si $I(F) = 1$, alors $I(G) = 1$.

Proposition 9.9. F satisfait G ssi $(F \rightarrow G)$ est valide.

Démonstration. Montrons l'équivalence des négations, c'est-à-dire : F ne satisfait pas G ssi $(F \rightarrow G)$ n'est pas valide.

• Si F ne satisfait pas G , alors, par définition, il existe une interprétation I telle que $I(F) = 1$ et $I(G) = 0$. Pour cette interprétation I , on a $I(F \rightarrow G) = \overline{I(F)} + I(G) = 0$, donc $(F \rightarrow G)$ n'est pas valide.

• Réciproquement, si $(F \rightarrow G)$ n'est pas valide, alors il existe une interprétation I telle que $I(F \rightarrow G) = 0$, avec $I(F \rightarrow G) = \overline{I(F)} + I(G)$. Pour que la somme soit nulle, il faut que $I(F) = 1$ et $I(G) = 0$, donc F ne satisfait pas G .

Conclusion : F satisfait G ssi $(F \rightarrow G)$ est valide. \square

Proposition 9.10. F est équivalente à G ssi $F \leftrightarrow G$ est valide.

La démonstration reprend le schéma précédent, elle est laissée en exercice. On peut aussi vérifier que $F \sim G$ ssi $F \models G$ et $G \models F$ et utiliser la proposition précédente.

On étend les définitions de la conséquence sémantique \models à des ensembles de formules.

Définition 9.11. Soit $\mathcal{F} = \{F_1, \dots, F_n\}$ un ensemble fini de formules et G une formule.

- On dit que \mathcal{F} est satisfaisable s'il existe une interprétation I telle que pour toute formule $F \in \mathcal{F}$, on ait $I(F) = 1$. Donc \mathcal{F} est satisfaisable si la formule $\bigwedge_{i=1}^n F_i$ est satisfaisable.
- On note $\mathcal{F} \models G$ si $\bigwedge_{i=1}^n F_i \models G$, c'est-à-dire : pour toute interprétation I , si pour toute formule $F \in \mathcal{F}$, $I(F) = 1$, alors $I(G) = 1$.

Exemple. Montrer que $\mathcal{H} = \{p, p \rightarrow q, \neg q\}$ n'est pas satisfaisable.

Définition 9.12. Un séquent est une paire (\mathcal{F}, G) où \mathcal{F} est un ensemble de formules et G une formule. Le séquent (\mathcal{F}, G) est valide si $\mathcal{F} \models G$.

Exemple. Soient $\mathcal{F} = \{p, p \rightarrow q\}$ et $G = q$, on vérifie que (\mathcal{F}, G) est un séquent valide.

Montrons que pour toute interprétation I , si pour toute $F \in \mathcal{F}$, $I(F) = 1$, alors $I(G) = 1$. Ceci revient à montrer $(p) \wedge (p \rightarrow q) \models q$.

Supposons $I(p) = I(p \rightarrow q) = 1$. Or $I(p \rightarrow q) = \overline{I(p)} + I(q) = 1$. Mais comme $I(p) = 1$, on a $\overline{I(p)} = 0$ et $I(q) = 1$. Ainsi, (\mathcal{F}, G) est un séquent valide.

Remarque. Soit \mathcal{F}' l'ensemble obtenu en remplaçant dans \mathcal{F} un des F_i par F'_i tel que $F_i \sim F'_i$. Alors :

- \mathcal{F} est satisfaisable ssi \mathcal{F}' est satisfaisable et
- (\mathcal{F}, G) est valide ssi (\mathcal{F}', G) est valide.

Proposition 9.13. Soit \mathcal{H} un ensemble de formules, et F, G deux formules. On a les équivalences suivantes :

1. $\mathcal{H} \models G$ ssi $\mathcal{H} \cup \{\neg G\}$ est non satisfaisable.
2. $\mathcal{H} \cup \{F\} \models G$ ssi $\mathcal{H} \models (F \rightarrow G)$

Démonstration. 1. On démontre l'équivalence des négations.

- Supposons $\mathcal{H} \cup \{\neg G\}$ satisfaisable.

