DUT INFO 1 / 2013-2014

Mis à jour le 27 février 2014 à 19:42

Mathématique discrète et informatique

Guillaume CONNAN

UNIVERSITÉ DE NANTES

IUT NANTES Informatique

InforMATHique

TABLE DES MATIÈRES

1 Logique des propositions **5**

1.1 Test préliminaire . . . 6
1.2 Contexte . . . 6
1.3 Syntaxe . . . 7
1.3.1 Les symboles . . . 7
1.3.2 Représentation à l'aide d'arbres . . . 7
1.3.3 Démonstration par induction . . . 8
1.3.4 Sous-formules . . . 8
1.4 Sémantique . . . 8
1.4.1 Valeurs sous un environnement . . . 8
1.4.2 Avec Haskell . . . 9
1.4.3 Tables de vérité . . . 9
1.4.4 Tautologies, formules satisfiables, insatisfiables . . . 9
1.4.5 Conséquences et équivalences logiques . . . 9
1.4.6 Formules d'équivalence de la logique des propositions . . . 11
1.4.7 Principe de déduction par réfutation de la conclusion . . . 11
1.5 Approche formelle de la logique propositionnelle (logique niveau II...) . . . 11
1.5.1 Principe général . . . 11
1.5.2 Théorème de la déduction . . . 13
1.5.3 Retour sur la démonstration par réfutation . . . 14
1.5.4 Raisonnement par l'absurde / par contraposée . . . 14
1.6 Récurrence . . . 15
1.6.1 Retour sur le raisonnement par récurrence et sur le sens de l'implication . . . 15
1.6.2 Récursion, récurrence, induction . . . 16
1.6.3 Génétique syldave . . . 16
1.7 EXERCICES . . . 17

2 Ensembles **22**

2.1 L'informaticien(ne) sur le terrain . . . 23
2.2 Tout est ensemble . . . 23
2.2.1 Une définition ? . . . 23
2.2.2 Kit de survie en logique des prédicats . . . 24
2.3 Parties d'un ensemble . . . 26
2.3.1 Notion d'égalité . . . 26
2.3.2 Inclusion . . . 27
2.3.3 Ensemble des parties d'un ensemble . . . 27
2.3.4 Construction récursive . . . 28
2.4 Algèbre des ensembles . . . 28
2.4.1 Intersection . . . 28
2.4.2 Union . . . 29
2.4.3 Différence . . . 29
2.4.4 Différence symétrique . . . 29
2.4.5 Complémentaire . . . 30
2.4.6 Lois de De Morgan . . . 30
2.4.7 Dualité . . . 31
2.5 Partition d'un ensemble . . . 31
2.6 Produit cartésien . . . 32
2.6.1 À table . . . 32
2.6.2 Paire ordonnée - Produit de deux ensembles . . . 33
2.6.3 n-uplets - Produit d'un nombre quelconque d'ensembles . . . 34

2.7 Notion de cardinal . . . 34
2.8 Fonction caractéristique (niveau II...) . . . 36
2.9 EXERCICES . . . 39

3 Relations et fonctions 41
3.1 Préliminaires... . . . 42
3.1.1 Un peu de calcul matriciel . . . 42
3.1.2 Calcul booléen . . . 43
3.2 Relations binaires . . . 44
3.2.1 Au CM1 . . . 44
3.2.2 Généalogie . . . 44
3.2.3 À l'IUT . . . 44
3.2.4 Retour au CE1 . . . 45
3.2.5 Domaine, codomaine . . . 45
3.2.6 Représentation d'une relation . . . 45
3.2.7 Relation transposée . . . 47
3.2.8 Image, contre image . . . 48
3.2.9 Égalité de deux relations . . . 49
3.2.10 Principales opérations sur les relations . . . 49
3.2.11 Composition de relations . . . 51
3.3 Fonctions . . . 52
3.3.1 Définitions . . . 52
3.3.2 Fonctions particulières . . . 53
3.3.3 Fonction injective . . . 54
3.3.4 Fonction surjective . . . 55
3.3.5 Fonction bijective . . . 55
3.4 Relations binaires sur un ensemble . . . 56
3.4.1 Au CE1 . . . 56
3.4.2 Définition . . . 56
3.4.3 Représentations . . . 56
3.4.4 Composition . . . 57
3.4.5 Chemin . . . 57
3.4.6 Propriétés des relations sur un ensemble . . . 57
3.4.7 Fermeture transitive et chemin . . . 59
3.5 Un peu d'ordre . . . 60
3.5.1 Pré-ordre . . . 60
3.5.2 Relations d'équivalence . . . 60
3.5.3 Relation d'ordre - Poset . . . 61
3.6 EXERCICES . . . 64
3.6.1 Relations binaires . . . 64
3.6.2 Relations binaires sur un ensemble . . . 65
3.6.3 Pré-ordres . . . 66

4 Programmation fonctionnelle et Haskell for dummies 69
4.1 Rapide introduction . . . 70
4.2 Un jeu . . . 73
4.3 Une programmation désaffectée . . . 78
4.3.1 Environnement . . . 78
4.3.2 Les fonctions . . . 79
4.3.3 Un exemple . . . 79
4.4 Polymorphisme - abstraction - récursion - listes . . . 81
4.4.1 Récursion, induction, récurrence . . . 82
4.4.2 Zoom sur les listes . . . 86

5 Arithmétique : un premier survol 90
5.1 Structures mères . . . 91
5.1.1 Lois de composition interne . . . 91
5.1.2 Les groupes . . . 92

5.1.3 Le groupe (Z/nZ,+) . . . 92
5.2 Divisibilité dans Z . . . 98
5.2.1 Propriété fondamentale de N . . . 98
5.2.2 PGCD . . . 98
5.2.3 Égalité de Bézout . . . 98
5.2.4 Algorithme des différences . . . 99
5.2.5 Algorithme d'Euclide I . . . 99
5.2.6 Algorithme d'Euclide II . . . 100
5.2.7 Nombres premiers entre eux . . . 101
5.3 À la recherche des nombres premiers . . . 102
5.3.1 Définition . . . 102
5.3.2 Comment vérifier qu'un nombre est premier ? . . . 103
5.3.3 Tests et cribles . . . 103
5.3.4 Décomposition des entiers en produit de nombres premiers . . . 104
5.4 Éléments inversibles de Z/nZ . . . 105
5.4.1 Anneaux et corps . . . 105
5.4.2 Comment calculer l'inverse modulaire d'un entier ? . . . 106
5.4.3 Petit théorème de FERMAT . . . 106
5.5 EXERCICES . . . 108
5.5.1 Structures mères . . . 108
5.5.2 Division euclidienne . . . 108
5.5.3 PGCD . . . 110
5.5.4 Nombres premiers . . . 110
5.5.5 Complexité et relation de domination . . . 110
5.5.6 Exercices « papier-crayon » . . . 111
5.5.7 Exponentiation rapide . . . 117

A Raccourcis emacs **119**

B Module logique en Haskell **122**

C Mon livre de CM1 **125**

Lectures recommandées **140**

CHAPITRE 1

Logique des propositions

La logique est omniprésente en informatique, dès le plus bas niveau de l'architecture des ordinateurs (tout dépend en effet de petites portes logiques qui laissent ou ne laissent pas passer le courant) mais aussi dans le domaine de pointe de l'intelligence artificielle où un « robot » est confronté au problème de conséquence logique : à partir d'un certain nombre de connaissances acquises, que puis-je en déduire ? Il existe de plus des langages utilisant une programmation dite logique comme le célèbre PROLOG.
Nous ne ferons dans un premier temps qu'étudier sommairement la logique des propositions, c'est-à-dire une logique en dehors de tout contexte (spatial, temporel,...)

Dr. McCoy : *Mr. Spock, remind me to tell you that I'm sick and tired of your logic.*

Spock : *That is a most illogical attitude.*

in « Star Trek : The Galileo Seven » (1967)

1 Test préliminaire

Voici un petit test proposé [a] il y a quelques années à des étudiants entrant à l'Université et à des professeurs de mathématique :

> *Soit un entier naturel n inférieur à 20. On considère la proposition « si n est pair, alors son successeur est premier ». Quels sont les entiers qui rendent cette proposition vraie ?*

Ne lisez pas ce qui suit et répondez tout de suite [b].

Ça y est ? Bon, analysons les réponses possibles. Notons $E = [\![0, 20]\!]$. Vous avez répondu...

— ... « 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19 » : vous avez triché ou bien vous avez eu un bon cours de logique au lycée et vous l'avez bien étudié. La lecture des paragraphes suivants va vous permettre d'aller plus loin ;

— ... « 2, 4, 6, 10, 12, 16, 18 » : c'est bien, vous n'avez pas triché mais malheureusement ce n'est pas la bonne réponse. La lecture des paragraphes suivants devrait vous permettre de progresser en logique pour affronter sereinement vos deux années d'IUT.

— ... « tous » : vous avez eu raison de ne pas sortir le jeudi soir pour travailler et vous remettre à niveau. La lecture des paragraphes suivants devrait vous permettre de découvrir la logique, l'arithmétique et la mathématique en général pour affronter sereinement vos deux années d'IUT.

2 Contexte

On appellera *proposition* tout énoncé dont on peut décider s'il est vrai ou faux indépendamment de ses composantes.

On distinguera la logique des propositions de la logique des *prédicats* qui introduit des variables dans les assertions qui peuvent les rendre donc parfois vraies et parfois fausses.

Par exemple « *Igor dort* » est une proposition susceptible d'être vraie ou fausse dans toute situation alors que la valeur de vérité de « x *dort* » dépend de x.

Nous distinguerons également la *syntaxe* de la logique des propositions de sa *sémantique*.

Le dictionnaire propose les définitions suivantes :

— **syntaxe** : n. f. (bas latin syntaxis, du grec suntaksis, ordre) Partie de la grammaire qui décrit les règles par lesquelles les unités linguistiques se combinent en phrases.

— **sémantique** : n. f. (bas latin semanticus, du grec sêmantikos, qui signifie) **1.** Étude du sens des unités linguistiques et de leurs combinaisons. **2.** Étude des propositions d'une théorie déductive du point de vue de leur vérité ou de leur fausseté.

L'étude de l'aspect syntaxique consiste à préciser comment l'on construit les formules et l'aspect sémantique interprète les formules en terme de *Vrai* ou *Faux*.

a. Cité dans la note de synthèse de l'habilitation à diriger des recherches soutenue par Viviane DURAND-GUERRIER le 16 juin 2005 http://tel.archives-ouvertes.fr/docs/00/20/16/26/PDF/hdrvdg.pdf page 47.

b. Personne ne sera au courant de votre réponse, alors n'ayez pas peur...

3 Syntaxe

3 1 Les symboles

On doit d'abord vérifier si les formules sont bien écrites. On a donc d'abord besoin d'un alphabet, c'est-à-dire un ensemble de symboles.
L'alphabet du langage de la logique des propositions est constitué de :

- un ensemble fini (ou infini dénombrable, i.e. en bijection avec $\mathbb{N}$) de *variables propositionnelles*. Ce sont les atomes ou plutôt les *propositions atomiques*. Il s'agit des plus petites propositions pouvant être soit vraies, soit fausses. Par exemple : « Le prof de maths est génial ».
- la *constante* logique $\bot$ qui se lit « FAUX ». On peut parfois également introduite la variable $\top$ qui se lit « VRAI ».
- les *connecteurs* $\neg$ (NON), $\wedge$ (ET), $\vee$ (OU), $\rightarrow$ (IMPLIQUE) et $\leftrightarrow$ (ÉQUIVALENT) ;
- les parenthèses « (» et «) ».

Le connecteur $\neg$ est un *connecteur unaire* et les autres sont des *connecteurs binaires*.
L'*ensemble des formules* de la logique propositionnelle est le plus petit ensemble $\mathcal{F}$ tel que :

- toute variable propositionnelle est un élément de $\mathcal{F}$;
- $\bot$ est un élément de $\mathcal{F}$;
- si $p \in \mathcal{F}$, alors $(\neg p) \in \mathcal{F}$;
- si p et q sont dans $\mathcal{F}$, alors $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$, et $(p \leftrightarrow q)$ sont des éléments de $\mathcal{F}$;
- il n'y a pas d'autres expressions bien formées que celles décrites par les règles précédentes.

Remarque

Vous aurez remarqué que l'ensemble $\mathcal{F}$ est défini à partir de lui-même : on parle de définition *récursive*.

Par exemple, $s \rightarrow ((p \wedge (\neg q)) \vee (p \leftrightarrow (\neg r)))$ est une proposition mais $\neg \wedge (p) \vee$ ne l'est pas.
On peut se passer de quelques parenthèses en adoptant les règles de priorité suivantes :

- $\neg$ est prioritaire sur les autres opérateurs ;
- $\vee$ et $\wedge$ sont prioritaires sur $\rightarrow$ et $\leftrightarrow$.

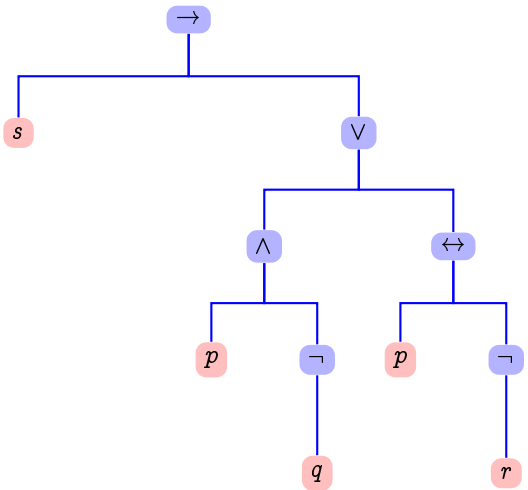
Mais attention ! $p \vee q \wedge r$ est ambigu.

Remarque

Il y a bien plus de connecteurs logiques que ceux évoqués ici : combien y en a-t-il à votre avis ?
On parle aussi d'*arité* d'un connecteur. Par exemple, $\neg$ est un connecteur d'arité 1 et $\wedge$ est un connecteur d'arité 2.

3 2 Représentation à l'aide d'arbres

Une formule logique peut être représentée par un arbre binaire. Reprenons la formule introduite plus haut : $s \rightarrow ((p \wedge (\neg q)) \vee (p \leftrightarrow (\neg r)))$



Si une formule contient au moins un connecteur, on appelle *connecteur principal* de la formule le connecteur qui se trouve à la racine de l'arbre (ici c'est $\rightarrow$).

3.3 Démonstration par induction

Théorème 1 - 1

Si une propriété P portant sur les formules de $\mathcal{F}$ est telle que :

- toute variable propositionnelle vérifie P ;
- $\bot$ vérifie P ;
- si la formule p vérifie P, alors $(\neg p)$ vérifie P ;
- si p et q vérifient P, alors $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$ et $(p \leftrightarrow q)$ vérifient P ;

alors toutes les formules de $\mathcal{F}$ vérifient P.

Ce théorème permet par exemple de prouver le théorème suivant :

Théorème 1 - 2

Toute formule de $\mathcal{F}$ a autant de parenthèses ouvrantes que de parenthèses fermantes.

On vérifie en effet les quatre critères assez aisément.

3.4 Sous-formules

Nous n'entrerons pas dans les détails mais on peut définir par induction les sous-formules d'une formule donnée : on retiendra juste qu'il s'agit de toutes les formules apparaissant dans une formule donnée.
Par exemple, l'ensemble des sous-formules de $(p \rightarrow q) \vee \neg(q \leftrightarrow r)$ est

$$\{p \rightarrow q, \neg(q \leftrightarrow r), q \leftrightarrow r, p, q, r\}$$

4 Sémantique

Si l'on dispose d'une formule bien formée, on va pouvoir s'intéresser à sa valeur de vérité selon les « mondes » possibles, c'est-à-dire en remplaçant chacune des propositions atomiques qui la composent par VRAI ou FAUX. On a alors effectué une *interprétation* de la formule : c'est une fonction de l'ensemble des formules dans $\mathcal{B}_2$.

4.1 Valeurs sous un environnement

Définition 1 - 1

Une *distribution de vérité* (ou *environnement* ou *interprétation propositionnelle*) est une application v des variables atomiques de $\mathcal{F}$ dans $\mathcal{B}_2$.

La valeur de vérité d'une formule va dépendre de l'environnement dans lequel on décide d'évaluer la formule.
La *valeur booléenne* d'une formule f *sous l'environnement* v est définie récursivement par :

- $\mathcal{V}(\bot, v) = 0$;
- si p est une variable atomique alors $\mathcal{V}(p, v) = v(p)$;
- $\mathcal{V}(\neg p, v) = 1$ ssi, $\mathcal{V}(p, v) = 0$ (NÉGATION) ;
- $\mathcal{V}(p \wedge q, v) = 1$ ssi $\mathcal{V}(p, v) = \mathcal{V}(q, v) = 1$ (CONJONCTION) ;
- $\mathcal{V}(p \vee q, v) = 0$ ssi, $\mathcal{V}(p, v) = \mathcal{V}(q, v) = 0$ (DISJONCTION) ;
- $\mathcal{V}(p \rightarrow q, v) = 0$ ssi, $\mathcal{V}(p, v) = 1$ et $\mathcal{V}(q, v) = 0$ (IMPLICATION) ;
- $\mathcal{V}(p \leftrightarrow q, v) = 1$ ssi, $\mathcal{V}(p, v) = \mathcal{V}(q, v)$ (ÉQUIVALENCE),

avec p et q des éléments quelconques de $\mathcal{F}$.

Recherche

Soit la formule $f = (p \wedge (q \vee r))$ et l'environnement $\langle v(p) = 1, v(q) = 0, v(r) = 1\rangle$. Déterminez $\mathcal{V}(f, v)$.

$\mathcal{V}(f, \langle 1, 0, 1\rangle) = 1$

Remarque

Notez bien la différence entre $\bot$ et 0 ! $\bot$ est une proposition appartenant à l'alphabet de la logique des propositions alors que 0 est la valeur prise par cette proposition sous tous les environnements : c'est un élément de $\mathcal{B}_2$.
Par exemple, $\mathcal{V}(\bot, \langle 0, 1, 1\rangle) = 0$.

4 2 Avec Haskell

Sans entrer dans le détail, essayez de deviner ce qui se cache dans les lignes de code de l'annexe B page 123.

4 3 Tables de vérité

Lorsqu'une formule ne fait intervenir que peu de variables atomiques, on peut regrouper dans un tableau (une *table de vérité*) les 2^n (2 valeurs pour chacune des n variables) distributions de vérité possibles.
Considérons par exemple la formule $p \vee (\neg q \rightarrow p)$:

p	q	$\neg q$	$\neg q \rightarrow p$	$p \vee (\neg q \rightarrow p)$
1	1	0	1	1
1	0	1	1	1
0	1	0	1	1
0	0	1	0	0

4 4 Tautologies, formules satisfiables, insatisfiables

Un peu de vocabulaire....

Définition 1 - 2

Un *modèle* d'une formule donnée est un environnement pour lequel la formule est vraie.

Par exemple, $\langle v(p) = 1, v(q) = 0\rangle$ est un modèle de $p \vee (\neg q \rightarrow p)$.

Définition 1 - 3

Une formule est *satisfiable* si, et seulement si, elle admet au moins un modèle.

Par exemple $p \vee (\neg q \rightarrow p)$ est satisfiable.

Définition 1 - 4

Une formule f vraie pour toutes les interprétations de ses variables atomiques est une *tautologie*. On note alors $\models f$.

Par exemple, vérifiez que $\models (\neg p \vee p)$.

Définition 1 - 5

Une formule qui n'admet aucun modèle est dite *insatisfiable*.

Par exemple $\neg p \wedge p$ est insatisfiable.

4 5 Conséquences et équivalences logiques

Définition 1 - 6

Soit F une formule ou un ensemble de formules et G une formule. On dit que G est une *conséquence logique* de F si, et seulement si, tout modèle de F est aussi un modèle de G. On note alors $F \models G$.

Recherche

Prenons un exemple concret. Un juge vous dit :
« Je vous acquitte seulement si vous êtes innocent. Or vous n'êtes pas innocent donc je ne vous acquitte pas. »
Ce raisonnement est-il correct ?
Notons a la variable atomique : « Le juge vous acquitte » et i la variable : « vous êtes innocent ».
Si le juge vous acquitte, cela implique que vous êtes innocent (Le fait que vous soyez innocent est une condition nécessaire à l'acquittement) donc on a $a \rightarrow i$. De plus on sait que $\neg i$. La conséquence logique en est $\neg a$.
Le raisonnement du juge peut donc être modélisé par :

$$a \rightarrow i,\ \neg i \models \neg a$$

Est-il correct ? Il faut vérifier que tout modèle de $a \rightarrow i,\ \neg i$ est un modèle de $\neg a$.
Les modèles de $a \rightarrow i$ sont $\langle v(a) = 0, v(i) = 0\rangle$, $\langle v(a) = 0, v(i) = 1\rangle$ et $\langle v(a) = 1, v(i) = 1\rangle$.
Les modèles de $\neg i$ sont $\langle v(a) = 0, v(i) = 0\rangle$ et $\langle v(a) = 1, v(i) = 0\rangle$.
L'unique modèle de $a \rightarrow i,\ \neg i \models \neg a$ est donc $\langle v(a) = 0, v(i) = 0\rangle$.
Les modèles de $\neg a$ sont $\langle v(a) = 0, v(i) = 0\rangle$ et $\langle v(a) = 0, v(i) = 1\rangle$.
L'unique modèle de $a \rightarrow i,\ \neg i \models \neg a$ est donc aussi un modèle de $\neg a$.
Le raisonnement est donc correct : si on n'acquitte que les innocents, le fait d'être coupable entraîne d'être accusé.

Quid de celui-ci :
« Je vous acquitte seulement si vous êtes innocent. Or je ne vous acquitte pas donc vous n'êtes pas innocent »
Le raisonnement du juge est maintenant :

$$a \rightarrow i,\ \neg a \models \neg i$$

Or $\langle v(a) = 0, v(i) = 1\rangle$ est un des modèles de $a \rightarrow i,\ \neg a$ mais n'est pas un modèle de $\neg i$.
Le raisonnement n'est donc pas correct et est mis en défaut par la situation : « le juge n'a pas acquitté un innocent ».
Un système juridique où la défense doit prouver l'innocence permet donc d'accuser des innocents.
Vous pouvez vérifier qu'un système juridique où l'accusation doit prouver la culpabilité permet d'acquitter des coupables.

Remarque

Notez la différence entre $\rightarrow$ et $\models$!... En fait « $F \models G$ » signifie que « $F \rightarrow G$ est une tautologie » ou encore « $\models (F \rightarrow G)$ ».
Si F est un ensemble de formules, « $F_1, F_2, \ldots F_p \models G$ » signifie « $(F_1 \wedge F_2 \wedge \cdots \wedge F_p) \rightarrow G$ est une tautologie ».

Considérez la table suivante :

a	i	$a \rightarrow i$	$\neg i$	$(a \rightarrow i) \wedge \neg i$	$\neg a$	$((a \rightarrow i) \wedge \neg i) \rightarrow \neg a$
1	1	1	0	0	0	1
1	0	0	1	0	0	1
0	1	1	0	0	1	1
0	0	1	1	1	1	1

On remarque que $((a \rightarrow i) \wedge \neg i) \rightarrow \neg a$ est une tautologie donc que $((a \rightarrow i) \wedge \neg i) \models \neg a$ (i.e. $\neg a$ est une conséquence logique de $((a \rightarrow i) \wedge \neg i)$) ou encore $\models \big(((a \rightarrow i) \wedge \neg i) \rightarrow \neg a\big)$, ce qui confirme le raisonnement effectué au paragraphe précédent.

4 6 Formules d'équivalence de la logique des propositions

Définition 1 - 7

Soient F et G deux formules. On dit que F et G sont logiquement équivalentes si, et seulement si, $F \models G$ et $G \models F$. On note alors $F \equiv G$.

On peut prouver les différentes équivalences logiques suivantes à l'aide de tables de vérité par exemple :

Équivalence entre connecteurs	$p \to q \equiv \neg p \vee q$
	$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p) \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
Double négation	$\neg\neg p \equiv p$
Lois de De Morgan	$\neg(p \wedge q) \equiv \neg p \vee \neg q$
	$\neg(p \vee q) \equiv \neg p \wedge \neg q$
Idempotence	$p \vee p \equiv p \wedge p \equiv p$
Commutativité	$p \wedge q \equiv q \wedge p$
	$p \vee q \equiv q \vee p$
Associativité	$p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r \equiv p \wedge q \wedge r$
	$p \vee (q \vee r) \equiv (p \vee q) \vee r \equiv p \vee q \vee r$
Contradiction	$p \wedge \neg p \equiv \bot$
Tiers exclus	$p \vee \neg p \equiv \top$
Lois de domination	$p \vee \top \equiv \top$
	$p \wedge \bot \equiv \bot$
Lois d'identité	$p \vee \bot \equiv p$
	$p \wedge \top \equiv p$
Distributivité	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
Absorption	$p \vee (p \wedge q) \equiv p$
	$p \wedge (p \vee q) \equiv p$

4 7 Principe de déduction par réfutation de la conclusion

Voici un principe très utilisé qui utilise des équivalences du tableau précédent :

Théorème 1 - 3

Pour montrer que $P_1 \wedge P_2 \wedge \cdots \wedge P_n \models C$ il faut et il suffit que la *formule de réfutation* $P_1 \wedge P_2 \wedge \cdots \wedge P_n \wedge (\neg C)$ soit insatisfiable.

On dit alors qu'on a montré la validité par réfutation de la conclusion.
On montre ce théorème à l'aide d'équivalences du tableau précédent :

$$p \to q \equiv \neg p \vee q \text{ et } \neg p \vee q \equiv \neg(p \wedge \neg q) \text{ donc } p \to q \equiv \neg(p \wedge \neg q)$$

Ainsi, $p \to q$ est une tautologie si, et seulement si, $\neg(p \wedge \neg q)$ est une tautologie, i.e. $p \wedge \neg q$ est insatisfiable.

5 Approche formelle de la logique propositionnelle (logique niveau II...)

5 1 Principe général

Pour pouvoir effectuer une déduction logique uniquement à l'aide d'une analyse syntaxique (à l'aide d'un ordinateur par exemple), on utilise trois outils :

Axiome : c'est une proposition primitive, admise a priori et considérée comme non démontrable (Par exemple, en géométrie euclidienne, celle du collège et du lycée, on considère que par

deux points distincts il ne passe qu'une seule droite. On ne le démontre pas, on le pose comme axiome. Dans d'autres géométries, on n'utilise pas cet axiome et plusieurs droites peuvent passer par un même couple de points...).

Théorème : c'est une proposition obtenue à partir des axiomes ou d'autres théorèmes à l'aide de règles appliquées aux symboles la constituant (ce sont des règles portant sur la syntaxe).

Si T est un théorème, on note $\vdash T$.

Règle d'inférence schéma de raisonnement permettant de produire de nouveaux théorèmes à partir de *prémisses* qui sont soit des théorèmes, soit des hypothèses.

Si $P_1, P_2, ..., P_n$ sont les prémisses et T le théorème, on note $P_1, P_2, ..., P_n \vdash T$.

Voyons tout de suite un exemple :

« Si je travaille en cours, j'aurai une bonne note. Or je travaille en cours donc j'aurai une bonne note. »

Ce raisonnement est-il correct ? Représentons le schéma de raisonnement sous une forme canonique.

Notons C : « Je travaille en cours » et N : « j'ai une bonne note ».

$$\begin{array}{ll} (P_1) & C \rightarrow N \\ (P_2) & C \\ \hline (T) & N \end{array}$$

Cette règle d'inférence est très importante et porte le nom latin de *modus ponens* (en latin, *ponere* veut dire « poser ». Cela veut donc dire « posant C on a N » ;-).

On peut donc écrire :

$$C \rightarrow N,\ C \vdash N$$

Remarque

Notez bien la différence avec $C \rightarrow N,\ C \models N$! Pour l'obtenir, on utilise des tables de vérité, on étudie donc la sémantique des propositions. Dans le cas de $C \rightarrow N,\ C \vdash N$, on *regarde* juste si le schéma correspond à l'une de nos règles, en l'occurrence ici le *modus ponens*. Cette dernière façon de prouver convient bien à une machine.
Quand l'approche formelle rejoint l'approche sémantique, on a un système logique qui est dit *consistant* et *complet*. C'est le cas de la logique des propositions mais ce n'est pas le cas de toutes les logiques, mais ceci est une autre histoire...
Ainsi, dans le cadre de votre cours de mathématiques, on a $A \models B$ si, et seulement si, $A \vdash B$.

Voyons maintenant un autre exemple :

« Si je travaille en cours, j'aurai une bonne note. Or j'ai une mauvaise note donc je ne travaille pas en cours ».

Avec les notations précédentes, on obtient :

$$\begin{array}{ll} (P_1) & C \rightarrow N \\ (P_2) & \neg N \\ \hline (T) & \neg C \end{array}$$

Ce schéma est correct et s'appelle *modus tollens* (en latin, *tollere* signifie enlever).

Voici un petit tableau des principales règles dans le système de HILBERT où A, B et C sont des formules quelconques :

Règle de combinaison	$A, B \vdash A \wedge B$
Règle de simplification	$A \wedge B \vdash B$
Règle d'addition	$A \vdash A \vee B$
Modus ponens	$A, A \rightarrow B \vdash B$
Modus tollens	$\neg B, A \rightarrow B \vdash \neg A$
Syllogisme hypothétique	$A \rightarrow B, B \rightarrow C \vdash A \rightarrow C$
Syllogisme disjonctif	$A \vee B, \neg B \vdash A$
Règle des cas	$A \rightarrow B, \neg A \rightarrow B \vdash B$
Élimination de l'équivalence	$A \leftrightarrow B \vdash A \rightarrow B$
Introduction de l'équivalence	$A \rightarrow B, B \rightarrow A \vdash A \leftrightarrow B$
Règle d'inconsistance	$A, \neg A \vdash B$

Remarque

La dernière règle est rigolote : avec des prémisses absurdes, on peut démontrer n'importe quoi...

Remarque

Il existe d'autres systèmes formels de la LP, notamment le système de LUKASIEWICZ qui n'a que trois axiomes :

1. $A \rightarrow (B \rightarrow A)$
2. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (B \rightarrow C))$
3. $(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$

avec comme unique règle d'inférence le *modus ponens*.

5 2 Théorème de la déduction

L'utilisation de ces règles est grandement facilitée par le théorème suivant que nous admettrons : si on a une démonstration d'une formule B utilisant une hypothèse A alors on peut construire une démonstration de $A \rightarrow B$ qui n'utilise plus cette hypothèse...en d'autres termes :

Théorème 1 - 4

Théorème de la déduction
Si $A, P \vdash B$, alors $P \vdash A \rightarrow B$.

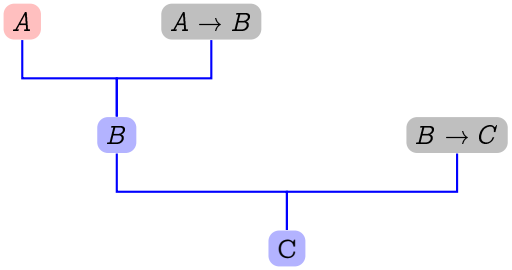
L'algorithme est le suivant :

1. On fait l'*hypothèse* de A : on le rajoute temporairement aux prémisses ;
2. on démontre B en utilisant A et le reste des prémisses ;
3. on fait abstraction de A : A n'est plus forcément valide mais on obtient la conclusion $A \rightarrow B$.

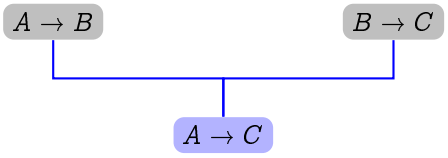
Démontrons par exemple le syllogisme hypothétique en utilisant uniquement le *modus ponens* (MP).
Les prémisses au départ sont $A \rightarrow B$ et $B \rightarrow C$:

1. $A \rightarrow B$ (première prémisse)
2. $B \rightarrow C$ (deuxième prémisse)
3. A (on rajoute temporairement A aux prémisses : on fait l'hypothèse de A)
4. $A, A \rightarrow B \vdash B$ d'après le MP en utilisant 3. et 1. : on peut mettre B dans les prémisses
5. $B, B \rightarrow C \vdash C$ d'après le MP en utilisant 4. et 2.
6. on fait abstraction de A : A n'est plus forcément valide mais on obtient la conclusion $A \rightarrow C$.

L'emploi d'un *arbre* binaire *de déduction* est peut-être plus clair. Les nœuds sont des prémisses ou des arbres de déduction. La racine est la conclusion. Le théorème de déduction consiste donc à rajouter la feuille A puis à la couper et remplacer la racine B par $A \rightarrow B$:

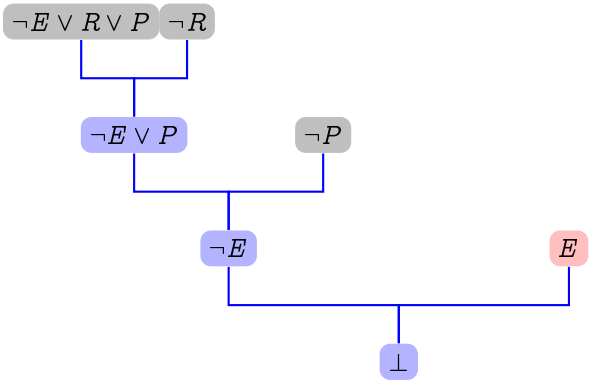


donne le nouveau théorème :



5 3 Retour sur la démonstration par réfutation

On peut s'inspirer de cette méthode pour retrouver le principe de démonstration par réfutation. Prenons par exemple le raisonnement suivant émis par David VINCENT : « Si John est un envahisseur, il ne rigolera pas à mes blagues ou il aura le petit doigt de la main droite écarté. John a rigolé à mes blagues et a le petit doigt serré contre l'annulaire. Ce n'est donc pas un envahisseur »
Ajoutons dans les prémisses la réfutation de la conclusion et voyons ce qui se passe...
Soit E : « John est un envahisseur », R : « il ne rigolera pas de mes blagues », P :« il a le petit doigt écarté ».
Les prémisses sont donc $E \to (R \vee P)$, $\neg R$, et $\neg P$ et la conclusion $\neg E$.
Rappelons le principe de la réfutation : pour montrer que $P_1, P_2, ..., P_n \models C$ il faut et il suffit que la *formule de réfutation* $P_1 \wedge P_2 \wedge \cdots \wedge P_n \wedge (\neg C)$ soit insatisfiable.
Nous allons donc mettre nos prémisses sous forme conjonctive.
Rappelons que $A \to B \equiv \neg A \vee B$ donc, dans notre cas d'étude, $E \to (R \vee P) \equiv \neg E \vee (R \vee P) \equiv \neg E \vee R \vee P$. Tout est donc prêt...Rajoutons la négation de la conclusion à nos prémisses, à savoir E.



La conclusion étant $\bot$, notre hypothèse E est donc fausse et heureusement pour David VINCENT, John n'est pas un envahisseur.

Remarque

Un raisonnement effectué à l'aide d'un arbre : c'est magique non ? En tout cas, ça rend la pensée programmable sur une machine...

5 4 Raisonnement par l'absurde / par contraposée

« *Avec le mot SI on peut faire tout ce qu'on ne peut pas faire.* »

in *Y'a du mou dans la corde à noeuds* de Pierre DAC

On emploie parfois (et même souvent...) en classe d'autres termes que ceux utilisés dans ce chapitre.

— La démonstration par **contraposée** est en fait le *modus tollens* :

$$\begin{array}{ll} (P_1) & C \to R \\ (P_2) & \neg R \\ \hline (T) & \neg C \end{array}$$

R est une condition nécessaire de C. Si R n'est pas vérifiée alors C ne peut pas l'être.

Par exemple, si R est « ce quadrilatère est un rectangle » et C « ce quadrilatère est un carré ».

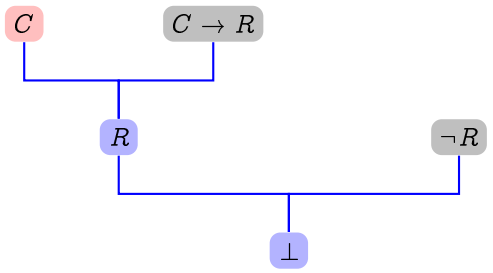
On a bien $C \to R$ comme prémisse. Si l'on démontre que le quadrilatère n'est pas un rectangle, on pourra en déduire que ce n'est pas un carré.

On utilise juste l'équivalence des contraposées : $C \to R \equiv \neg R \to \neg C$.

On démontre $\neg C$ avec comme seule hypothèse $\neg R$.

— La démonstration par l'**absurde** est plus compliquée car il faut rajouter une hypothèse.

On a toujours comme hypothèse $\neg R$ et on rajoute à nos prémisses la négation de notre supposée conclusion à savoir l'hypothèse $\neg(\neg C)$, c'est-à-dire C.



La conclusion étant $\bot$, l'hypothèse C est donc fausse et d'après la règle du tiers exclus, $\neg C$ est vraie.

On suppose que le quadrilatère n'est pas un rectangle. On suppose de plus que c'est un carré. On arrive à une contradiction. L'hypothèse « est un carré » est donc fausse : le quadrilatère n'est pas un carré.

La faiblesse du raisonnement par l'absurde est qu'il faut rajouter une hypothèse et de plus supposer vrai quelque chose que l'on sait faux. Comme le disait encore Pierre Dac dans ses « *Pensées* » parues en 1972 : « *Si ma tante en avait on l'appellerait mon oncle* ».
C'est pourquoi on préfère n'utiliser ce raisonnement que si l'on ne peut pas faire autrement.

6 Récurrence

6.1 Retour sur le raisonnement par récurrence et sur le sens de l'implication

Souvenez-vous du test initial introduisant ce chapitre : « si n est pair alors son successeur est premier ».
Pourquoi a-t-on rajouté les nombres impairs à l'ensemble des entiers rendant cette proposition vraie ?
Parce que $A \to B$ est toujours vraie sauf dans le cas A est vraie et B est fausse. En particulier $A \to B$ reste vraie quand A est fausse.
Ceci est primordial dans la démonstration par récurrence, en particulier dans le preuve de l'hérédité.
Soit P_n une proposition dépendant d'un entier n.
On démontre l'hérédité $P_n \to P_{n+1}$ en utilisant le théorème de la déduction mais le fait que $P_n \to P_{n+1}$ soit vraie n'assure pas que P_n le soit.
Il faut de plus s'assurer que la propriété est vraie au moins une fois.
Ce type de raisonnement n'est donc pas une escroquerie comme certains élèves le pensent trop souvent : « ah oui, on suppose que c'est vrai au rang n et on en déduit que c'est vrai au rang n... ».

6 2 Récursion, récurrence, induction

...des mots qui tournent autour du même thème mais qu'il faut savoir distinguer.
L'*induction* consiste à conclure à partir d'observations préalables : cela correspond à la démarche expérimentale classique. Sans précautions, cela peut conduire à certaines contre-vérités. Par exemple 3, 5 et 7 sont premiers donc on pourrait en déduire, par induction, que tous les nombres impairs à partir de 3 sont premiers et pourtant...
L'*induction mathématique* ou *raisonnement par récurrence* corrige ce défaut. On part toujours d'un résultat observé mais on *démontre* qu'il est vrai pour tous les éléments d'un ensemble donné, éventuellement infini. C'est l'induction expérimentale corrigée par la rigueur du raisonnement mathématique.
En informatique, une *fonction récursive* (ou un *type récursif*) est une fonction (ou un type) qui fait référence à elle(lui)-même dans sa définition. Ce mécanisme est très puissant et permet de condenser l'écriture d'un programme.