Alors il existe une interprétation I telle que $I(F) = 1$ pour toute formule F de \mathcal{H} et $I(\neg G) = 1$, donc $I(G) = 0$.

Donc on n'a pas $\mathcal{H} \models G$.

- Réciproquement, supposons que $\mathcal{H} \models G$ est faux. Alors il existe une interprétation I telle que $I(F) = 1$ pour toute F de \mathcal{H} et $I(G) = 0$.

Alors $I(\neg G) = 1$ donc I satisfait $\mathcal{H} \cup \{\neg G\}$.

2. Montrons cette proposition à l'aide d'équivalences.

$$\begin{aligned}
\mathcal{H} \cup \{F\} \models G & \text{ ssi } \mathcal{H} \cup \{F, \neg G\} \text{ non satisfaisable d'après (1.)} \\
& \text{ssi } \mathcal{H} \cup \{\neg(F \rightarrow G)\} \text{ non satisfaisable} \\
& \text{ssi } \mathcal{H} \models (F \rightarrow G) \text{ encore par (1.)}
\end{aligned}$$

□

9.5 Conséquence logique (ou déduction)

Définition 9.14. Un séquent (\mathcal{F}, G) est dit *prouvable*, noté $\mathcal{F} \vdash G$, s'il est obtenu après un nombre fini d'applications des six règles suivantes, où \mathcal{H} est un ensemble de formules, F et G des formules :

- (a) *Utilisation d'une hypothèse* : si $F \in \mathcal{H}$, alors $\mathcal{H} \vdash F$
- (b) *Augmentation d'hypothèse* : si $G \notin \mathcal{H}$ et $\mathcal{H} \vdash F$, alors $\mathcal{H} \cup \{G\} \vdash F$
- (c) *Modus ponens* : si $\mathcal{H} \vdash (F \rightarrow G)$ et $\mathcal{H} \vdash F$, alors $\mathcal{H} \vdash G$
- (d) *Retrait d'hypothèse (synthèse)* : si $\mathcal{H} \cup \{F\} \vdash G$, alors $\mathcal{H} \vdash (F \rightarrow G)$
- (e) *Double négation* : $\mathcal{H} \vdash F$ ssi $\mathcal{H} \vdash \neg\neg F$
- (f) *Absurde* : si $\mathcal{H} \cup \{F\} \vdash G$ et $\mathcal{H} \cup \{F\} \vdash \neg G$, alors $\mathcal{H} \vdash \neg F$

Exemple. Preuve de la démonstration par contraposée :

On veut prouver : $p \rightarrow q \vdash (\neg q \rightarrow \neg p)$

1. $\{p \rightarrow q, \neg q, p\} \vdash p$ d'après a)
2. $\{p \rightarrow q, \neg q, p\} \vdash \neg q$ d'après a)
3. $\{p \rightarrow q, \neg q, p\} \vdash p \rightarrow q$ d'après a)
4. $\{p \rightarrow q, \neg q, p\} \vdash q$ d'après c) appliquée à 1 et 3
5. $\{p \rightarrow q, \neg q\} \vdash \neg p$ d'après f) sur 2 et 4
6. $\{p \rightarrow q\} \vdash (\neg q \rightarrow \neg p)$ d'après d)

Théorème 9.15. Un séquent (\mathcal{F}, G) est valide ssi il est prouvable.

Remarque. Signification des deux sens de l'équivalence :

- Sens \Rightarrow : Complétude - *Ce qui est vrai peut être prouvé.*
- Sens \Leftarrow : Correction/adéquation : *Ce qui peut être prouvé est vrai.*

Principe de la démonstration de correction. Par induction sur la longueur de la preuve.

A partir d'un séquent valide, en appliquant une des six règles a), ..., f) on obtient un nouveau séquent valide.