C'est l'induction mathématique qui nous intéresse ici. Rappelons quelques principes.
Pour prouver qu'une propriété $\mathcal{P}_n$ dépendant uniquement d'un paramètre n est vraie pour tout $n \geqslant n_0$, il faut vérifier que :

— $\mathcal{P}_{n_0}$ est vraie (on parle parfois d'initialisation) ;

— pour tout $n \geqslant n_0$, $\mathcal{P}_n \to \mathcal{P}_{n+1}$ (on parle parfois d'hérédité).

Quand on a besoin de supposer que non seulement la propriété est vraie pour un certain n mais aussi pour tous les entiers qui lui sont inférieurs (pas seulement le père mais aussi tous les ascendants) on parle alors de *récurrence forte*.
Voyons un exemple...

6 3 Génétique syldave

Les scientifiques syldaves viennent de mettre en évidence que la terrible maladie de Mathieu est en fait héréditaire : cette maladie frappe depuis des siècles les petits syldaves et les fait naître avec un unique mais énorme cheveu sur la tête.
C'est Vaclav Grtschtsz qui, le premier, contracta cette maladie en 1643 après être rentré en contact avec des vénusiens : ce fait peu connu marque la cause de l'apparition de la maladie en Syldavie. Depuis, tous ses descendants ont souffert de ce terrible mal et aucun médicament terrestre ne semble en mesure de stopper cette calamité.
Résumons les faits :

1. la maladie de Mathieu fait naître les nouveaux nés avec un énorme et unique cheveu sur la tête. Notons n la n^{e} génération aprés Vaclav.
   Notons $\mathcal{P}(n)$ la propriété : « la n^{e} génération sera infectée par la maladie ».
2. initialisation : un premier syldavien est infecté en 1643, donc $\mathcal{P}(0)$ est vraie ;
3. l'hérédité de la maladie a été prouvée : si un des parents de la k^{e} génération est atteint, alors ses enfants de la $k+1^{\text{e}}$ génération seront également infectés, ce qui se traduit par
   $$\mathcal{P}(k) \to \mathcal{P}(k+1)$$
4. nous en déduisons que, quelque soit la génération n des descendants de Vaclav, ceux-ci seront infectés, c'est à dire que $\mathcal{P}(n)$ est vraie quelque soit l'entier naturel n.

EXERCICES

Exercice 1 - 1

Quelles sont les sous-formules de :
$s \rightarrow ((p \wedge (\neg q)) \vee (p \leftrightarrow (\neg r)))$

Exercice 1 - 2

Remplissez la table suivante :

p	m	$p \rightarrow m$	$\neg m$	$(p \rightarrow m) \wedge \neg m$	$\neg p$	$((p \rightarrow m) \wedge \neg m) \rightarrow \neg p$
1	1					
1	0					
0	1					
0	0					

Exercice 1 - 3

Prouvez, à l'aide de tables de vérité, la première loi de De Morgan et la distributivité de la disjonction sur la conjonction.

Exercice 1 - 4

L'implication est-elle associative ?

Exercice 1 - 5

Écrire une fonction $(x, y) \mapsto x \wedge y$ avec un « Si...ALORS...SINON »

Exercice 1 - 6

Déterminez une table de vérité du connecteur IFTE défini sur $\mathcal{B}_2^3$ par $IFTE(x, y, z) = 1$ si, et seulement si, `SI x vrai ALORS y vrai SINON z vrai` est valide.

Exercice 1 - 7

Soit p la proposition « Je mâche du foin » et q la proposition « Je mâche du grain ». Traduire par une phrase en français :

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

Exercice 1 - 8

On considère les propositions atomiques B, T, V, C représentant les assertions suivantes : « Hélène est belle », « Hélène joue au Tennis », « Hélène fait du vélo », « Chri-Chri aime Hélène ». Énoncez des phrases simples traduisant les propositions suivantes :

1. i. $\neg B$
   ii. $\neg B \wedge \neg T$
   iii. $\neg(B \vee T)$
   iv. $(B \wedge V) \rightarrow C$
   v. $C \rightarrow (B \leftrightarrow V)$
   vi. $((B \vee T) \wedge \neg V) \rightarrow \neg C$
2. Traduisez par une proposition simple les phrases
   i. « Chri-Chri aime Hélène seulement si elle fait du vélo ».
   ii. « Chri-Chri aime Hélène si elle fait du vélo ».
   iii. « Chri-Chri aime Hélène si, et seulement si, elle fait du vélo ».
   iv. « Si Chri-Chri aime Hélène alors elle fait du vélo ».
   v. « Si Hélène fait du vélo alors Chri-Chri l'aime ».
   vi. « Il est suffisant qu'Hélène fasse du vélo pour que Chri-Chri l'aime ».
   vii. « Il est nécessaire qu'Hélène fasse du vélo pour que Chri-Chri l'aime ».

Exercice 1 - 9

Commentez les phrases suivantes en fonction de la logique des propositions :

1. Si les poules ont des dents alors $2+2=5$;
2. Si $2+2=5$ alors les poules n'ont pas de dents ;
3. Si $2+2=5$ alors les poules ont des dents.

Exercice 1 - 10

On considère les deux propositions :

— A :« Si je fait du vélo alors Chri-Chri m'aime » ;

— B : « Si Chri-Chri m'aime alors je ne fais pas de vélo ».

La proposition $A \wedge B$ est-elle satisfiable ?

Exercice 1 - 11

Réécrire les phrases suivantes pour qu'elles soient équivalentes au sens de la logique des propositions en utilisant « suffisant » ou « nécessaire » :

1. Si ses poules ne se lavent pas les dents, il ne vient pas en cours.
2. Si je range ma chambre, Hélène ne m'aimera pas.

Exercice 1 - 12

Déterminez les contraposées, les réciproques puis les négations des propositions suivantes :

1. Si Max n'étudie pas son cours de maths, les filles le fuient.
2. Bill embrassera Céleste seulement s'il écrit de bons programmes en Haskell
3. Paul marquera un but seulement s'il lit la vie de Carl Gauss en latin.

Exercice 1 - 13

> « In summary, I would like to convey a little positive message to you...I do not have any...Would two negative messages suit you ? »
>
> Woody Allen

Reformulez la phrase suivante sans négations, prononcées lors du grand meeting du PUB, Parti Unique Bordure, afin de choisir l'unique candidat des futurs élections dictatoriales.
Dieter de Villenstein : « Il faut soutenir Josef Chrtzw. Et soutenir l'action de M. Chrtzw, ce n'est pas ne pas soutenir notre ami à tous : Nikalaï Schrpuntz ».

Exercice 1 - 14

Toute formule de $\mathcal{F}$ a autant de parenthèses ouvrantes que de parenthèses fermantes.

Exercice 1 - 15

Montrez que le produit d'un irrationnel par un rationnel non nul est toujours un irrationnel. Avez-vous raisonné par l'absurde ou par contraposée ?

Exercice 1 - 16

1. Écrire $x \wedge \neg(\neg y \wedge z)$ sous forme normale disjonctive.
2. Écrire $\neg(\neg(x \wedge y) \wedge z) \wedge \neg((\neg x \vee z) \wedge (\neg y \vee \neg z))$ sous forme normale (disjonctive ou conjonctive, à vous de choisir...)

Exercice 1 - 17

Écrire les formules suivantes sous forme normale disjonctives et dire s'il s'agit de tautologies :

1. $p \wedge q \wedge r \to p \vee q$
2. $((p \to q) \wedge (q \to r)) \to (p \to r)$
3. $(p \to q) \to p$
4. $(p \leftrightarrow (q \vee r)) \to (\neg q \to p \vee r)$

Exercice 1 - 18

— L'opérateur $\downarrow$ est défini par $1 \downarrow 1 = 1 \downarrow 0 = 0 \downarrow 1 = 0$ et $0 \downarrow 0 = 1$ qu'on appellera « NOR » ;

— L'opérateur $\uparrow$ est défini par $1 \uparrow 0 = 0 \uparrow 1 = 0 \uparrow 0 = 1$ et $1 \uparrow 1 = 0$ qu'on appellera « NAND ».

Montrer que :

1. $\neg x \equiv x \uparrow x \equiv x \downarrow x$
2. $x \wedge y \equiv (x \uparrow y) \uparrow (x \uparrow y)$
3. $x \vee y \equiv (x \uparrow x) \uparrow (y \uparrow y)$
4. $x \wedge y \equiv (x \downarrow x) \downarrow (y \downarrow y)$
5. $x \vee y \equiv (x \downarrow y) \downarrow (x \downarrow y)$

Exercice 1 - 19

Le connecteur$\oplus$ est défini par : $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 1 = 1$ et $0 \oplus 0 = 0$.

1. Simplifier :

i. $x \oplus 0$ **ii.** $x \oplus 1$ **iii.** $x \oplus x$ **iv.** $x \oplus \neg x$

2. Montrer que :

i. $x \oplus y \equiv (x \vee y) \wedge \neg (x \wedge y)$ **ii.** $x \oplus y \equiv (x \wedge \neg y) \vee (\neg x \wedge y)$

3. L'opération $\oplus$ est-elle commutative ?

4. Vrai ou faux ?

i. $x \oplus (y \oplus z) \equiv (x \oplus y) \oplus z$ **ii.** $x \vee (y \oplus z) \equiv (x \vee y) \oplus (x \vee z)$ **iii.** $x \oplus (y \vee z) \equiv (x \oplus y) \vee (x \oplus z)$

Exercice 1 - 20 Système complet

Sachant que $\{\neg, \wedge, \vee\}$ est un scc, montrez que $\{\neg, \rightarrow\}$ est lui aussi un scc (ce qui sera utile pour le système formel de ŁUKASIEWICZ (philosophe polonais (1878 - 1956))...). $\{\wedge, \vee\}$ est-il un scc ?
$\{\uparrow\}$ est-il un scc ?

Exercice 1 - 21 Demi-additionneur

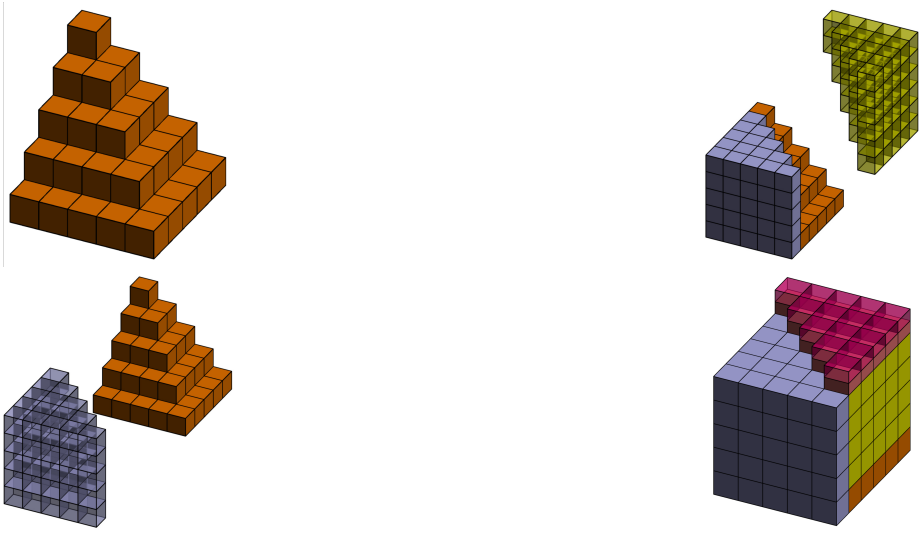
On veut additionner deux nombres binaires de un bit. La sortie sera double : une variable u contiendra l'« unité » et une variable r contiendra la « retenue ». Dresser un tableau puis dessiner un circuit ayant deux sorties avec les portes NOT, AND et OR.

Exercice 1 - 22 Divisibilité booléenne

Pour représenter les nombres en base 2, on va utiliser quatre atomes b_3, b_2, b_1, b_0 correspondant chacune aux bits de poids décroissants. Par exemple, $\langle 11 \rangle_{10} = \langle 1011 \rangle_2$ sera représenté par $v(b_3) = 1$, $v(b_2) = 0$, $v(b_1) = 1$ et $v(b_0) = 1$. Déterminez une formule qui est vraie si $\langle b_3 b_2 b_1 b_0 \rangle_2$ est divisible par 4 ou 5. Vous donnerez votre réponse sous fnc ou fnd.

Exercice 1 - 23

Que vous inspirent les petits dessins suivants :



Exercice 1 - 24

Les images des entiers naturels par la fonction $p : n \mapsto n^2 - n + 41$ sont des nombres premiers.

Exercice 1 - 25

On se propose de démontrer que tous les étudiants d'IUT ont le même âge et, pour cela, on note $P(n)$ l'affirmation « *si on choisit n étudiants ($n \in \mathbb{N}^*$), il est sûr qu'ils ont tous le même âge* »
Il est clair que $P(1)$ est vraie.
Démontrons que $P(n) \rightarrow P(n+1)$. Pour cela nous supposons que $P(n)$ est vraie (c'est l'hypothèse de récurrence) et nous choisissons un groupe quelconque de $n+1$ étudiants que nous ordonnons par ordre alphabétique (pourquoi pas). D'après l'hypothèse de récurrence, les n premiers de l'ordre alphabétique ont tous le même âge ainsi que les n derniers. Comme ces deux groupes de n étudiants ont au moins un étudiant en commun, on en déduit qu'ils ont tous le même âge.
Nous venons de démontrer que $P(n) \rightarrow P(n+1)$ pour tout $n \geq 1$ et, comme $P(1)$ est vrai, $P(n)$ est toujours vrai pour tout $n \geqslant 1$. **Y a-t-il une erreur dans le raisonnement ?**

Exercice 1 - 26

Vous allez voir votre médecin en espérant qu'il a suivi des cours de logique.
Voici la situation :

— Les gens qui ont la maladie de Schplurz doivent prendre le médicament Xlurbix ;

— Les gens qui ont de la fièvre et des poils qui poussent au fond de la gorge ont la maladie de Schplurz ;

— Ceux qui ont une température supérieure à 38° sont considérés comme ayant de la fièvre ;

— Vous avez des poils dans la gorge et 39,5° de température.

Le médecin va-t-il vous prescrire du Xlurbix ?

Exercice 1 - 27

C'est bientôt la fin de l'épisode de « Police squad ». L'inspecteur Franck Drebin livre sa version des faits au juge pour savoir qui de Vincent Ludwig ou de Quentin Hapsburg est le véritable auteur du meurtre de la reine.
« *Si Vincent n'a pas rencontré Quentin l'autre nuit, c'est que Quentin est le meurtrier ou Vincent est un menteur. Si Quentin n'est pas le meurtrier, alors Vincent n'a pas rencontré Quentin l'autre nuit et le crime a eu lieu après minuit. Si le crime a eu lieu après minuit, alors Quentin est le meurtrier ou Vincent n'est pas un menteur. Donc Quentin est le meurtrier.* »
L'inspecteur Drebin est-il un bon logicien ?

Exercice 1 - 28

Les jeux virtuels sur ordinateur font souvent appel à des énigmes logiques régies par le calcul des propositions. Vous participez actuellement à une partie dont les règles sont les suivantes :
Les propositions composant une énigme sont alternativement vraies et fausses, c'est-à-dire que :

— *soit les propositions de numéro pair sont vraies et les propositions de numéro impair fausses ;*

— *soit les propositions de numéro pair sont fausses et les propositions de numéro impair vraies.*

Dans un labyrinthe, vous vous retrouvez bloqué dans une salle face à une porte sur laquelle se trouvent deux interrupteurs etiquetés A et B en position ouverte. Sur son seuil figure l'inscription suivante :
Pour ouvrir la porte :

P1 *Il faut fermer l'interrupteur A.*

P2 *Il faut fermer simultanément les interrupteurs A et B.*

P3 *Il ne faut pas fermer l'interrupteur B.*

Attention, en cas d'erreur la salle s'auto-détruira...

1. Exprimer P1 , P2 et P3 sous la forme de formules du calcul des propositions dépendant de A et de B.
2. Exprimer la règle du jeu dans le contexte des propositions P1 , P2 et P3.
3. En utilisant le calcul des propositions, déterminer l'action à effectuer pour ouvrir la porte.

Exercice 1 - 29 Sujet CCP 2011

Dans un futur lointain, l'espèce humaine a découvert une autre espèce consciente. L'étude de cette espèce a permis de découvrir qu'elle est capable de percevoir si quelqu'un dit la vérité ou un mensonge. Les membres de cette espèce respectent les règles de politesse suivantes lors des discussions au sein d'un groupe : « Les orateurs doivent rester constants au cours d'une discussion : soit ils disent toujours la vérité, soit ils mentent toujours. De plus, si un orateur dit la vérité alors l'orateur suivant doit également dire la vérité. Si le sujet de la discussion change, les orateurs sont libres de changer leurs comportements. ».
Vous assistez à une discussion sur les moyens d'attaque et de défense que peut posséder la faune de cette planète entre trois membres de cette espèce que nous appellerons A, B et C.
A : « Le kjalt peut avoir un dard ou des griffes. »
B : « Non, il n'a pas de dard. »
C : « Il a des pinces et des griffes. »
Nous noterons D, G et P les variables propositionnelles associées au fait qu'un kjalt possède respec- tivement un dard, des griffes et des pinces.
Nous noterons A1 , B1 et C1 les formules propositionnelles associées aux déclarations de A, B et C dans cette première discussion.
C quitte le groupe et la discussion change de sujet pour parler de la flore de la planète.
A : « Un lyop peut être de couleur mauve mais pas de couleur jaune. »
B : « Il ne peut pas être de couleur verte. »

A : « Il ne peut être de couleur verte que s'il peut être de couleur jaune. »
Nous noterons J, M et V les variables propositionnelles associées au fait qu'un lyop peut être respectivement de couleur jaune, mauve et verte.
Nous noterons A2 , A3 et B2 les formules propositionnelles associées aux déclarations de A et B dans cette seconde discussion.

1. Représenter les règles de politesse appliquées à la première discussion sous la forme d'une formule du calcul des propositions dépendant des formules A1 , B1 et C1 .
2. Représenter les informations données par les participants de la première discussion sous la forme de formules du calcul des propositions A1 , B1 et C1 dépendant des variables D, G et P.
3. En utilisant le calcul des propositions (résolution avec les formules de De Morgan), déterminer le (ou les) moyen(s) d'attaque et de défense que peut posséder un kjalt.
4. Représenter les règles de politesse appliquées à la seconde discussion sous la forme d'une formule du calcul des propositions dépendant des formules A2 , A3 et B2 .
5. Représenter les informations données par les participants lors de la seconde discussion sous la forme de trois formules du calcul des propositions A2 , A3 et B2 dépendant des variables J, M et V .
6. En utilisant le calcul des propositions (résolution avec les tables de vérité), déterminer la (ou les) couleur(s) possible(s) pour un lyop.

CHAPITRE

2

Ensembles

En 1934, de jeunes mathématiciens français se réunissent dans un café du quartier latin. Quatre ans plus tard, le groupe *Bourbaki* publie le premier chapitre des « *Éléments de mathématique* » qui bouleversera l'enseignement de la mathématique au XX^{e} siècle. En particulier, dans l'introduction, le groupe précise « *qu'autrefois, on a pu croire que chaque branche des mathématiques dépendait d'intuitions particulières [...] ce qui eût entraîné pour chacune la nécessité d'un langage formalisé qui lui appartînt en propre. On sait aujourd'hui qu'il est possible, logiquement parlant, de faire dériver presque toute la mathématique d'une source unique, la* **Théorie des Ensembles**. »

Ils introduisent ensuite ce qu'ils appellent *la méthode axiomatique* qui est « *l'art de rédiger des textes dont la* **formalisation** *est facile à concevoir.* »

Cette méthode « *permet [alors] lorsqu'on a affaire à des êtres mathématiques complexes, d'en dissocier les propriétés et de les regrouper autour d'un petit nombre de notions, c'est-à-dire [...] de les classer selon les* **structures** *auxquelles elles appartiennent.* ».

Dix ans plus tard, les premiers ordinateurs apparaissent, dix ans après, le premier véritable langage *structuré*, Fortran, est introduit, notamment sous l'égide de **John Backus** qui en fait vingt ans après une sévère critique dans un célèbre article « *Can programming be liberated from the Von Neumann style ? A* **functionnal** *style and its* **algebra** *of programs* ».

Enfin, depuis une dizaine d'années, le paradigme fonctionnel est en plein essor car, comme l'avait précisé Backus dès 1978, il permet de combler de grosses lacunes des langages traditionnels.

L'informatique a donc une histoire, fortement liée à celle de la mathématique et c'est cette histoire commune que nous allons explorer à partir d'aujourd'hui pour vous permettre de devenir les informaticen(ne)s de demain...

1 L'informaticien(ne) sur le terrain

Imaginons que vous effectuiez votre stage au service de l'état civil de la mairie de Klow. On vous demande de construire un logiciel qui va permettre aux employé(e)s de gérer les données de l'ancien fichier papier.
Pour permettre de vous organiser, vous dressez des petits schémas du style :

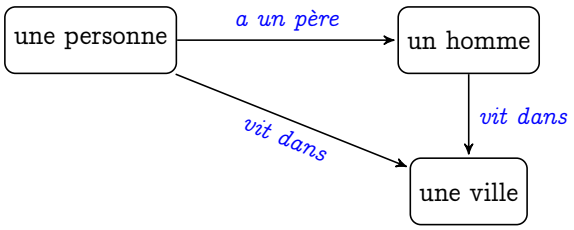


Schéma 1

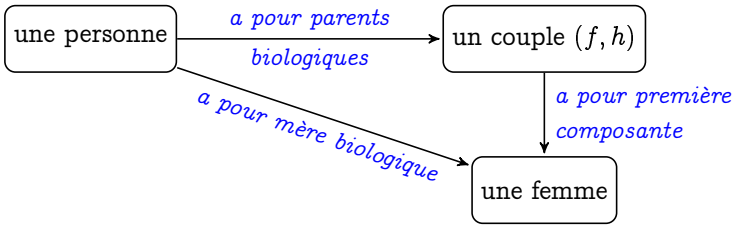


Schéma 2

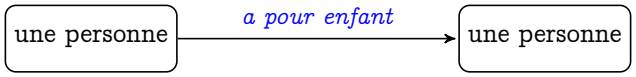


Schéma 3

Que vous inspire ces différents schémas ? Leurs différences ? Leurs points communs ?
Pour régler ces problèmes, l'informaticien(ne) va devoir *formaliser, spécifier, abstraire*, faire appel aux notions mathématiques d'*ensemble, produit cartésien, relation, fonction, composition, structure* et quand il sera très fort il pourra parler de *catégorie*.
Il faudra ensuite penser à une *implémentation* dans un *langage* donné, en construisant au besoin des *types*.
Il sera même utile de *compter* les données collectées et il ne faut pas oublier que l'ordinateur est un « *computer* » pour qui tout est nombre...
Plus généralement, l'informatique est une science (jeune) et l'informaticien(ne) doit donc employer une *démarche scientifique* vis-à-vis des programmes, algorithmes, etc. : la mathématique, une science bien plus ancienne et bien « rodée », lui fournit un modèle utile et même indispensable.

2 Tout est ensemble

Tous ensemble, tous ensemble, hé, hé
Tous ensemble, tous ensemble, hé, hé
Tous ensemble, tous ensemble, hé, hé

Nantes, Rue de Strasbourg,13 Mai 2003

2 1 Une définition ?

La notion d'ensemble peut paraître évidente : le dictionnaire parle de « groupe, assemblage d'éléments formant un tout ou ayant les mêmes caractéristiques ». Les mathématiques du XX e

siècle ont pour base la *théorie des ensembles* : il s'agit donc d'un concept de base et même DU concept de base, encore plus basique que la notion de nombre car nous verrons que les nombres entiers peuvent être définis et construits à l'aide d'ensembles.

Georg CANTOR
(1845 - 1918)

Pourtant, une définition correcte est très délicate à donner : comme souvent, plus la notion paraît simple, plus il est compliqué de la définir. L'adage « *plus c'est simple, plus c'est compliqué* » est d'ailleurs une des grandes frustrations en informatique : il faut expliquer des choses qui paraissent évidentes à des machines stupides...Pourtant, c'est aussi une chance : réfléchir et bien définir chaque notion est une contrainte mais cela permet à l'informaticien(ne) de pouvoir avancer avec confiance.

La définition usuelle due à ZERMELO et FRAENKEL (début du XX^{e} siècle) est bien trop compliquée à notre niveau et nous utiliserons plutôt la *théorie naïve* introduite de manière révolutionnaire à la fin du XIX^{e} par CANTOR.

2 2 Kit de survie en logique des prédicats

Pour construire notre théorie mathématique, il faut passer par tout un stade assez aride de mise au point des « briques » sur lesquelles tout va se construire, sachant qu'on part de preque rien : les premiers circuits électroniques de l'ordinateur...

Nous passerons cette étape fondamentale mais difficile : il faudra cependant être conscient de son existence.

Nous allons voir rapidement quelques notions supplémentaires de logique dont nous avons besoin tout de suite.

Au tout début du début, une théorie mathématique comporte des règles qui permettent de dire si des assemblages de signes sont des relations, des termes ou des théorèmes de la théorie...Mmmmouais...

Bon, disons que l'on considère un « tout » contenant des « objets » distincts. On peut vérifier si certains de ces « objets » vérifient ou non une « propriété ».

Par exemple « être un garçon » est une propriété vérifiable si l'on considère l'amas d'étudiant(e)s d'INFO 1 : on peut répondre par oui ou non à la question « truc est un garçon ».

On peut alors former l'ensemble G des garçons d'INFO 1 : ses *éléments* sont ceux qui vérifient la propriété. On peut noter alors :

$$G = \{ \text{étudiant} \mid (\text{étudiant est en INFO1}) \text{ ET } (\text{étudiant est un garçon}) \}$$

On notera indistinctement « Robert appartient à G », « Robert est un élément de G », « G contient Robert », « Robert $\in G$ », « $G \ni$ Robert ».

Notations

La notation $\in$ est due au mathématicien anglais Bertrand RUSSEL en 1903 qui reprenait l'epsilon ε de l'italien Giuseppe PEANO en 1889 mais en « l'arrondissant » compte-tenu des contraintes typographiques anglaises sous la forme ϵ qui a donné $\in$.

Par exemple « *est un entier pair* » est une propriété définie sur l'ensemble $\mathbb{N}$ des entiers naturels que nous noterons P. Alors 2 vérifie P mais pas 3.
Ces notions s'introduiront naturellement sur notre futur ami Haskell dont nous parlerons bientôt :

Haskell

```
naturels = [0..]
pairs = [n | n <- naturels, mod n 2 == 0]
petits_pairs = [n | n <- [0..15], mod n 2 == 0]
petits_pairs_bis = [2*k | k <- [0..7]]
```

Exemple

Alors on obtient par exemple :

Haskell

```
Prelude> petits_pairs
[0,2,4,6,8,10,12,14]
```

On notera aussi $2 \in$ `pairs` ou encore $2 \in \{x|(x \in \mathbb{N}) \wedge P(x)\}$ ou encore $2 \in \{t|(t \in \mathbb{N}) \wedge P(t)\}$ ou encore $P(2)$ ou encore $2 \in \{x|x \text{ est un entier naturel et } x \text{ est pair}\}$.
Ainsi $P(x)$ signifie que l'objet x vérifie la propriété P. Pour signifier que x ne vérifie pas P, on notera $\neg P(x)$ (« non P de x »).
On notera donc aussi $3 \notin$ `pairs` ou encore $3 \in \{x|(x \in \mathbb{N}) \wedge \neg P(x)\}$ ou encore $3 \notin \{t|(t \in \mathbb{N}) \wedge P(t)\}$ (est-ce équivalent :-) ?)ou encore $\neg P(3)$.

Considérons une autre propriété R : un nombre x vérifie R si, et seulement si, son carré vaut 2.
Quelque soit l'entier x, on a $x^2 \neq 2$. On notera cette propriété par $\forall x(x \in \mathbb{N} \wedge \neg R(x))$.
$\forall$ est un *quantificateur universel* (c'est un A à l'envers, car en allemand, ce symbole se lit *für Alle* et a été introduit en 1934 par Gerhard Gentzel...).
Considérons une autre propriété S : un nombre x vérifie S si, et seulement si, son carré vaut 4.
Il existe au moins un entier qui vérifie S : il s'agit de 2.
On note alors $\exists x(x \in \mathbb{N} \wedge S(x))$.
$\exists$ est l'autre quantificateur universel introduit par Giuseppe Peano en 1897 (un E à l'envers comme dans *Esiste almeno uno*).
On peut même préciser que cet entier est *unique* en faisant suivre le quantificateur d'un point d'exclamation :
$\exists !x(x \in \mathbb{N} \wedge S(x))$
Avec Haskell :

Haskell

```
s :: (Num a, Eq a) => a -> Bool
s =                   \x -> x^2 == 4
```

donne

Haskell

```
*Main> s 2
True
*Main> s 2.0
True
*Main> s 3
False
*Main> any s [0..]
True
*Main> find s [0..]
Just 2
*Main> filter s [0..100]
[2]
```

Pour aller plus loin

Paradoxe de Russel
Dans une ville, il n'y a qu'un barbier qui rase tous ceux qui ne se rasent pas eux-mêmes : qui rase le barbier ?...
Plus formellement, cette chose :

$$S = \{E | E \text{ est un ensemble} \wedge E \notin E\}$$

ne peut pas être un ensemble (pourquoi ?).
Pour éviter ce genre de problème, nous n'utiliserons nos définitions par compréhension que pour des objets appartenant à des ensembles connus.
C'est un peu l'esprit de la correction de la théorie naïve donnée par FRAENKEL.

Voici par exemple un critère de définition de $\forall$ et $\exists$ qui pourrait nous être utile pour définir notre type « ensemble » sur Haskell :

Théorème 2 - 1

$$\forall x(x \in E \to P(x)) \equiv E = \{x \in E \mid P(x)\}$$

$$\exists x(x \in E \to P(x)) \equiv \{x \in E \mid P(x)\} \neq \emptyset$$

$$\neg(\forall x(x \in E \to P(x))) \equiv \exists x(x \in E \to \neg P(x))$$

$$\neg(\exists x(x \in E \to P(x))) \equiv \forall x(x \in E \to \neg P(x))$$

3 Parties d'un ensemble

3 1 Notion d'égalité

Définition 2 - 1

Axiome d'extensionalité
Deux ensembles A et B sont égaux si, et seulement si, ils contiennent les mêmes éléments.
On écrit alors $A = B$.

Ça paraît idiot mais cela permet de bien cerner le problème : c'est la notion d'appartenance d'un élément à un ensemble qui est primordiale.

Recherche

On note $E = \{x \mid x \in \mathbb{Z} \wedge x^2 = 1\}$ et $F = \{x \mid x \in \mathbb{R} \wedge \lfloor x \rfloor = 1\}$.
Que pouvez-vous dire de E par rapport à F ?

La notion d'égalité est très importante et peut être compliquée sur machine :

Haskell
```
Prelude> [1,2,3] == [1,3,2]
False
Prelude> 3 * 0.1 == 0.3
False
Prelude> 2 * 0.1 == 0.2
True
```

Nous reparlerons de manière approfondie de tout cela dans le chapitre suivant lors de l'étude des relations d'équivalence qui seront nos égalités créées selon des critères que nous choisirons. Par exemple, comment créer un test d'égalité de deux ensembles pour que la machine réponde vrai à `{1,2,3} == {1,3,2}` ?...

3 2 Inclusion

Définition 2 - 2

inclusion
L'ensemble X est inclus dans l'ensemble Y si, et seulement si, tous les éléments de X sont des éléments de Y. Cela se note

$$(X \subseteq Y) \leftrightarrow ((z \in X) \rightarrow (z \in Y))$$

On dira indifféremment « X *est contenu dans* Y », « Y *contient* X », « X *est un sous-ensemble de* Y », « X *est une partie de* Y ».

Voici maintenant un théorème qui a l'air bien bête...mais il faudrait quand même le démontrer (vous vous rendrez compte qu'en informatique, c'est souvent les choses qui paraissent évidentes qu'il est difficile d'implémenter !)

Théorème 2 - 2

Tout ensemble E contient l'ensemble vide.

La démonstration est l'occasion de bien comprendre les bases de la logique des propositions. Que pensez-vous de la proposition suivante :

$$(\emptyset \subseteq E) \leftrightarrow ((z \in \emptyset) \rightarrow (z \in E))$$

Danger

Notations
Il faudra bien distinguer $\subset$, $\subseteq$ et $\not\subseteq$. $B \subsetneq A$ signifie $(B \subseteq A$ et $B \neq A)$.
Il ne faut pas confondre avec $B \not\subseteq A$ qui exprime que B n'est pas inclus dans A. On dit que B est une partie propre de A si, et seulement si, $B \subseteq A$ avec $B \neq A$ et $B \neq \emptyset$ (il est nécessaire que A ne soit pas vide et ne soit pas un singleton).

La notion d'inclusion va nous permettre de réécrire l'axiome d'extensionalité :

Théorème 2 - 3

Axiome d'extensionalité version II

$$(\forall X)(\forall Y)(((X \subseteq Y) \wedge (Y \subseteq X)) \rightarrow (X = Y))$$

Pratiquement, ce sera le moyen le plus utile pour prouver que deux ensembles sont égaux et ce que nous utiliserons pour l'implémentation du module « ensemble » :

Haskell

```
est_inclus :: (Eq a) => Ens a -> Ens a -> Bool
est_inclus ens1 ens2 = pour_tout (contient ens2) ens1

instance (Eq a) => Eq (Ens a) where
  ens1 == ens2 = (est_inclus ens1 ens2) && (est_inclus ens2 ens1)
```

3 3 Ensemble des parties d'un ensemble

Voici un petit puzzle pour introduire notre propos...

Recherche

Montrez que

$$\{a\} = \{b\} \leftrightarrow a = b$$

puis que $\emptyset \neq \{\emptyset\}$ et $\{\emptyset\} \neq \{\{\emptyset\}\}$.

Soit maintenant E un ensemble, $\left\{ X \mid X \subseteq E \right\}$ est alors lui-même un ensemble. C'est l'ensemble des solutions de $X \subseteq E$ d'inconnue X.

Recherche

Cet ensemble admet toujours au moins deux éléments : lesquels ?

On appelle cet ensemble l'*ensemble des parties de E*.

Théorème 2 - 4

Nous admettrons que si E désigne un ensemble alors **l'ensemble des parties de E** est un ensemble noté $\mathcal{P}(E)$

$$\mathcal{P}(E) = \left\{ X \mid X \subseteq E \right\}$$

Prenons par exemple pour ensemble $E = \left\{ D, L, M \right\}$ alors :

$$\mathcal{P}(E) = \left\{ \emptyset, \{D\}, \{L\}, \{M\}, \{D,L\}, \{D,M\}\{L,M\}, E \right\}$$

Haskell

```
*Main> let e1 = lit ['D','L','M']
*Main> ens_parties e1
{ { } { 'M' } { 'L' } { 'L' 'M' } { 'D' } { 'D' 'M' } { 'D' 'L' } { 'D' 'L' 'M' } }
```

Que dire alors de $\mathcal{P}(\mathcal{P}(E))$? Que c'est un ensemble « compliqué », qu'il a 256 éléments, comme nous le prouverons bientôt, dont en voici quelques uns :

$$\mathcal{P}(\mathcal{P}(E)) = \left\{ \emptyset, \{\{D\}\}, \cdots, \{\{D\}, \{D,L\}\}, \cdots, \{\{D,L\}, \{D,M\}\}, \cdots \right\}$$

On comprend tout de suite la difficulté d'imaginer l'ensemble des parties d'un ensemble...

3 4 Construction récursive

La remarque qui va suivre va nous permettre à la fois de compter les éléments de $\mathcal{P}(E)$ en fonction du nombre d'éléments de E et également de construire informatiquement l'ensemble $\mathcal{P}(E)$.

Il suffit de remarquer que si $E = \emptyset$, alors $\mathcal{P}(E) = \left\{ \emptyset \right\}$.

Sinon, E contient au moins un élément x, les autres éléments de E formant un ensemble F avec $F = \left\{ a \mid a \in E \wedge a \neq x \right\}$.

Le monde des parties de E se partage alors en deux catégories : les parties qui contiennent x et les autres...

$\mathcal{P}(E)$ est alors l'union de $\mathcal{P}(F)$ et des éléments de $\mathcal{P}(F)$ auxquels on a ajouté x.

Par exemple :

Haskell

```
ens_parties ::  Eq a => Ens a -> Ens (Ens a)
ens_parties Vide = Ens (Vide,Vide)
ens_parties (Ens (t,q)) = union pq (applique (insere t) pq)
  where pq = ens_parties q
```

4 Algèbre des ensembles

Nous définirons bientôt ce qu'est une algèbre, ce qu'est une loi de composition. Contentons-nous pour l'instant de définitions basiques à partir de ce que nous avons déjà défini.

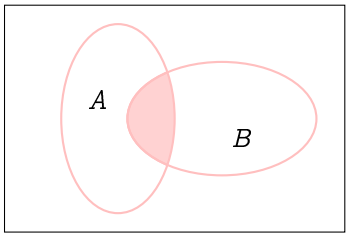
4 1 Intersection

Définition 2 - 3

L'**intersection** des ensembles A et B est l'ensemble $A \cap B$ constitué des éléments communs à A et B.

$$A \cap B = \left\{ x \mid (x \in A) \wedge (x \in B) \right\}$$

C'est une partie de E. De plus $(x \in A \cap B) \leftrightarrow (x \in A \wedge x \in B)$.



$A \cap B$

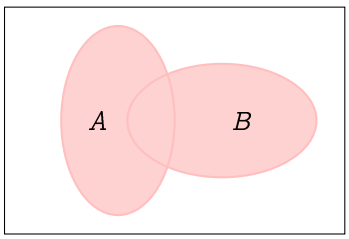
Recherche Démontrez que $(A \cap B = B) \leftrightarrow (B \subseteq A)$.

4 2 Union

Définition 2 - 4 L'**union** ou la **réunion** des ensembles A et B est l'ensemble :

$$A \cup B = \left\{ x \mid (x \in A) \vee (x \in B) \right\}$$

C'est une partie de E. De plus $(x \in A \cup B) \leftrightarrow (x \in A \vee x \in B)$.
On doit se persuader que le « ou » intervenant dans cette définition est un « ou non exclusif » et par conséquent on a $A \subseteq (A \cup B)$, $B \subseteq (A \cup B)$ et $(A \cap B) \subseteq (A \cup B)$.



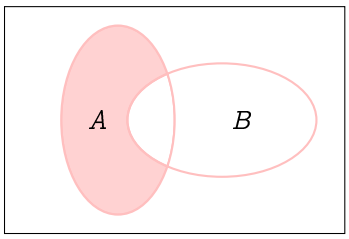
$A \cup B$

Recherche Démontrez que $(A \cup B = A) \leftrightarrow (B \subseteq A)$.