Par exemple avec la règle a) :

On suppose que $\mathcal{F} \vdash G$ a été obtenu par la règle a), donc : $G \in \mathcal{F}$. Si I est une interprétation telle que $I(F) = 1$ pour toute formule F de \mathcal{F} , alors $I(G) = 1$ puisque $G \in \mathcal{F}$, donc $\mathcal{F} \models G$ et le séquent est valide. □

10 Logique du premier ordre

La logique du premier ordre enrichit le calcul propositionnel en utilisant :

- des termes construits avec des variables et des fonctions,
- des formules construites à partir de relations sur les termes, avec des opérateurs booléens et des quantifications sur les variables.

Par exemple $F : \forall x \exists y R(x, y)$ est une formule de la logique du 1er ordre. Dans cette formule, x et y sont des variables et $R(x, y)$ est une formule atomique construite en utilisant une relation binaire R .

- Si F est interprétée sur les entiers naturels, avec pour R la relation $<$, on obtient : *Pour tout entier x , il existe un entier y strictement plus grand que x* , ce qui exprime que l'ensemble des entiers naturels n'a pas d'élément maximal.
- Si F est interprétée sur l'ensemble des personnes avec pour R la relation définie par $R(x, y)$ si y est la mère de x , la formule exprime que *toute personne a une mère*.
- Si F est interprétée dans les mondes de Tarski, avec pour R la relation *LeftOf*, elle exprime que tout objet est à gauche d'un autre objet.

10.1 Syntaxe

On considère un ensemble \mathcal{G} de symboles de fonctions et un ensemble \mathcal{R} de symboles de relations. En particulier, on notera :

- $\mathcal{C} = \mathcal{G}_0$ l'ensemble des symboles de fonctions sans argument, c'est-à-dire les constantes,
- $\mathcal{P} = \mathcal{R}_0$ l'ensemble des symboles de relations d'arité nulle, c'est-à-dire les propositions (du calcul propositionnel), qui seront interprétées dans $\mathbb{B} = \{0, 1\}$.

On considère aussi un ensemble X de variables et on définit les termes et les formules de la logique du premier ordre associés à $\mathcal{G} \cup \mathcal{R} \cup X$.

Définition 10.1 (Termes avec variables). *L'ensemble $T(\mathcal{G}, X)$ des termes est défini inductivement par :*

- (B) *toute constante de \mathcal{C} est un terme et toute variable de X est un terme,*
- (I) *si $f \in \mathcal{G}$ a n arguments et si t_1, \dots, t_n sont des termes, alors $f(t_1, \dots, t_n)$ est un terme.*

Définition 10.2 (Formules). *Les formules de la logique du premier ordre sur \mathcal{G} et \mathcal{R} sont définies inductivement par :*

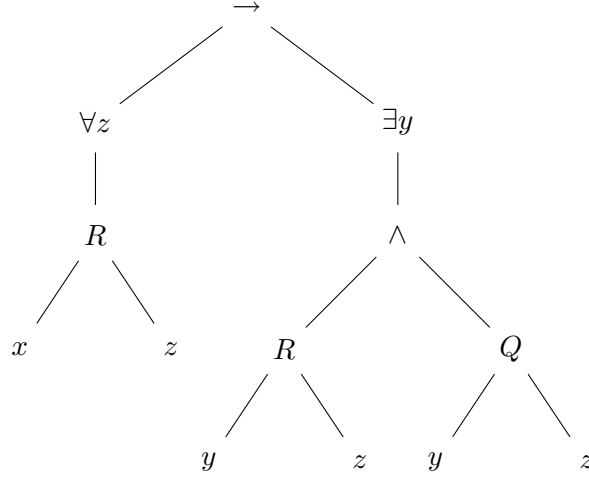
- (B) *si $R \in \mathcal{R}$ a n arguments et si t_1, \dots, t_n sont des termes, alors $R(t_1, \dots, t_n)$ est une formule dite atomique,*
- (I) *si F et G sont des formules alors $\neg F$, $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$ sont des formules, Si x une variable alors $\forall x F$ et $\exists x F$ sont des formules.*

Par exemple, dans la formule $\forall x \forall y (R(f(x, y), a) \rightarrow (R(x, a) \wedge R(y, a)))$, on trouve les termes a (constante), x, y (variables) et $f(x, y)$, pour une fonction f à deux arguments, ainsi qu'une relation binaire R . On pourra par la suite interpréter cette formule dans \mathbb{N} , avec l'addition pour f , l'égalité pour R , et la valeur 0 pour a .