4 3 Différence

Définition 2 - 5 La **différence** des ensembles A et B est l'ensemble

$$A \setminus B = \left\{ x \mid (x \in A) \wedge (x \notin B) \right\}$$



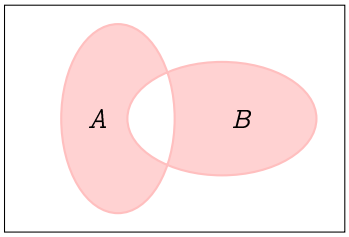
$A \setminus B$

Remarque Certains préfèrent utiliser la notation $A - B$ à la place de $A \setminus B$.

4 4 Différence symétrique

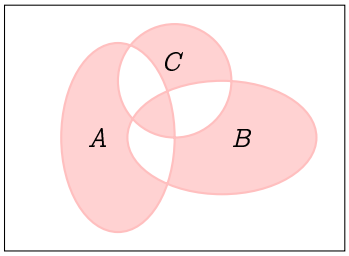
Définition 2 - 6 La **différence symétrique** des ensembles A et B est l'ensemble :

$$\begin{aligned} A \Delta B &= \left\{ x \mid (x \in A \setminus B) \vee (x \in B \setminus A) \right\} = (A \setminus B) \cup (B \setminus A) \\ A \Delta B &= \left\{ x \mid (x \in A \cup B) \wedge (x \notin A \cap B) \right\} = (A \cup B) \setminus (A \cap B) \end{aligned}$$



$A \Delta B$

Vous pourrez vérifier que $A \Delta B \Delta C$ peut être représenté par :



$A \Delta B \Delta C$

À retenir

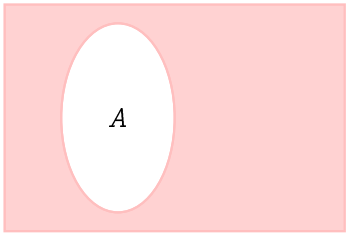
La différence symétrique correspond au « ou exclusif » : fromage ou dessert...

4 5 Complémentaire

Définition 2 - 7

A désignant une partie de E, le **complémentaire** de A par rapport à E ou le complémentaire de A dans E est l'ensemble noté $\complement_E A$ défini par

$$\complement_E A = E \setminus A = \left\{ x \mid (x \in E) \wedge (x \notin A) \right\}$$



$\complement_E A$

S'il n'y a pas d'ambiguïté sur le référentiel E, $\complement_E A$ est noté $\overline{A}$ et $\overline{\overline{A}} = A$.

Remarque

$A \setminus B = A \cap \overline{B}$ et $A \Delta B = \left(A \cap \overline{B}\right) \cup \left(\overline{A} \cap B\right)$.

4 6 Lois de De Morgan

Théorème 2 - 5

Lois de De Morgan

$$\overline{\left(\bigcup_{i=1}^{n} A_i\right)} = \bigcap_{i=1}^{n} \overline{A_i} \qquad \text{et} \qquad \overline{\left(\bigcap_{i=1}^{n} A_i\right)} = \bigcup_{i=1}^{n} \overline{A_i}$$

Ça se démontre...
Ceci s'exprime en français courant de la façon suivante :

— Le complémentaire d'une union est égale à l'intersection des complémentaires.

— Le complémentaire d'une intersection est égale à la réunion des complémentaires.

4 7 Dualité

Regroupons les lois algébriques vues précédemment dans un tableau. On considère des parties A, B et C d'un ensemble E.

Idempotence	$A \cup A = A$	$A \cap A = A$
Associativité	$(A \cup B) \cup C = A \cup (B \cup C)$	$(A \cap B) \cap C = A \cup (B \cap C)$
Commutativité	$A \cup B = B \cup A$	$A \cap B = B \cap A$
Distributivité	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Identité	$A \cup \emptyset = A$	$A \cap E = A$
Involution	$\overline{\overline{A}} = A$	
Complémentaire	$A \cup \overline{A} = E$ $\overline{E} = \emptyset$	$A \cap \overline{A} = \emptyset$ $\overline{\emptyset} = E$
De Morgan	$\overline{A \cup B} = \overline{A} \cap \overline{B}$	$\overline{A \cap B} = \overline{A} \cup \overline{B}$

Soit e une équation algébrique définie sur $\mathcal{P}(E)$. La duale $e^\star$ de e est l'équation obtenue en remplaçant $\cap$, $\cup$, E et $\emptyset$ respectivement par $\cup$, $\cap$, $\emptyset$ et E.
Le *principe de dualité* affirme que si e est vérifiée pour tout élément de $\mathcal{P}(E)$, alors sa duale $e^\star$ aussi : ça se démontre...

5 Partition d'un ensemble

Définition 2 - 8

E désignant ici un ensemble non vide. Soit P, une partie non vide de $\mathcal{P}(E)$ (on a donc $P \subseteq \mathcal{P}(E)$ et $P \neq \emptyset$) P est un ensemble non vide de parties de E. On dit que P est une **partition** de E si, et seulement si,

1. tout élément de P est non vide,
2. deux éléments distincts quelconques de P sont disjoints,
3. tout élément de E appartient à l'un des éléments de P.

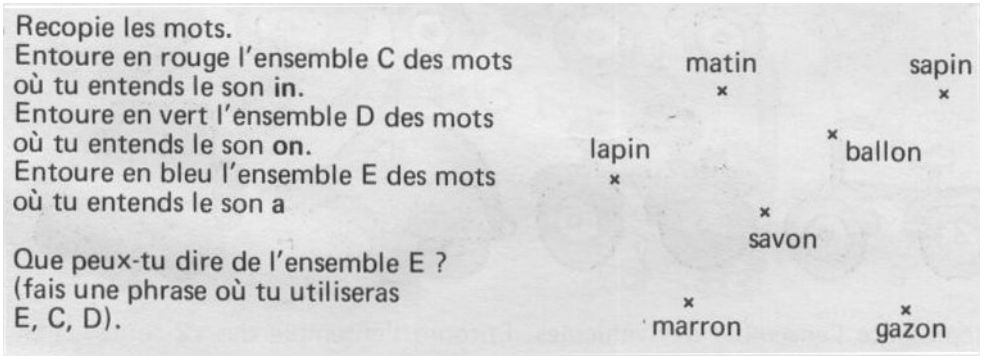
Si E est fini, toute partition P de E est du type $P = \{A_1, A_2, \ldots, A_k\}$ avec

$$A_i \subseteq E, A_i \neq \emptyset, A_i \cap A_j = \emptyset \text{ si } i \neq j \text{ et } \bigcup_{i=1}^{k} A_i = E$$

On peut imaginer une partition de E comme un découpage de E, aucun des morceaux étant non vide, les morceaux étant tous disjoints deux à deux. Par exemple

$$\{\{\beta\}, \{\alpha, \delta\} \{\varepsilon, \gamma\}\}$$

est une partition de l'ensemble $\{\alpha, \beta, \gamma, \delta, \varepsilon\}$.
Pour vous distraire, voici un exercice extrait de mon livre de CE1 :



Recherche

Ce qui va nous intéresser informatiquement, c'est une fonction qui crée une partition d'un ensemble selon une propriété, par exemple pour regrouper les entiers pairs parmi les entiers de 0 à 10 dans un ensemble et les autres entiers dans un deuxième ensemble.
À vous de l'imaginer, récursivement bien sûr !
Il faut savoir que cette fonction existe déjà en Haskell :

Haskell
```
Prelude> Data.List.partition (\x -> x < 0) [-1,5,-65,8,5,-12]
([-1,-65,-12],[5,8,5])
```

Mais bon, ce qui est rigolo, c'est de créer nos fonctions...

6 Produit cartésien

6 1 À table

Cette partie va nous permettre d'introduire des notions informatiques fondamentales, notamment dans le traitement des bases de données.
Vous êtes embauché(e) dans un restaurant. Vous disposez de trois ensembles :

— l'ensemble des entrées :

$$E = \{\text{Cuisses de sauterelles panées, œuf mou, huîtres de l'Erdre}\}$$

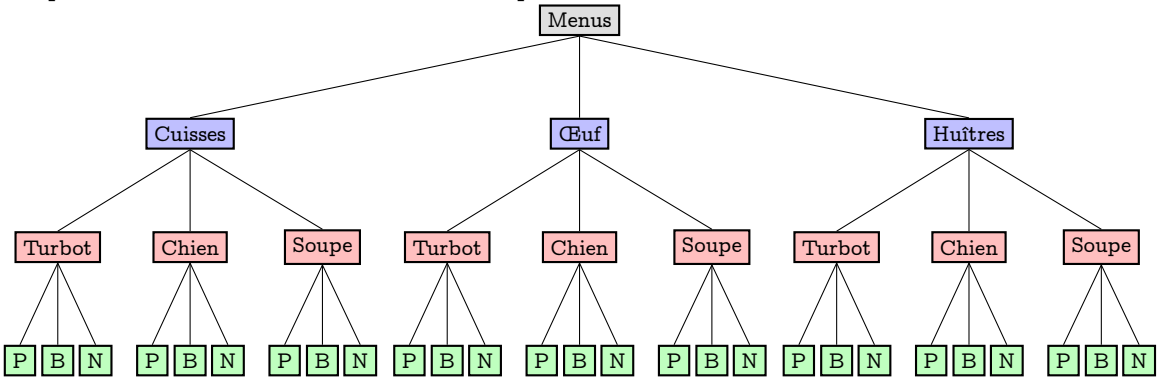
— l'ensemble des plats de résistance :

$$P = \{\text{Turbot à l'huile de ricin, Chien à l'andalouse, Soupe d'orties}\}$$

— l'ensemble des desserts :

$$D = \{\text{Pomme, Banane, Noix}\}$$

Vous avez envie de créer un nouvel ensemble, celui des menus possibles. Une première idée consiste à regrouper tout le monde en prenant $E \cup P \cup D$.
Je sais bien que la gastronomie n'est pas la principale préoccupation des jeunes au palet perverti par les tristes fastefoudes mais en général on commande UNE entrée SUIVIE D'UN plat de résistance SUIVI D'UN dessert or la simple union que vous avez proposée peut vous faire choisir un menu totalement différent : cinq desserts et vingt entrées, dans n'importe quel ordre, par exemple.
Nous avons besoin de créer un « objet » ordonné de trois composantes, chacune étant choisie respectivement dans E, P et D.
Par exemple, (œuf mou, chien à l'andalouse, pomme) est un menu. C'est un triplet, à ne pas confondre avec l'ensemble {œuf mou, chien à l'andalouse, pomme } qui n'est pas ordonné.
On peut avoir une vision de tous les menus possibles avec un arbre :



Avec Haskell :

Haskell
```
e = ["Cuisses","Oeuf","Huitres"]
```

```
p = ["Turbot","Chien","Soupe"]
d = ["Pomme","Banane","Noix"]

menus = [(x,y,z) | x <- e, y <- p, z <- d]
```

Les menus :

Haskell

```
*Main> menus
[("Cuisses","Turbot","Pomme"),("Cuisses","Turbot","Banane"),("Cuisses","Turbot","Noix"),("Cuiss
```

Le type de la liste des menus :

Haskell

```
*Main> :t menus
menus :: [([Char], [Char], [Char])]
```

6 2 Paire ordonnée - Produit de deux ensembles

Comment créer de l'ordre à partir du désordre...

Définition 2 - 9

Paire ordonnée

On note $\langle a,b\rangle = \{\{a\},\{a,b\}\}$

On remarque tout de suite que les rôles de a et b ne sont pas symétriques.
L'égalité des ensembles va nous permettre de définir une égalité des paires : ce n'est pas la même ! Un(e) informathicien(ne) doit bien s'en rendre compte...
Démontrez tout d'abord que :

Théorème 2 - 6

$$(\{a,b\} =_{\text{ens}} \{x,y\}) \equiv ((a=x \wedge b=y) \vee (a=y \wedge b=x))$$

puis que

Théorème 2 - 7

$$(\langle a,b\rangle =_{\text{paire}} \langle x,y\rangle) \equiv (a=x \wedge b=y)$$

On peut alors définir le produit (cartésien) de deux ensembles comme étant l'ensemble des paires formées d'éléments de ces ensembles...

Définition 2 - 10

$$A \otimes B = \{(a,b) \mid (a \in A) \wedge (b \in B)\}$$

Pour se représenter le produit cartésien $E \otimes F$ avec $E = \{a,b,c\}$ et $F = \{1,2,3,4,5\}$, on peut évidemment l'écrire en extension : il possède $3 \times 5 = 15$ éléments (voir à ce sujet le paragraphe suivant).

$$E \otimes F = \{(a,1),(a,2),\cdots,(c,5)\}$$

mais il est souvent préférable de faire appel à un tableau du type suivant :

$\otimes$	**1**	**2**	**3**	**4**	**5**
a	$(a,1)$	$(a,2)$	$(a,3)$	$(a,4)$	$(a,5)$
b	$(b,2)$	$(b,2)$	$(b,3)$	$(b,4)$	$(b,5)$
c	$(c,1)$	$(c,2)$	$(c,3)$	$(c,4)$	$(c,5)$

Avec Haskell :

Haskell

```
Prelude> [(x,y) | x <- ['a','b','c'], y <- [1,2,3,4,5]]
[('a',1),('a',2),('a',3),('a',4),('a',5),('b',1),('b',2),('b',3),('b',4),('b',5),('c',1),
('c',2),('c',3),('c',4),('c',5)]
Prelude> [(x,y) | x <- ['a','b','c'], y <- [1,2,3,4,5], mod y 2 == 0]
[('a',2),('a',4),('b',2),('b',4),('c',2),('c',4)]
```

Mais comment faire avec notre menu qui a trois plats ordonnés ?

6 3 n-uplets - Produit d'un nombre quelconque d'ensembles

On peut dire qu'un menu est un couple formé d'une entrée et d'un couple plat de résistance-dessert...
Définissons temporairement un triplet ordonné $\langle a, b, c\rangle$ par : $\langle a, b, c\rangle = \langle a, \langle b, c\rangle\rangle$.

Recherche

Est-ce que c'est licite ? C'est-à-dire est-ce que c'est une définition...qui définit vraiment un triplet ordonné ? Qu'est-ce que ça veut dire en fait « définit vraiment un triplet ordonné » ?
Comment va-t-on faire alors pour un quadruplet ? Un n-uplet ?
Qu'est-ce que c'est que ça : $E_1 \otimes E_2 \otimes \cdots \otimes E_n = \prod_{i=1}^{n} E_i$?
Je vous laisse un peu de place pour résumer les résultats...

Deux fonctions utiles :

Définition 2 - 11

Projections
On note π_1 (π_2) la fonction définie sur un produit $E \otimes F$ qui a un couple associe sa première (deuxième) composante

Sur Haskell, c'est **`fst`** et **`snd`** :

Haskell

```
Prelude> fst ('a','b')
'a'
Prelude> snd ('a','b')
'b'
```

Aparté

Contrairement à votre chambre, sur machine, il est beaucoup plus compliqué de créer du désordre que de l'ordre.
Nous suivrons donc le cheminement inverse de ce qui vient d'être fait : nous partirons de n-uplets ordonnés et nous créerons des ensembles non ordonnés...

7 Notion de cardinal

Le cardinal d'un ensemble, c'est en gros le nombre de ses éléments. La notion de cardinal est intimement liée à la notion de fonction et à l'ensemble $\mathbb{N}$ des entiers naturels que nous détaillerons plus tard. Nous nous contenterons dans un premier temps d'une approche intuitive. Le cardinal de E est indifféremment noté :$\mathrm{Card}(E)$ ou $|E|$ ou $\#E$

Par exemple $|\emptyset| = 0$, cette notation ne devra bien sûr pas être confondue avec la valeur absolue d'un réel ou bien le module d'un nombre complexe. Quoi qu'il en soit, le contexte permettra généralement de lever l'ambiguïté.
Nous admettons les résulats suivants qui ne concernent que des ensembles **finis** : il faut en effet savoir que les ensembles infinis ont aussi un cardinal, ce qu'a introduit CANTOR à la fin du XIX^{e} siècle...Cela l'a d'ailleurs rendu fou....

Propriétés 2 - 1

— Si $A \subseteq B$ alors $|A| \leqslant |B|$. Une conséquence intéressante est celle ci : si $A \subseteq B$ avec $|A| = |B|$ alors $A = B$.
— $|A - B| \leqslant |A|$.
— $|A \cup B| = |A| + |B| - |A \cap B|$
— **Si on a** $E_i \cap E_j = \emptyset$ **lorsque** $i \neq j$, alors $\left|\bigcup_{i=1}^{n} E_i\right| = \sum_{i=1}^{n} |E_i|$.
— $|E_1 \otimes E_2| = |E_1|.|E_2|$ et

$$\begin{aligned} |E_1 \otimes E_2 \otimes \cdots \otimes E_n| &= |E_1|.|E_2|.\cdots.|E_n| \\ |E^n| &= (|E|)^n \end{aligned}$$

— La formule suivante est absolument à connaître :

$$|\mathcal{P}(E)| = 2^{|E|}$$

Nous la démontrerons en exercice.

Une petite question d'anglais de spécialité... : nos amis anglo-saxons appellent $\mathcal{P}(E)$ *power set of* E. Pourquoi ?...

Recherche

Comment calculer récursivement le cardinal d'un ensemble ?

8 Fonction caractéristique (niveau II...)

Nous avons déjà indiqué quelques propriétés des opérations que nous venons de présenter, le plus souvent sans démonstration car ces dernières sont assez fastidieuses lorsque, pour démontrer l'égalité de deux ensembles, il faut envisager plein de cas. Les diagrammes de Venn peuvent nous aiguiller vers une démonstration mais ne peuvent en aucun cas la remplacer. Nous allons, ici, présenter un outil très efficace qui permet de faire beaucoup de démonstrations sans trop de peine.
Soit E un ensemble. Pour toute partie A de E on définit la fonction caractéristique $\mathbb{1}_A$ de A dans E par :

$$\mathbb{1}_A : \begin{array}{ccc} E & \to & \{0;1\} \\ x & \mapsto & \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si } x \notin A \end{cases} \end{array}$$

Une propriété essentielle de cette notion est donnée par l'équivalence $\mathbb{1}_A = \mathbb{1}_B \iff A = B$
Vous ne voyez peut-être pas trop où se trouve la simplification ! Un informaticien utilise un « *computer* », c'est-à-dire un calculateur. Nous voudrons donc pouvoir *calculer* avec des ensembles.

Notations

Pour simplifier les écritures, nous utiliserons des opérations sur ces applications caractéristiques :

- $\mathbb{1}_A.\mathbb{1}_B$ ou $\mathbb{1}_A\mathbb{1}_B$ définie pour tout x de E par $(\mathbb{1}_A\mathbb{1}_B)(x) = \mathbb{1}_A(x) \times \mathbb{1}_B(x)$;
- $1 - \mathbb{1}_A$ définie par $(1 - \mathbb{1}_A)(x) = 1 - \mathbb{1}_A(x)$;
- $\mathbb{1}_A + \mathbb{1}_B$ définie par $(\mathbb{1}_A + \mathbb{1}_B)(x) = \mathbb{1}_A(x) + \mathbb{1}_B(x)$.

On voudrait maintenant exprimer $\mathbb{1}_{\overline{A}}$, $\mathbb{1}_{A\cap B}$, $\mathbb{1}_{A-B}$, $\mathbb{1}_{A\cup B}$ et $\mathbb{1}_{A\triangle B}$ à l'aide de $\mathbb{1}_A$ et de $\mathbb{1}_B$.