Remarque 1. Les formules du calcul propositionnel sont des cas particuliers de cet ensemble.

Remarque 2. Les formules sont représentées par des arbres.

Par exemple, la formule $F : \forall z R(x, z) \rightarrow \exists y (R(y, z) \wedge Q(y, z))$ est représentée par l'arbre :



10.2 Variables libres et liées

Définition 10.3 (Variables d'un terme ou d'une formule).

1. Les variables d'un terme sont définies inductivement par :
 - $Var(x) = \{x\}$ si $x \in X$ et $Var(c) = \emptyset$ si $c \in \mathcal{C}$,
 - $Var(f(t_1, \dots, t_n)) = \bigcup_{i=1}^n Var(t_i)$ pour un terme $f(t_1, \dots, t_n)$.
2. Les variables d'une formule sont définies inductivement par :
 - $Var(R(t_1, \dots, t_n)) = \bigcup_{i=1}^n Var(t_i)$ pour une formule atomique,
 - si F et G sont des formules, x est une variable et $*$ $\in \{\wedge, \vee, \rightarrow\}$, alors : $Var(\neg F) = Var(F)$, $Var(F * G) = Var(F) \cup Var(G)$, $Var(\exists x F) = Var(\forall x F) = Var(F) \cup \{x\}$.

Remarque. Les variables et les constantes n'ayant pas d'argument, elles n'ont pas de descendant et sont toujours en position de feuilles dans l'arbre associé à une formule.

Définition 10.4 (Variables libres et liées).

- Dans l'arbre d'une formule F , une feuille d'étiquette $x \in X$ est une occurrence libre de x s'il n'y a aucun quantificateur $\forall x$ ou $\exists x$ dans les ascendants de cette feuille. Sinon, l'occurrence est dite liée.
- Une variable est libre dans une formule F si elle a **au moins** une occurrence libre dans cette formule. Elle est liée dans une formule si elle n'est pas libre dans cette formule.

On note $L(F)$ l'ensemble des variables libres dans F et $B(F) = Var(F) \setminus L(F)$ l'ensemble des variables liées dans F (B pour *bound* en anglais).

Par exemple, dans la formule F ci-dessus, en considérant les feuilles de gauche à droite, l'occurrence de x est libre, l'occurrence de z est liée, puis les deux occurrences de y sont liées tandis que les deux occurrences de z sont libres.

Par conséquent, $L(F) = \{x, z\}$ et $B(F) = \{y\}$

Définition 10.5 (Formule close). Une formule est dite close si elle n'a aucune variable libre.

Proposition 10.6. Les variables libres d'une formule sont définies inductivement par :

- (B) $L(R(t_1, \dots, t_n)) = \bigcup_{i=1}^n Var(t_i)$
- (I) Pour F et G deux formules,

- $L(\neg F) = L(F)$ et $L(F \star G) = L(F) \cup L(G)$ pour $\star \in \{\wedge, \vee, \rightarrow\}$
- $L(\forall x F) = L(\exists x F) = L(F) \setminus \{x\}$

10.3 Sémantique

Pour interpréter les formules, on va considérer une structure \mathcal{M} , donnée par :

- un domaine D ,
- pour toute fonction f de \mathcal{G} à n arguments, une fonction $f_D: D^n \longrightarrow D$
- Pour toute relation R de \mathcal{R} à n arguments, une relation $R_D \subseteq D^n$

En particulier, une constante a correspond à un élément a_D de D et une proposition p (relation sans argument) correspond à un élément de $\mathbb{B} = \{0, 1\}$.

On note $\mathcal{M} = (D, (f_D)_{f \in \mathcal{G}}, (R_D)_{R \in \mathcal{R}})$ une telle structure.

Exemple 1. Dans la structure $\mathcal{M} = (\mathbb{R}, 0, 1, +, \times, =)$, le domaine est $D = \mathbb{R}$, l'ensemble des nombres réels, les fonctions sont : les constantes 0 et 1, l'addition, la multiplication et il y a un seul prédicat qui est l'égalité.