- $\mathbb{1}_{\overline{A}} = 1 - \mathbb{1}_A$, c'est évident, ces deux applications coïncident sur E.
- $\mathbb{1}_{A\cap B} = \mathbb{1}_A\mathbb{1}_B$, c'est tout aussi évident, elles coïncident sur E ; on en déduit que $\mathbb{1}_{A\cap A} = \mathbb{1}_A\mathbb{1}_A = \mathbb{1}_A$.
- $\mathbb{1}_{A-B} = \mathbb{1}_A - \mathbb{1}_A\mathbb{1}_B$. On peut le démontrer en vérifiant la coïncidence de ces deux applications sur E mais il est plus judicieux de remarquer que $A - B = A \cap \overline{B}$ et du coup

$$\mathbb{1}_{A-B} = \mathbb{1}_{A\cap\overline{B}} = \mathbb{1}_A\mathbb{1}_{\overline{B}} = \mathbb{1}_A(1 - \mathbb{1}_B) = \mathbb{1}_A - \mathbb{1}_A\mathbb{1}_B$$

- Concernant $\mathbb{1}_{A\cup B}$, pour trouver une formule similaire aux précédentes, il faut faire preuve d'un peu d'imagination : nous utilisons le résultat $\overline{X \cup Y} = \overline{X} \cap \overline{Y}$ que nous démontrons à l'aide du tableau suivant

$x \in X$	$x \in Y$	$x \in \overline{X \cup Y}$	$x \in \overline{X} \cap \overline{Y}$
non	non	oui	oui
non	oui	non	non
oui	non	non	non
oui	oui	non	non

  Ceci étant, $A \cup B = \overline{\overline{A \cup B}} = \overline{\overline{A} \cap \overline{B}}$; on obtient alors

$$\mathbb{1}_{A\cup B} = \mathbb{1}_{\overline{\overline{A}\cap\overline{B}}} = 1 - \mathbb{1}_{\overline{A}\cap\overline{B}}$$

  soit $\mathbb{1}_{A\cup B} = 1 - (1 - \mathbb{1}_A)(1 - \mathbb{1}_B)$ qui se développe en

$$\mathbb{1}_{A\cup B} = \mathbb{1}_A + \mathbb{1}_B - \mathbb{1}_A\mathbb{1}_B$$

- Pour $\mathbb{1}_{A\triangle B}$ je vous laisse prouver que : $\mathbb{1}_{A\triangle B} = \mathbb{1}_A + \mathbb{1}_B - 2\mathbb{1}_A\mathbb{1}_B$.

Recherche

Voici quelques propriétés (en vrac) importantes. Nous ne donnons que quelques démonstrations : le lecteur étant invité à les reprendre et à en faire d'autres...

- $\cup$ et $\cap$ sont associatives et commutatives. Démontrons l'associativité de l'intersection. Pour cela choisissons trois parties quelconques A, B et C de $\mathcal{P}(E)$. Il nous faut démontrer l'égalité des

ensembles $U = A \cap (B \cap C)$ et $V = (A \cap B) \cap C$ et pour cela il suffit de prouver l'égalité des applications caractéristiques de U et V dans E. Allons-y : $\mathbb{1}_U = \mathbb{1}_{A\cap(B\cap C)} = \mathbb{1}_A\mathbb{1}_{B\cap C} = \mathbb{1}_A\mathbb{1}_B\mathbb{1}_C = \mathbb{1}_{A\cap B}\mathbb{1}_C = \mathbb{1}_V$. L'associativité de $\cap$ et $\cup$ permet d'utiliser les notations, lorsque $k \in \mathbb{N}^*$,

$$\bigcup_{i=1}^{k} A_i = A_1 \cup A_2 \cup \ldots \cup A_n$$
$$\bigcap_{i=1}^{k} A_i = A_1 \cap A_2 \cap \ldots \cap A_n$$

— $A \subseteq B \iff A \cup B = B \iff A \cap B = A \iff A - B = \{\}$
— $A \subseteq (A \cup B), (A \cap B) \subseteq A$
— $A - A = A \bigtriangleup A = A \cap \{\} = \{\}$
— $A \cap A = A \cup A = A \cup \{\} = A \cap E = A \bigtriangleup \{\} = A$
— $A \cap (A \cup B) = A = A \cup (A \cap B)$
— $\overline{(\overline{A})} = A, \overline{A} \cap A = \{\}, \overline{A} \cup A = E, \complement_E\{\} = \overline{\{\}} = E, \complement_E E = \overline{E} = \{\}$
— $\cap$ et $\cup$ sont distributives l'une par rapport à l'autre :

$$\begin{aligned} A \cap (B \cup C) &= (A \cap B) \cup (A \cap C) \\ A \cup (B \cap C) &= (A \cup B) \cap (A \cup C) \end{aligned}$$

La démonstration se fait, soit en vérifiant la double inclusion (c'est un bon exercice), soit en utilisant les applications caractéristiques.

— La différence symétrique est associative et commutative. Démontrons l'associativité toujours à l'aide des applications caractéristiques :

$$\mathbb{1}_{A\bigtriangleup(B\bigtriangleup C)} = \mathbb{1}_A + \mathbb{1}_{B\bigtriangleup C} - 2\mathbb{1}_A\mathbb{1}_{B\bigtriangleup C} \text{ avec } \mathbb{1}_{B\bigtriangleup C} = \mathbb{1}_B + \mathbb{1}_C - 2\mathbb{1}_B\mathbb{1}_C$$

donne

$$\mathbb{1}_{A\bigtriangleup(B\bigtriangleup C)} = \mathbb{1}_A + \mathbb{1}_B + \mathbb{1}_C - 2\mathbb{1}_A\mathbb{1}_B - 2\mathbb{1}_A\mathbb{1}_C - 2\mathbb{1}_B\mathbb{1}_C + 4\mathbb{1}_A\mathbb{1}_B\mathbb{1}_C$$

On vérifie que $\mathbb{1}_{(A\bigtriangleup B)\bigtriangleup C}$ s'exprime de la même façon.

— Rappelons les lois de De Morgan

$$\begin{cases} \overline{\left(\bigcup_{i=1}^{n} A_i\right)} = \bigcap_{i=1}^{n} \overline{A_i} \\ \overline{\left(\bigcap_{i=1}^{n} A_i\right)} = \bigcup_{i=1}^{n} \overline{A_i} \end{cases}$$

Nous avons déjà démontré $\overline{A_1 \cup A_2} = \overline{A_1} \cap \overline{A_2}$, pour démontrer

$$\overline{\left(\bigcup_{i=1}^{n} A_i\right)} = \bigcap_{i=1}^{n} \overline{A_i}$$

une récurrence immédiate suffit :

$$\overline{\left(\bigcup_{i=1}^{n+1} A_i\right)} = \overline{\left(\left(\bigcup_{i=1}^{n} A_i\right) \cup A_{n+1}\right)} = \overline{\left(\bigcup_{i=1}^{n} A_i\right)} \cap \overline{A}_{n+1} = \left(\bigcap_{i=1}^{n} \overline{A_i}\right) \cap \overline{A}_{n+1}$$

et l'associativité de l'intersection permet de conclure. Dans la formule

$$\overline{\left(\bigcup_{i=1}^{n} A_i\right)} = \bigcap_{i=1}^{n} \overline{A_i}$$

remplaçons chaque A_i par $\overline{A_i}$:

$$\overline{\left(\bigcup_{i=1}^{n} \overline{A_i}\right)} = \bigcap_{i=1}^{n} \overline{\overline{A_i}} = \bigcap A_i$$

et en écrivant l'égalité des complémentaires de chaque membre il vient

$$\overline{\left(\bigcap_{i=1}^{n} A_i\right)} = \bigcup_{i=1}^{n} \overline{A_i}$$

Remarque

L'intérêt des fonctions caractéristiques est de pouvoir effectuer des calculs sur des ensembles comme avec les nombres ce qui facilite la programmation des opérations ensemblistes..

Haskell

```haskell
data Ens a = Ens (a -> Bool)

contient :: Ens a -> a -> Bool
contient (Ens f) e = f e

insere :: (Eq a) => a -> Ens a -> Ens a
insere e ens = Ens (\x -> (x == e) || (contient ens x))

union :: (Eq a) => Ens a -> Ens a -> Ens a
union ens1 ens2 = Ens (\x -> (contient ens1 x) || (contient ens2 x))

inter :: (Eq a) => Ens a -> Ens a -> Ens a
inter ens1 ens2 = Ens (\x -> (contient ens1 x) && (contient ens2 x))

filtre ::(Eq a) => (a -> Bool) -> (Ens a) -> (Ens a)
filtre pred ens = Ens (\x -> (contient ens x) && (pred x))
```

EXERCICES

Exercice 2 - 1

L'ensemble universel est $S = \{1, 2, 3, 4, 5\}$. Pour chacune des propositions suivantes, donnez-en la négation puis la valeur de vérité.

1. $(\exists x)(x + 3 = 10)$
2. $(\forall x)(x + 3 < 10)$
3. $(\exists x)(x + 3 < 5)$
4. $(\forall x)(x + 3 \leqslant 7)$

Exercice 2 - 2

L'ensemble universel est $S = \{1, 2, 3\}$. Pour chacune des propositions suivantes, donnez-en la négation puis la valeur de vérité.

1. $(\forall y)(\exists x)(x^2 < y + 1)$
2. $(\exists x)(\forall y)(x^2 < y + 1)$
3. $(\forall x)(\exists y)(x^2 < y + 1)$
4. $(\exists y)(\forall x)(x^2 < y + 1)$
5. $(\forall x)(\exists y)(x^2 + y^2 < 12)$
6. $(\forall x)(\forall y)(x^2 + y^2 < 12)$

Exercice 2 - 3

Donnez la négation des propositions suivantes :

1. Tous les étudiants mangent du foin.
2. Il y a un garçon qui joue à la Barbie.
3. Il existe un seul ordinateur dans cette salle.
4. Si j'avais su, il y aurait eu une poche à la culotte du zouave.
5. Tous les étudiants travaillent et demain je serai riche.
6. Il y a des étudiants qui ne travaillent pas ou le Batoustan va quitter le Gourbi Occidental.

Exercice 2 - 4

Parmi les ensembles suivants, quels sont ceux qui sont égaux ?

1. $A = \{x \mid x \in \mathbb{R} \text{ et } x^2 - 4x + 3 = 0\}$
2. $B = \{x \mid x \in \mathbb{R} \text{ et } x^2 - 3x + 2 = 0\}$
3. $C = \{x \mid x \in \mathbb{N} \text{ et } x < 3\}$
4. $D = \{x \mid x \in \mathbb{N} \text{ et } x < 5 \text{ et } x \text{ est impair }\}$
5. $E = \{1, 2\}$
6. $F = \{1, 2, 1\}$
7. $G = \{3, 1\}$
8. $H = \{1, 1, 3\}$

Exercice 2 - 5

$E = \{0, 1, 2, 3, 4, 5, 6\}$. Définir en extension les ensembles suivants :

1. $A_1 = \{x \in \mathbb{N} \mid x^2 \in E\}$
2. $A_2 = \{x \in \mathbb{R} \mid x^2 \in E\}$
3. $A_3 = \{x \in E \mid x^2 \in E\}$
4. $A_4 = \{x \in E \mid \sqrt{x} \in E\}$
5. $A_5 = \{x \in E \mid 2x \in E\}$
6. $A_6 = \{x \in E \mid \frac{x}{2} \in E\}$

Exercice 2 - 6

$E = \{0, 1, 2, 3, 4\}$. Compléter, lorsque c'est possible, par un des symboles (il peut y avoir plusieurs solutions, mais on s'obligera à choisir celle qui donne le plus « de renseignements ») :

$$\in, \ni, \subseteq, \supseteq, =, \neq, \subsetneq, \not\subseteq, \cdots$$

1. $2 \cdots E$,
2. $\{2, 3\} \cdots E$,
3. $\{2\} \cdots E$,
4. $\{2, 3, 4\} \cdots \{4, 3, 2\}$,
5. $\{2, 3, 4\} \cdots \{4, 3, 0\}$,
6. $\{\} \cdots E$,
7. $E \cdots E$,
8. $E \cdots \{E\}$,
9. $\{\} \cdots \{E\}$,
10. $E \cdots \{0, 1, 2, 3, \ldots, 10\}$,
11. $\{0, 1, 2, 3, 4, 5\} \cdots E$

Exercice 2 - 7

La notion de type en informatique est encore un peu floue mais vous en avez déjà une vague idée. Nous allons en parler bientôt.
La commande `:t` sur Haskell nous renseigne sur le type d'objet utilisé :

Haskell

```
Prelude> :t 'a'
'a' :: Char
Prelude> :t ['a','b']
```

```
['a','b'] :: [Char]
Prelude> :t 3 == (2 + 1)
3 == (2 + 1) :: Bool
Prelude> :t ('a',True)
('a',True) :: (Char, Bool)
Prelude> :t pi
pi :: Floating a => a
```

Comment « typeriez-vous » les expressions suivantes :

1. $x \in \{x \mid P(x)\}$
2. $\{x \mid P(x)\}$
3. $(A \cup B) \subseteq C$
4. $(A \cup B) \cap C$
5. $\forall x(x \in A \to x \in B)$
6. $A = B$
7. $x \in A \to x \in B$

Exercice 2 - 8

Soit $A = \{1, 2, ..., 8, 9\}$, $B = \{2, 4, 6, 8\}$, $C = \{1, 3, 5, 7, 9\}$, $D = \{3, 4, 5\}$, $E = \{3, 5\}$ et $\Omega = \{A, B, C, D, E\}$.
Écrivez les ensembles suivants par compréhension puis déterminez les éléments de Ω qui vérifient les conditions suivantes :

1. $X \cup B = \emptyset$;
2. $X \subseteq D \wedge X \not\subseteq B$;
3. $X \subseteq A \wedge X \not\subseteq C$;
4. $X \subseteq C \wedge X \not\subseteq A$.

Exercice 2 - 9

Trouvez des analogies et des différences entre $\subseteq$ définie sur les ensembles et $\leqslant$ définie sur les réels.

Exercice 2 - 10

Simplifiez au maximum les égalités suivantes :

1. $(A \cap B) \cup (A \cap \overline{B})$
2. $(A \cap \overline{B}) \cup (\overline{A} \cap B) \cup (A \cap B)$

Exercice 2 - 11

Démontrez les propriétés de la section 2.4.7 page 31.

Exercice 2 - 12

On définit un opérateur $\uparrow$ sur les ensembles de la manière suivante : $A \uparrow B = \overline{(A \cap B)}$.
Démontrer les égalités suivantes :

1. $A \uparrow A = \overline{A}$;
2. $(A \uparrow A) \uparrow (B \uparrow B) = A \cup B$;
3. $(A \uparrow B) \uparrow (A \uparrow B) = A$??? B.
4. $\uparrow$ est-il associatif ?

Exercice 2 - 13

1. Expliciter
   i. $\mathcal{P}(\{a, \{a\}\})$
   ii. $\mathcal{P}(\{\})$
   iii. $\mathcal{P}(\mathcal{P}(\{\}))$
   iv. $\mathcal{P}(\mathcal{P}(\mathbb{N}_1))$
2. Donner cinq éléments de $\mathcal{P}(\mathcal{P}(\mathcal{P}(\{a, b, c\})))$
3. i. Démontrer par récurrence que si E possède n éléments alors $\mathcal{P}(E)$ possède 2^n éléments.
   ii. Combien d'éléments $\mathcal{P}(\mathcal{P}(\mathcal{P}(\{a, b, c\})))$ contient-il ?
   iii. Combien d'éléments a $\mathcal{P}(\mathcal{P}(\mathcal{P}(\mathcal{P}(\emptyset))))$? Combien y a-t-il d'atomes dans l'Univers ?...

Exercice 2 - 14

1. Soit $A = \{0, 1\}$. Déterminer $\left(A^2 \setminus \{(0, 0)\}\right) \otimes A$.
2. Soit $A = \{3, 5, 7\}$ et $B = \{a, b\}$. Déterminer A^2, B^2, $A \otimes B$, $B \otimes A$, $B^2 \otimes A$.

Exercice 2 - 15

A, B et C sont trois ensembles tous différents de l'ensemble vide. Démontrer que

$$(A \cap B) \otimes C = (A \otimes C) \cap (B \otimes C).$$

Exercice 2 - 16

1. On pose $A = \{a, b\}$ et $B = \{b, c\}$.
   i. Comparer $\mathcal{P}(A) \cap \mathcal{P}(B)$ et $\mathcal{P}(A \cap B)$.
   ii. Comparer $\mathcal{P}(A) \cup \mathcal{P}(B)$ et $\mathcal{P}(A \cup B)$.
2. Répondre aux mêmes questions avec A et B des ensembles quelconques.

Exercice 2 - 17

Soient A et B deux ensembles. Démontrez que

$$(A \otimes B = \emptyset) \leftrightarrow (A = \emptyset \vee B = \emptyset)$$

CHAPITRE 3

Relations et fonctions

Hélène aime Chri-Chri et Chri-Chri aime Johanna mais est-ce que Hélène aime Johanna ? Chri-Chri aime Johanna mais est-ce que Johanna aime Chri-Chri en retour ? Est-ce que Johanna s'aime elle-même ? Ces questions fondamentales sont liées à l'étude des propriétés de la relation « ...aime... ».
Quels liens rajouter entre des pages web pour pouvoir aller de n'importe quelle page vers n'importe quelle autre ? C'est un problème de fermeture transitive d'une relation...
Le SQL : de la programmation relationnelle...
Les réseaux sociaux : mettre en relation des sites marchands avec de potentiels acheteurs.
Vous l'avez compris : les relations, ça compte en informatique.
Ça compte aussi beaucoup de définitions : il y en a 42 dans ce chapitre...

Johanna : *Si votre relation est aussi compliquée, pourquoi ne pas laisser tomber ? C'est peu vraisemblable que ça fonctionne un jour.*
Hélène : *Quand on aime quelqu'un, se battre vaut le coup, peu importe les probabilités.*

Hélène et les garçons, épisode 217 (1992)

1 Préliminaires...

Nous aurons besoin à l'occasion de quelques outils de calculs qui seront repris plus en détail dans les mois à venir.

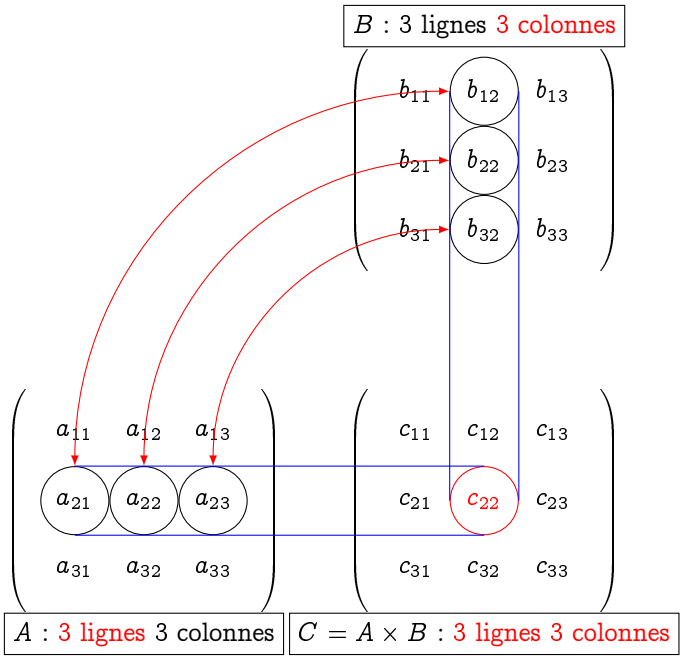
1.1 Un peu de calcul matriciel

Nous aurons besoin de multiplier des matrices entre elles. Voici une disposition pratique pour effectuer les calculs.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

avec n le nombre de colonnes de A qui doit être égal au nombre de lignes de B.

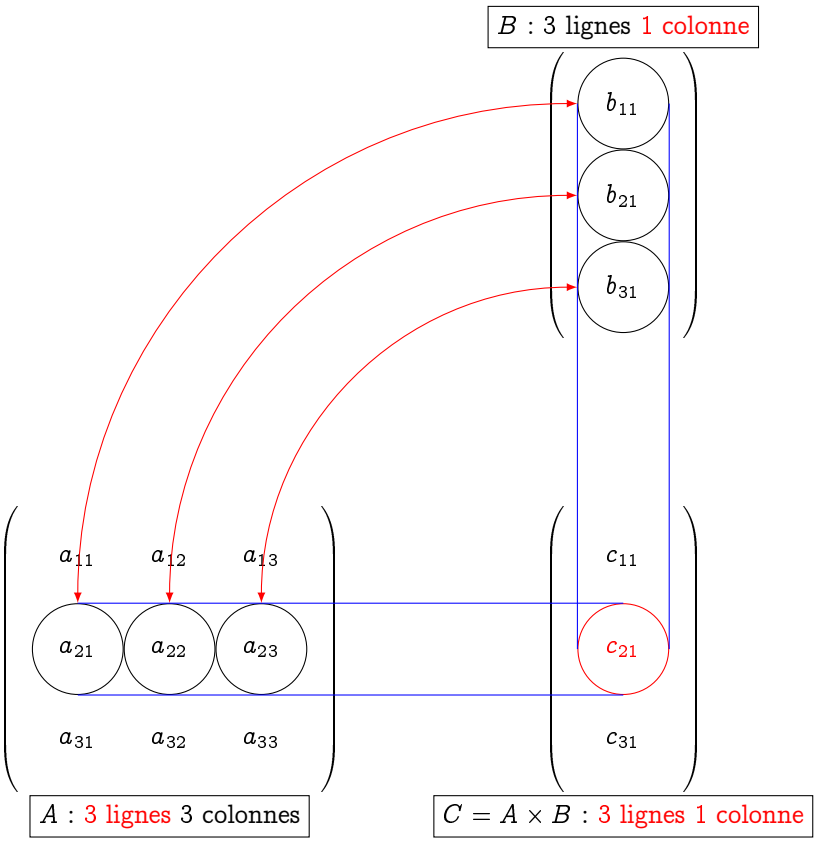
1.1.1 Produit de deux matrices « carré »



Danger

produit de matrice non commutatif
Le produit de deux matrices n'est pas commutatif. Cela signifie qu'en général $A \times B \neq B \times A$.

1 1 2 Produit d'une matrice « carré » et d'une matrice « colonne »



1 2 Calcul booléen

Nous allons définir trois opérations déjà évoquées en logique sur l'ensemble $\{0,1\}$:

$\oplus$	0	1
0	0	1
1	1	0

$\overline{\wedge}$	0	1
0	0	0
1	0	1

$\veebar$	0	1
0	0	1
1	1	1

On pourra combiner le tout et multiplier et additionner des matrices booléennes avec ces opérations.

On utilisera aussi souvent la matrice Attila(n,m) qui ne contient que des 1.

Soit par exemple

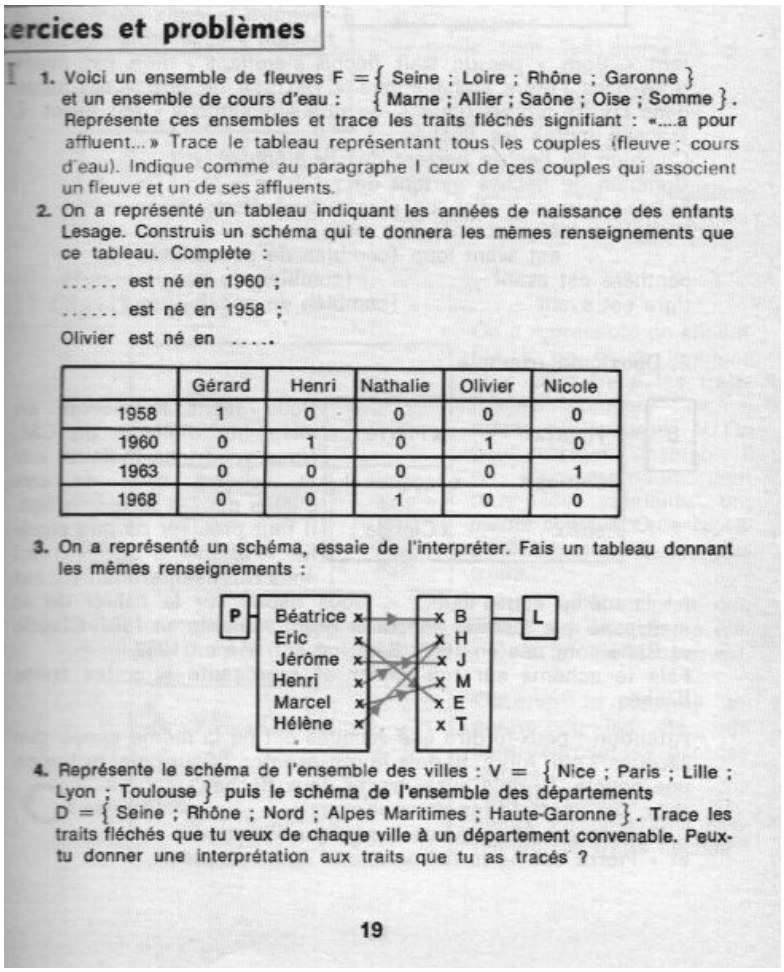
$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Recherche

Calculer $A \oplus B$ (somme booléenne terme à terme), $A \oplus$ Attila$(3,3)$ et $A \overline{\wedge} B$ (produit matriciel en utilisant le produit booléen $\overline{\wedge}$ pour le produit des termes et $\oplus$ pour leur somme).

2 Relations binaires

2 1 Au CM1



ercices et problèmes

1. Voici un ensemble de fleuves F = { Seine ; Loire ; Rhône ; Garonne }
et un ensemble de cours d'eau : { Marne ; Allier ; Saône ; Oise ; Somme }.
Représente ces ensembles et trace les traits fléchés signifiant : «....a pour affluent...» Trace le tableau représentant tous les couples (fleuve ; cours d'eau). Indique comme au paragraphe I ceux de ces couples qui associent un fleuve et un de ses affluents.
2. On a représenté un tableau indiquant les années de naissance des enfants Lesage. Construis un schéma qui te donnera les mêmes renseignements que ce tableau. Complète :
..... est né en 1960 ;
..... est né en 1958 ;
Olivier est né en .....

	Gérard	Henri	Nathalie	Olivier	Nicole
1958	1	0	0	0	0
1960	0	1	0	1	0
1963	0	0	0	0	1
1968	0	0	1	0	0

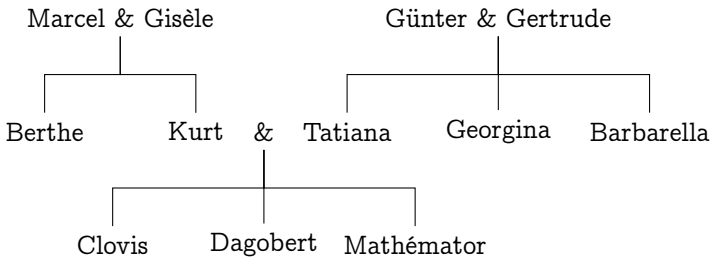
3. On a représenté un schéma, essaie de l'interpréter. Fais un tableau donnant les mêmes renseignements :

4. Représente le schéma de l'ensemble des villes : V = { Nice ; Paris ; Lille ; Lyon ; Toulouse } puis le schéma de l'ensemble des départements D = { Seine ; Rhône ; Nord ; Alpes Maritimes ; Haute-Garonne }. Trace les traits fléchés que tu veux de chaque ville à un département convenable. Peux-tu donner une interprétation aux traits que tu as tracés ?

19

2 2 Généalogie

Voici mon arbre généalogique :



Formez les couples (x, y) tels que x soit le grand-père de y.

2 3 À l'IUT

Une relation a pour but de décrire les « liens » entre les éléments de deux voire plusieurs ensembles. Leur champ d'application est immense.

Définition 3 - 1

Une *relation binaire* entre deux ensemble E et F est la donnée d'un triplet $(E, F, \mathcal{G}_\mathcal{R})$ où $\mathcal{G}_\mathcal{R}$ est un sous-ensemble du produit cartésien $E \times F$. On l'appelle **graphe de la relation**.
E est l'**ensemble de départ** de la relation et F son **ensemble d'arrivée**.
Une *relation n-aire* entre n ensembles E_1, E_2,...,E_n est la donnée d'un $n+1$-uplet constitué de ces ensembles et d'un sous-ensemble du produit cartésien $E_1 \times E_2 \times \cdots \times E_n$.

Dans ce chapitre, nous nous contenterons d'étudier les relations binaires.

Si $(x,y) \in G_{\mathcal{R}}$, on dit que l'élément x de l'ensemble de départ E est en relation par $\mathcal{R}$ avec l'élément y de l'ensemble d'arrivée F. L'écriture $x\mathcal{R}y$ ou $\mathcal{R}(x,y)$ est équivalente à $(x,y) \in G_{\mathcal{R}}$. On dit que y est **UNE image** de x par la relation $\mathcal{R}$ et que x est **UN antécédent** de y. Si $(x,y) \notin G_{\mathcal{R}}$, x n'est pas en relation avec y.

Remarque

Méthode B
Le couple (x,y) de $G_{\mathcal{R}}$ est aussi noté $x \mapsto y$, c'est une notation de la méthode B.

2 4 Retour au CE1

Reprenons la figure de mon vieux livre de CM1 page ci-contre.
La relation $\mathcal{R}$ est définie par « ...a choisi... (la) » avec un ensemble de départ qui est en fait l'ensemble des élèves : $E = \{C, N, M, O, D, H\}$ et un ensemble d'arrivée qui est en fait l'ensemble des jeux : $J = \{l, p, b, f, t\}$
Son graphe est :

$$G_{\mathcal{R}} = \{(C,l),(C,p),(N,l),(M,p),(O,b),(O,t),(D,t),(H,t)\}$$

Les écritures suivantes sont équivalentes :

$$(C,p) \in G_{\mathcal{R}} \leftrightarrow C \ \mathcal{R} \ p$$

Aparté

Posons-nous la question « combien existe-t-il de relations (binaires) de E vers F » ? Nous venons de dire que, les ensembles E et F étant précisés, la relation est déterminée par son graphe qui est une partie de $E \times F$. Il y a donc autant de relations de E vers F que de parties dans $E \times F$, c'est à dire $2^{|E \times F|}$.
Si, par exemple E a 106 éléments et F a 8 éléments, alors il existe 2^{848} relations de E vers F or $2^{848} \approx 2 \times 10^{255}$ et le nombre d'atomes estimé dans tout l'univers est 10^{80} : il y a donc pas mal de relations possibles entre 106 garçons et 8 filles d'INFO 1...

2 5 Domaine, codomaine

Définition 3 - 2

L'ensemble des éléments de E qui ont au moins une image par $\mathcal{R}$ est l'ensemble de définition ou **domaine** de définition de la relation $\mathcal{R}$ que l'on note le plus souvent par $\mathcal{D}_{\mathcal{R}}$ ou $\mathrm{dom}(\mathcal{R})$. Remarquons que l'on a forcément $\mathrm{dom}(\mathcal{R}) \subseteq E$.

$$\mathrm{dom}(\mathcal{R}) = \{x \mid \exists y \, ((x,y) \in \mathcal{G}_{\mathcal{R}})\}$$

Si on est sûr que $\mathrm{dom}(\mathcal{R}) = E$, c'est-à-dire que tout élément de l'ensemble de départ a au moins une image, on dit que la relation $\mathcal{R}$ est **totale**, sinon est est **partielle**.

Définition 3 - 3

L'ensemble des éléments de F qui ont au moins un antécédent dans E est appelé l'image de $\mathcal{R}$ ou l'image de E par $\mathcal{R}$ ou le **codomaine** de $\mathcal{R}$. On utilise indifféremment les notations suivantes pour désigner le codomaine de $\mathcal{R}$noté $\mathrm{Im}\,\mathcal{R}$ ou $\mathrm{Im}(\mathcal{R})$ ou $\mathrm{Ran}(\mathcal{R})$ ou $\mathrm{codom}(\mathcal{R})$

$$\mathrm{Ran}(\mathcal{R}) = \{y \mid \exists x \, ((x,y) \in \mathcal{G}_{\mathcal{R}})\}$$

$\mathrm{Im}(\mathcal{R})$ se lit « im de $\mathcal{R}$ » ou « image de $\mathcal{R}$ », la notation $\mathrm{Ran}(\mathcal{R})$ venant de l'anglais « range » (cible en français). Il faut tout de suite remarquer que $\mathrm{Im}(\mathcal{R})$ n'est autre que l'ensemble de toutes les images des éléments de E par la relation $\mathcal{R}$ et que l'on a forcément $\mathrm{Im}(\mathcal{R}) \subseteq F$.

2 6 Représentation d'une relation

Lorsque E et F sont finis on a forcément $G_{\mathcal{R}} \subseteq E \times F$ qui est fini et on peut représenter la relation $\mathcal{R}$ ou son graphe $G_{\mathcal{R}}$ de différentes manières. En voici quelques exemples.

2.6.1 Représentation linéaire ensembliste

Prenons pour $E = \{a, b, c, d, e\}$, $F = \{\alpha, \beta, \gamma, \delta\}$ et pour graphe :

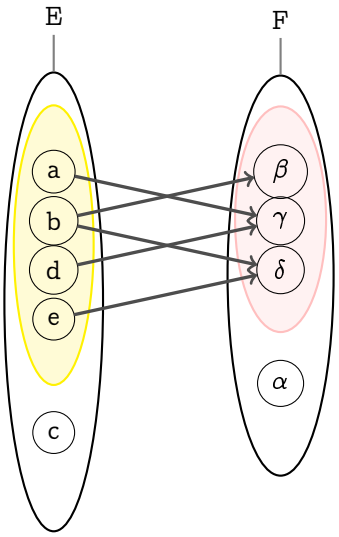
$$G_{\mathcal{R}} = \{(a, \gamma), (b, \beta), (b, \delta), (d, \gamma), (e, \delta)\}$$

$G_{\mathcal{R}}$ est aussi noté $G_{\mathcal{R}} = \{a \mapsto \gamma, b \mapsto \beta, b \mapsto \delta, d \mapsto \gamma, e \mapsto \delta\}$.

Le domaine de $\mathcal{R}$ est $\text{dom}(\mathcal{R}) = \{a, b, d, e\}$, c'est l'ensemble constitué des éléments de E qui ont au moins une image dans F. Le codomaine de $\mathcal{R}$ est $\text{codom}(\mathcal{R}) = \{\beta, \gamma, \delta\}$, c'est l'ensemble constitué des éléments de F qui ont au moins un antécédent ou encore l'ensemble des images des éléments de E par $\mathcal{R}$.

2.6.2 Représentation sagittale

Les ensembles E et F sont représentés par des « patates » et un élément x de E (un point de la patate E) est relié à un élément y de F par une flèche orientée de E vers F si, et seulemnt si, $x\mathcal{R}y$.



Cette relation n'est pas totale car il y a au moins un élément de l'ensemble de départ E qui n'a pas d'image, en l'occurence c n'a pas d'image.

On visualise immédiatement $\text{dom}(\mathcal{R}) = \{a, b, d, e\}$ et $\text{Im}(\mathcal{R}) = \text{codom}(\mathcal{R}) = \text{Ran}(\mathcal{R}) = \{\beta, \gamma, \delta\}$.

2.6.3 Table relationnelle

C'est la même chose sous forme de tableau :

De	Vers
a	γ
b	β
b	δ
d	γ
e	δ

2.6.4 Dictionnaire

La même chose en plus ramassé :

De	Vers
a	γ
b	β,δ
d	γ
e	δ

2 6 5 Représentation matricielle

Si les ensembles E et F sont définis par :

$$E = \{x_1, x_2, \cdots, x_n\} \qquad F = \{y_1, y_2, \cdots, y_p\}$$

alors la relation $\mathcal{R}$ est définie par la matrice $R = (r_{i,j}) \in \mathfrak{M}_{n,p}$ définie par :

$$R = \begin{pmatrix} r_{1,1} & \cdots & r_{1,j} & \cdots & \cdots & r_{1,p} \\ \vdots & & \vdots & & & \vdots \\ r_{i,1} & & r_{i,j} & & & r_{i,p} \\ \vdots & & \vdots & & & \vdots \\ r_{n,1} & \cdots & r_{n,j} & \cdots & \cdots & r_{n,p} \end{pmatrix} \text{ avec } r_{i,j} = \begin{cases} 1 \text{ si } x_i \mathcal{R} y_j \\ 0 \text{ sinon} \end{cases}$$

$r_{i,j}$ étant l'élément se trouvant sur la $i^{ème}$ ligne et $j^{ème}$ colonne, les lignes étant numérotées du haut vers le bas et les colonnes de la gauche vers la droite. La matrice R est appelée la matrice d'adjacence ou d'incidence de la relation $\mathcal{R}$.
Si l'on reprend l'exemple précédent, il nous faut décider d'ordonner les éléments de E et de F et, pour s'assurer que celui qui va nous lire ne fera pas de confusions, il faut s'obliger à faire figurer les éléments à côté de la matrice d'autant plus que certains utilisateurs peuvent très bien décider d'un autre placement pour les éléments de E et de F puisque les éléments de E et de F ne sont pas indicés :

$\curvearrowright$	α	β	γ	δ
a	**0**	**0**	**1**	**0**
b	**0**	**1**	**0**	**1**
c	**0**	**0**	**0**	**0**
d	**0**	**0**	**1**	**0**
e	**0**	**0**	**0**	**1**

Mais bon, généralement, on place en ligne les élément de départ et en colonne les élément d'arrivée donc on peut se contenter de la matrice :

$$R = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

mais il faut être sûr de ce que l'on fait.

2 7 Relation transposée

Définition 3 - 4

La **relation transposée** de la relation $\mathcal{R} = (E, F, G_{\mathcal{R}})$ est la relation notée ${}^t\mathcal{R}$ définie par ${}^t\mathcal{R} = (F, E, G_{{}^t\mathcal{R}})$ avec $G_{{}^t\mathcal{R}} = \{(y, x) \mid (x, y) \in G_{\mathcal{R}}\}$, c'est-à-dire $y\,{}^t\mathcal{R}\,x \Leftrightarrow x\mathcal{R}y$.

Pour obtenir la représentation sagittale de ${}^t\mathcal{R}$, il suffit de modifier le sens des flèches de la représentation sagittale de $\mathcal{R}$.

Remarque

Si R est la matrice de $\mathcal{R}$ alors ${}^tR$, la transposée de R, est la matrice de ${}^t\mathcal{R}$, c'est-à-dire la matrice obtenue en inversant les lignes et les colonnes, ce qui est assez logique compte tenu de notre définition de la représentation matricielle.
En reprenant l'exemple précédent :

$$R = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^tR = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Les résultats suivants découlent immédiatement de la définition :

— $\text{dom}(\mathcal{R}) = \text{Im}({}^t\mathcal{R})$
— $\text{dom}({}^t\mathcal{R}) = \text{Im}(\mathcal{R})$
— ${}^t({}^t\mathcal{R}) = \mathcal{R}$

Un petit exemple alors...Considérons les ensembles et la relation suivants :

— Étudiants$= \{$Roger, Berthe, Jean-Pierre, Gudrun$\}$;
— Chaînes$= \{$ TF1, Gulli, Zen TV$\}$;
— TV$= \{$Roger $\mapsto$ Gulli, Berthe $\mapsto$ Gulli, Jean-Pierre $\mapsto$ Zen TV, Gudrun $\mapsto$ Gulli$\}$.