Le terme $((x \odot x) \oplus x) \oplus a$ peut être interprété sur ce domaine, avec $a_{\mathbb{R}} = 1$, $\oplus_{\mathbb{R}}$ est l'addition, $\odot_{\mathbb{R}}$ est la multiplication. Il représente donc le polynôme P défini par $P(x) = x^2 + x + 1$.

Considérons maintenant la formule atomique $F: Q(((x \odot x) \oplus x) \oplus a, b)$. En interprétant le prédicat binaire Q comme l'égalité et avec $b_{\mathbb{R}} = 0$, cette formule s'interprète comme un prédicat unaire avec x comme variable libre : $P(x) = 0$.

La formule $\exists x F$ correspond alors à l'énoncé : le polynôme P a une racine dans \mathbb{R} .

Exemple 2. Lorsque \mathcal{M} décrit une base de données, les requêtes sont des formules.

Définition 10.7. Pour une structure \mathcal{M} avec domaine D , une valuation est une application $v: X \longrightarrow D$.

Proposition 10.8. Etant données une structure \mathcal{M} associée à $\mathcal{G} \cup \mathcal{R}$, avec domaine D , et une valuation $v: X \longrightarrow D$, la valeur d'un terme $v^*(t) \in D$ est définie inductivement par :

- (B) $v^*(a) = a_D$ pour une constante a et $v^*(x) = v(x)$ pour une variable $x \in X$,
- (I) Si $t = f(t_1, \dots, t_n)$ pour une fonction f à n arguments et des termes t_1, \dots, t_n , alors $v^*(t) = f_D(v^*(t_1), \dots, v^*(t_n))$.

Remarque. A chaque $v: X \longrightarrow \mathcal{D}$, on associe $v^*: T(\mathcal{G}, X) \longrightarrow \mathcal{D}$

$$t \longmapsto v^*(t)$$

Définition 10.9. Soient \mathcal{M} une structure de domaine D , $v: X \longrightarrow D$ une valuation, $x \in X$ une variable et $a_D \in \mathcal{D}$.

La valuation $v' = v[x \mapsto a_D]$ est définie par :
$$\begin{cases} v'(y) = v(y) & \text{si } y \neq x \\ v'(x) = a_D \end{cases}$$

Exemple. On considère $X = \{x, y, z, w\}$ et $D = \mathbb{N}$.

Déterminer $v_1 = v[z \rightarrow 3]$ et $v_2 = v_1[x \rightarrow 1]$ pour la valuation v définie par :

$$v: \begin{cases} x \mapsto 2 \\ y \mapsto 8 \\ z \mapsto 7 \\ w \mapsto 14 \end{cases}$$

Définition 10.10. On définit la valuation (ou valeur de vérité, ou interprétation) d'une formule F , notée $\hat{v}(F)$, inductivement par :

- (B) Si $F = R(t_1, \dots, t_n)$, alors $\hat{v}(F) = 1$ ssi $(v^*(t_1), \dots, v^*(t_n)) \in R_D$
- (B) Pour deux formules F et G , et une variable x ,
 - $\hat{v}(\neg F) = \overline{\hat{v}(F)}$
 - $\hat{v}(F \wedge G) = \hat{v}(F) \hat{v}(G)$
 - $\hat{v}(F \vee G) = \hat{v}(F) + \hat{v}(G)$
 - $\hat{v}(F \rightarrow G) = \overline{\hat{v}(F)} + \hat{v}(G)$
 - $\hat{v}(\forall x F) = 1$ ssi pour toute $a_D \in D$, $v[\widehat{x \mapsto a_D}](F) = 1$
 - $\hat{v}(\exists x F) = 1$ ssi il existe $a_D \in D$ tel que $v[\widehat{x \mapsto a_D}](F) = 1$

Exemple 1. Soit $F : Q((x \odot x) \oplus x) \oplus a, b$, la formule considérée précédemment, avec le même modèle, et la valuation v telle que $v(x) = 2$. Alors $\hat{v}(F) = 0$. De plus, comme aucune valeur de x ne peut être racine, on a aussi $\hat{v}(\exists x F) = 0$.

Exemple 2. Soit $F = R(f(x), g(y))$ et la structure $\mathcal{M} = (\mathbb{N}, f_{\mathbb{N}}, g_{\mathbb{N}}, \leq)$, avec $f_{\mathbb{N}}(n) = n + 1$ et $g_{\mathbb{N}}(n) = n + 3$, et soit v_0 la valuation définie par : $v_0 \begin{cases} x \mapsto 4 \\ y \mapsto 3 \end{cases}$

- $v_0^*(f(x)) = f_{\mathbb{N}}(v_0(x)) = f_{\mathbb{N}}(4) = 5$
- $v_0^*(g(y)) = g_{\mathbb{N}}(v_0(y)) = 6$

Ainsi, $\hat{v}_0(F) = 1$ car $5 \leq 6$ et, plus généralement, si v est une valuation quelconque, $\hat{v}(F) = 1$ ssi $v(x) + 1 \leq v(y) + 3$.

Définition 10.11.

1. Etant données une formule F et une structure \mathcal{M} ,
 - (a) F est satisfaisable pour \mathcal{M} , s'il existe une valuation v telle que $\hat{v}(F) = 1$.
 - (b) F est valide pour \mathcal{M} , si pour toute valuation v , $\hat{v}(F) = 1$.
On dit alors que \mathcal{M} est un modèle de F (noté $\mathcal{M} \models F$).
2. Etant donnée une formule F :
 - (a) F est satisfaisable s'il existe une structure \mathcal{M} telle que F est satisfaisable pour \mathcal{M} .
 - (b) F est valide (ou universellement valide), si pour toute structure \mathcal{M} , F est valide pour \mathcal{M} .

Remarque. Le problème 2.a est indécidable.

Le problème 1.a est décidable pour des structures finies. C'est le cas par exemple pour la satisfaisabilité d'une requête dans une base de données.

Proposition 10.12. Il existe un algorithme qui prend en entrée une structure finie \mathcal{M} et une formule du premier ordre F , et qui décide s'il existe une valuation v telle que $\hat{v}(F) = 1$.

Pour les structures infinies, c'est plus compliqué :

- Pour $\mathcal{M}_1 = (\underbrace{\mathbb{N}}_D, \underbrace{0, 1, +, \times, \exp}_G, \underbrace{=}_R)$, la satisfaisabilité (d'une formule du premier ordre) est indécidable.
- Dommage... : $\exists n \exists x \exists y \exists z (x^n + y^n = z^n) \wedge (n \geq 3)$

- Pour $\mathcal{M}_2 = (\mathbb{N}, 0, 1, +, \times, =)$, le problème est également indécidable.
Dommage... : $\exists x P(x) = 0$ où P est un polynôme.
- Mais pour $\mathcal{M}_3 = (\mathbb{R}, +, \times, <)$, le problème est décidable !

Exemples de modèles. Tout ensemble E muni d'une relation binaire R_E réflexive est un modèle de la formule $F_1 : \forall x R(x, x)$.

Tout ensemble ordonné $\mathcal{M} = (E, \preceq)$ est un modèle de la formule $F_1 \wedge F_2 \wedge F_3$ avec F_1 comme ci-dessus et :

$$\begin{aligned} F_2 : & \forall x \forall y ((R(x, y) \wedge R(y, x)) \rightarrow x = y) \\ F_3 : & \forall x \forall y \forall z ((R(x, y) \wedge R(y, z)) \rightarrow R(x, z)) \end{aligned}$$

10.4 Propriétés sémantiques

Définition 10.13. Soit \mathcal{M} une structure, F et G deux formules.

- $F \models G$ si, pour toute valuation v de \mathcal{M} , si $\hat{v}(F) = 1$, alors $\hat{v}(G) = 1$.
- F et G sont équivalentes, noté $F \sim G$ si pour toute structure \mathcal{M} et pour toute valuation v , $\hat{v}(F) = \hat{v}(G)$

Remarques.

- On note parfois $\models_{\mathcal{M}}$ au lieu de \models lorsqu'il y a ambiguïté sur la structure considérée.
- Comme pour le calcul propositionnel, $F \sim G$ ssi pour toute structure \mathcal{M} , on a : $F \models G$ et $G \models F$.
- On a toutes les équivalences du calcul propositionnel :
 $\neg(F \vee G) \sim \neg F \wedge \neg G$, $\neg\neg F \sim F$, ...
- On voudrait des équivalences plus « riches », qui impliquent les variables.

Proposition 10.14. Soit F une formule. On a l'équivalence suivante :

$$\neg\forall x F \sim \exists x \neg F$$

Démonstration. Soit \mathcal{M} une structure et v une valuation. On a :

$$\begin{aligned} \hat{v}(\neg\forall x F) = 1 & \quad \text{ssi} \quad \hat{v}(\forall x F) = 0 \\ & \quad \text{ssi} \quad \text{il existe } a \in \mathcal{D} \text{ tq } v[\widehat{x \rightarrow a}](F) = 0 \\ & \quad \text{ssi} \quad \text{il existe } a \in \mathcal{D} \text{ tq } v[\widehat{x \rightarrow a}](\neg F) = 1 \\ & \quad \text{ssi} \quad \hat{v}(\exists x \neg F) = 1 \end{aligned}$$

□

Lemme 10.15. Soit \mathcal{M} une structure, v_1 et v_2 deux valuations.

1. Si t est un terme tel que $v_1|_{\text{Var}(t)} = v_2|_{\text{Var}(t)}$ (v_1 et v_2 coïncident sur $\text{Var}(t)$), alors $v_1^*(t) = v_2^*(t)$
2. Si F est une formule telle que $v_1|_{L(F)} = v_2|_{L(F)}$ (v_1 et v_2 coïncident sur les variables libres de F), alors $\hat{v}_1(F) = \hat{v}_2(F)$

Exemple pour un terme. Soit $t = g(x, y)$ et deux valuations v_1 et v_2 telles que $v_1(x) = v_2(x)$ et $v_1(y) = v_2(y)$.

Alors $v_1^*(t) = g_{\mathcal{D}}(v_1(x), v_1(y)) = g_{\mathcal{D}}(v_2(x), v_2(y)) = v_2^*(t)$.

Remarque. Ce lemme exprime que la valeur de vérité d'une formule ne dépend que des valeurs de ses variables libres. Par exemple, la valeur de la formule $F: \forall x R(y, x)$ ne dépend que de y .

Corollaire 10.16 (Conséquences). *Si F est une formule close (c'est-à-dire une formule sans variable libre), alors $\hat{v}(F)$ est constante, indépendante de v .*

Ainsi, F et $\forall x F$ sont équivalentes si x n'est pas libre dans F .

De même, $F \sim \exists x F$ si $x \notin L(F)$.

Exemple. Soit $F: \forall x \exists y R(x, y)$ interprétée avec pour \mathcal{D} l'ensemble des personnes et R la relation définie par $R(x, y)$ si « y est la mère de x ». Alors, pour toute valuation v , $\hat{v}(F) = 1$.

Rappel : D'après la définition inductive de $L(F)$, on a :

$L(\forall x F) = L(F) \setminus \{x\} = L(F)$ si x n'est pas libre dans F .

Démontrons la propriété : si $x \notin L(F)$ alors $F \sim \forall x F$.

On se rappelle tout d'abord que, d'après la définition inductive de $L(F)$, on a : $L(\forall x F) = L(F) \setminus \{x\} = L(F)$ si x n'est pas libre dans F .

Soit maintenant v une valuation. On a par définition de $\hat{v}(\forall x F)$:

$\hat{v}(\forall x F) = 1$ si pour tout $a \in \mathcal{D}$, $v[\widehat{x \rightarrow a}](F) = 1$.

Mais v et $v[x \rightarrow a]$ coïncident partout sauf sur x , donc elles coïncident sur $L(F) = L(\forall x F)$ d'après ce qui précède.

Donc d'après le lemme : $\hat{v}(F) = v[\widehat{x \rightarrow a}](F) = 1$, et on obtient bien $\hat{v}(F) = \hat{v}(\forall x F)$.