Ainsi la relation $TV \in$ Étudiants $\longleftrightarrow$ Chaîne peut s'interpréter en « ... regarde ... ».
Que pensez-vous de la relation réciproque ?

2 8 Image, contre image

Définition 3 - 5

Si U est une partie de E, $\mathcal{R}(U)$ désigne l'ensemble des images des éléments de U par $\mathcal{R}$.

Remarque

Si U n'est pas une partie de E, $\mathcal{R}(U)$ n'a pas de sens !

Définition 3 - 6

Si V est une partie de F, la contre image (ou l'image transposée) de V par $\mathcal{R}$ est l'image de V par ${}^t\mathcal{R}$. Elle est donc notée ${}^t\mathcal{R}(V)$.

On pourra remarquer que ${}^t\mathcal{R}(F) = \text{dom}\,\mathcal{R}$.
La relation $\mathcal{R}$ définie par son diagramme sagittal :

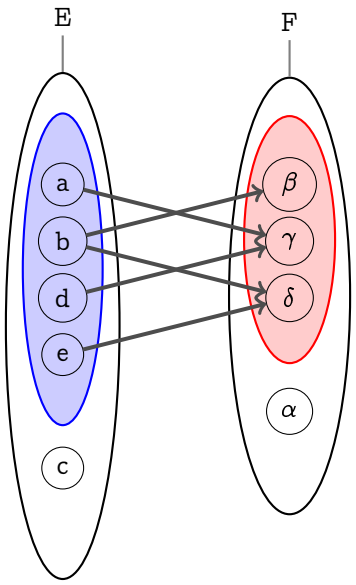


Image et contre-image

Nous avons $\mathcal{R}(\{a,b\}) = \{\beta,\gamma\}, \mathcal{R}(\{c\}) = \{\gamma\}, {}^t\mathcal{R}(\{\alpha\}) = \{\}, {}^t\mathcal{R}(\{\gamma\}) = \{a,c,d\}$ mais aussi $\mathcal{R}(\{\}) = \{\}$ et $\mathcal{R}(\{a,\beta\})$ n'a pas de sens car $\{a,\beta\}$ n'est pas une partie de l'ensemble de départ de $\mathcal{R}$.

Allez, vous voulez reprendre un exemple...

Qui regarde Gulli ? Il faut définir ${}^tTV(\{\text{Gulli}\})$.

2 9 Égalité de deux relations

Définition 3 - 7

Les relations $\mathcal{R}_1 = (E_1, F_1, G_{\mathcal{R}_1})$ et $\mathcal{R}_2 = (E_2, F_2, G_{\mathcal{R}_2})$ sont égales si, et seulement si, $E_1 = E_2$ **et** $F_1 = F_2$ **et** $G_{\mathcal{R}_1} = G_{\mathcal{R}_2}$.

Vous êtes priés de ne pas modifier cette définition en fonction des circonstances. Il faut comprendre que les relations :

$$\mathcal{R} = (E, F, G_{\mathcal{R}}) \text{ et } \mathcal{R}_1 = (\text{dom}(\mathcal{R}), \text{Im}(\mathcal{R}), G_{\mathcal{R}})$$

ne sont pas forcément égales même si on a l'impression que « c'est pareil ». Tout ce que l'on peut dire c'est qu'elles ont le même graphe et c'est déjà beaucoup. Imaginons que E et F soient deux fichiers *informatiques* et que $G_{\mathcal{R}}$ nous donne les couples de fiches correspondant à une certaine requête (relation). Ce n'est pas parce que certaines fiches (éléments des fichiers) ne seront pas utilisées que vous devez les ignorer ou les faire disparaître des fichiers !

2 10 Principales opérations sur les relations

Une relation $\mathcal{R}$ de E vers F étant assimilée à un ensemble, une partie de $E \times F$, on est naturellement amené à utiliser les opérations ensemblistes élémentaires.

Dans ce qui suit $\mathcal{R}_1 = (E, F, G_{\mathcal{R}_1})$ et $\mathcal{R}_2 = (E, F, G_{\mathcal{R}_2})$ sont deux relations de E vers F et R_1 et R_2 sont, respectivement, leurs matrices.

2 10 1 Négation

Définition 3 - 8

La **négation** de $\mathcal{R}_1$ (on dit aussi le complémentaire de $\mathcal{R}_1$) est la relation :

$$\overline{\mathcal{R}_1} = (E, F, \complement_{E\times F} G_{\mathcal{R}_1})$$

Cela signifie tout simplement que $x\overline{\mathcal{R}_1}y$ si, et seulement si, $(x,y) \notin G_{\mathcal{R}_1}$ ou encore que x n'est pas en relation avec y par $\mathcal{R}_1$.

La relation $\overline{\mathcal{R}_1}$ est aussi notée $\neg\mathcal{R}_1$.

Par exemple en reprenant la relation TV, Berthe $\overline{TV}$ TF1 : Berthe ne regarde pas TF1.

La matrice booléenne de $\overline{\mathcal{R}_1}$ s'obtient en remplaçant dans la matrice booléenne de $\mathcal{R}_1$ les « 0 » par des « 1 » et les « 1 » par des « 0 ». Cela revient à calculer $R_1 \oplus \text{Attila}(|E|, |F|)$.

2 10 2 Inclusion

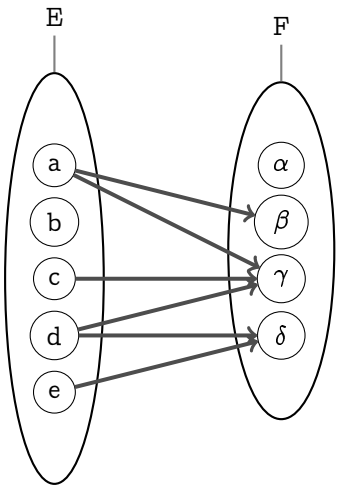
Définition 3 - 9

On dit que la relation $\mathcal{R}_1$ est incluse dans la relation $\mathcal{R}_2$ ou que la relation $\mathcal{R}_1$ est une **sous relation** de la relation $\mathcal{R}_2$ si, et seulement si, $G_{\mathcal{R}_1} \subseteq G_{\mathcal{R}_2}$. Cela signifie que

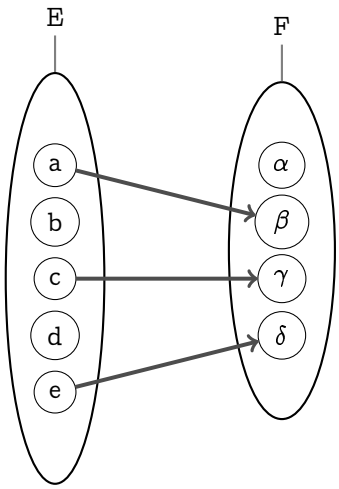
$$x\mathcal{R}_1 y \Rightarrow x\mathcal{R}_2 y$$

et on écrit alors $\mathcal{R}_1 \subseteq \mathcal{R}_2$.

Visuellement, une sous relation est obtenue en « supprimant » des flèches :



$\mathcal{R}_1$



$\mathcal{R}_2$ est une sous-relation de $\mathcal{R}_1$

Pour être plus concret, modifions un peu notre exemple télévisuel en supposant qu'on peut maintenant regarder plusieurs chaînes de télévision.

Par exemple, posons à présent TV2= $\{$Roger $\mapsto$ Gulli, Berthe $\mapsto$ Gulli, Berthe $\mapsto$ Zen TV, Jean-Pierre $\mapsto$ Zen TV, Gudrun $\mapsto$ Gulli, Gudrun $\mapsto$ TF1, Gudrun $\mapsto$ Zen TV$\}$.

Alors $TV \subseteq TV2$.

Matriciellement cela se traduit par le fait que tous les « 1 » de la matrice R_1 sont aussi des « 1 » de la matrice R_2 ce qui donne, en utilisant le calcul booléen ($1 \veebar 1 = 1$) :

$$\mathcal{R}_1 \subseteq \mathcal{R}_2 \Leftrightarrow R_1 \veebar R_2 = R_2$$

Méthode B

Voici les notations utilisées en méthode B, $\mathcal{R}$ désignant une relation de E vers F.

Obligatoirement $A \subseteq E$ **et** $B \subseteq F$.

Notations

- *La relation* $(E, F, G_{\mathcal{R}} \cap (A \times F))$ *est notée* $A \lhd \mathcal{R}$, *c'est une sous relation de* $\mathcal{R}$ *et les informaticiens parlent de « restriction de domaine ».*
- *La relation* $(E, F, G_{\mathcal{R}} \cap (E \times B))$ *est notée* $\mathcal{R} \rhd B$, *c'est une sous relation de* $\mathcal{R}$ *et les informaticiens parlent de « restriction de codomaine ».*
- *La relation* $(E, F, G_{\mathcal{R}} \cap (A \times B))$ *est notée* $A \lhd \mathcal{R} \rhd B$, *c'est une sous relation de* $\mathcal{R}$ *et les informaticiens parlent de « restriction de domaine et de codomaine ».*

2 10 3 Réunion et intersection

Définition 3 - 10

La **réunion** de $\mathcal{R}_1$ et de $\mathcal{R}_2$ est la relation $\mathcal{R}_1 \cup \mathcal{R}_2 = (E, F, G_{\mathcal{R}_1} \cup G_{\mathcal{R}_2})$.
L'**intersection** de $\mathcal{R}_1$ et de $\mathcal{R}_2$ est la relation $\mathcal{R}_1 \cap \mathcal{R}_2 = (E, F, G_{\mathcal{R}_1} \cap G_{\mathcal{R}_2})$.

Cela revient en fait à ajouter ou à enlever des flèches.

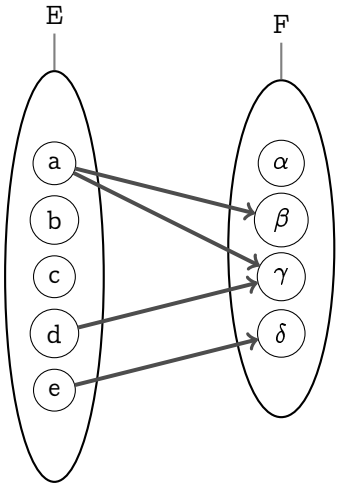
Danger

On notera bien que l'on ne peut comparer ou faire l'union ou l'intersection de deux relations qu'à la condition qu'elles aient le même ensemble de départ et le même ensemble d'arrivée.

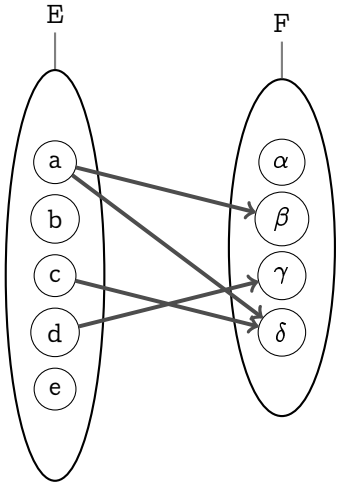
La matrice booléenne de $\mathcal{R}_1 \cup \mathcal{R}_2$ est $R_1 \veebar R_2$.

La matrice booléeen de $\mathcal{R}_1 \cap \mathcal{R}_2$ s'obtient « en ne gardant que les 1 qui ont la même place » dans R_1 et R_2. On va donc effectuer $E_1 \barwedge E_2$.

Rien ne vaut un bon exemple...Considérons deux relations $\mathcal{R}_1$ et $\mathcal{R}_2$:

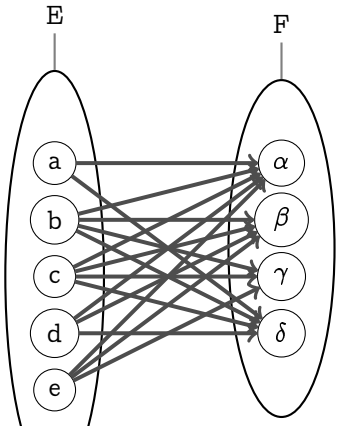


$\mathcal{R}_1$

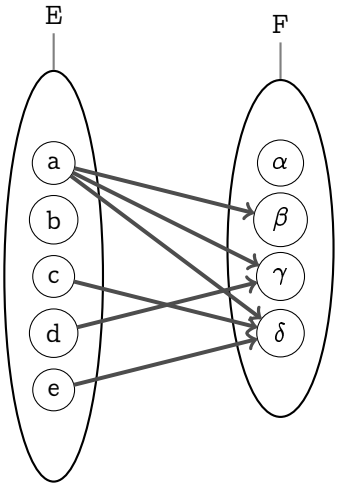


$\mathcal{R}_2$

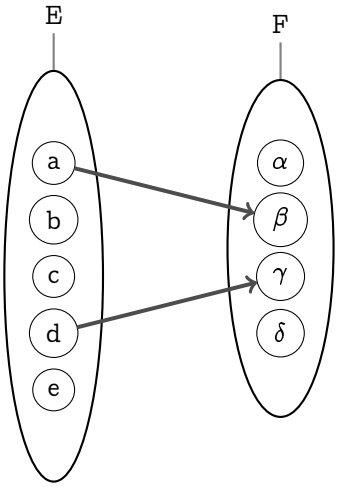
On remarque qu'aucune n'est incluse dans l'autre et on obtient :



$\overline{\mathcal{R}_1}$



$\mathcal{R}_1 \cup \mathcal{R}_2$



$\mathcal{R}_1 \cap \mathcal{R}_2$

Rallumons la télé mais cette fois en distinguant les chaînes regardées le midi de celles regardées le soir :
TVmidi= {Roger ↦ Gulli, Berthe ↦ Gulli, Berthe ↦ Zen TV, Jean-Pierre ↦ Zen TV, Gudrun ↦ Gulli, Gudrun ↦ TF1, Gudrun ↦ Zen TV}.
TVsoir== {Roger ↦ TF1, Berthe ↦ Gulli, Berthe ↦ TF1, Jean-Pierre ↦ TF1, Gudrun ↦ Gulli, Gudrun ↦ TF1}.
Comment décririez-vous les relations TVmidi ∩ TVsoir et TVmidi ∪ TVsoir ?

2 11 Composition de relations

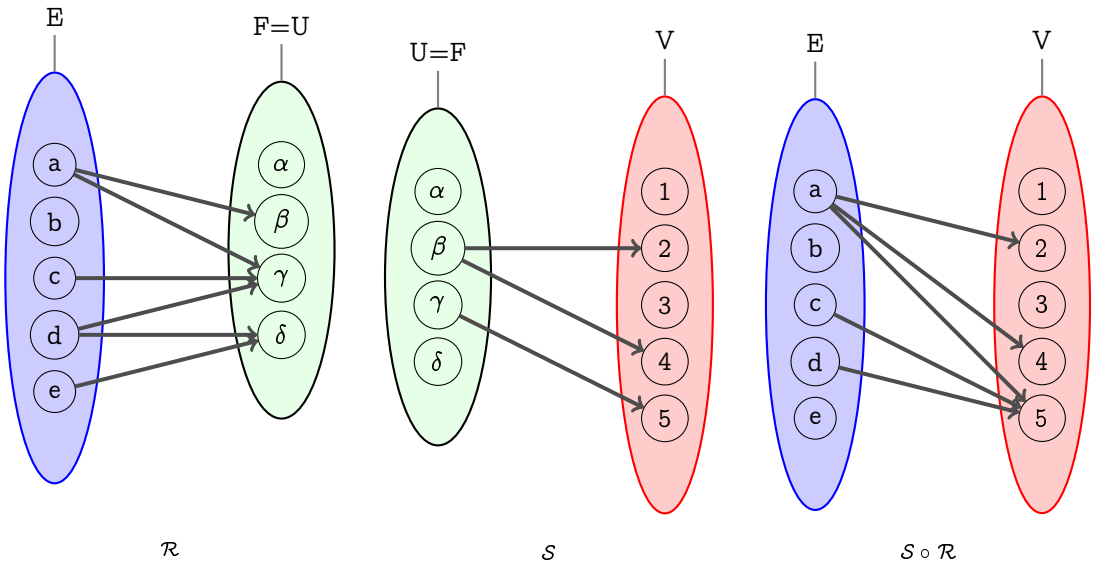
Définition 3 - 11

$\mathcal{R} = (E, F, G_{\mathcal{R}})$ et $\mathcal{S} = (U, V, G_{\mathcal{S}})$ sont deux relations. La composée de $\mathcal{R}$ et $\mathcal{S}$ est la relation $(E, V, G_{\mathcal{S}\circ\mathcal{R}})$ avec

$$G_{\mathcal{S}\circ\mathcal{R}} = \{(x,z) \in E \times V \mid \exists y \in F,\ (x,y) \in G_{\mathcal{R}} \wedge (y,z) \in G_{\mathcal{S}}\}$$

La **composition** (ou la composée) de $\mathcal{R}$ par $\mathcal{S}$ (attention à l'ordre) **ne peut se faire que si** $F = U$ ou, si l'on préfère, que si l'ensemble d'arrivée de $\mathcal{R}$ est égal à l'ensemble de départ de $\mathcal{S}$.
Dans ce cas la composée de $\mathcal{R}$ par $\mathcal{S}$ est la relation notée $\mathcal{S} \circ \mathcal{R}$ qui a pour ensemble de départ E (l'ensemble de départ de $\mathcal{R}$), pour ensemble d'arrivée V (l'ensemble d'arrivée de $\mathcal{S}$)
Pour simplifier les choses, $\mathcal{S} \circ \mathcal{R}$ est aussi notée $\mathcal{R}; \mathcal{S}$ ($\mathcal{R}$ suivie de $\mathcal{S}$: attention à l'ordre !). En effet, si $x\mathcal{R}y$ et $y\mathcal{S}z$ il est plus « naturel » d'écrire $x\mathcal{R}.\mathcal{S}z$.

Le mieux est encore d'analyser un exemple :



Recherche

Si R est la matrice de $\mathcal{R}$ et S la matrice de $\mathcal{S}$ alors quelle sera la matrice de R ; S ?

Recherche

Que pensez de ${}^t\mathcal{R}\circ\mathcal{R}$ ou de $\mathcal{R}\circ{}^t\mathcal{R}$? Regardez ce que cela donne avec la relation $\mathcal{R}$ introduite précédemment.
Quelle est la relation transposée de $\mathcal{S}\circ\mathcal{R}$

Un exemple ? Allez, on va changer un peu. Cette fois considérons des animaux, leurs petits noms et leurs cris :

— Baptême=$\{$ alouette$\mapsto$Josette, alouette$\mapsto$Pépette, albatros$\mapsto$Bernard, bécasse$\mapsto$Germaine$\}$;
— Cri=$\{$Josette$\mapsto$turlute, Josette$\mapsto$grisole, Pépette$\mapsto$turlute, Bernard$\mapsto$piaule, Germaine$\mapsto$croule$\}$

Quelle est la tête de la relation (Baptême ; Cri) et comment l'interpréter ?

3 Fonctions

Une fonction, c'est quelque chose qui transforme un objet en un autre...

3.1 Définitions

Définition 3 - 12

On dit que **la relation** $f=(E,F,G_f)$ est une **fonction** de E vers (ou dans) F si, et seulement si, tout élément de E a **au plus** (cela veut dire : soit zéro ou une) une image dans F.

Cela signifie aussi que si x est un élément quelconque de E, $f(\{x\})$ est soit l'ensemble vide soit un singleton. Si x a au moins une image, c'est-à-dire s'il existe $y\in F$ tel que $(x,y)\in G_f$, y est forcément unique et est noté $f(x)$. Lorsque la relation $f=(E,F,G_f)$ est une fonction on préfère le plus souvent utiliser la notation :

$$f:\begin{array}{ccc} E & \to & F \\ x & \mapsto & f(x)=y \end{array}$$

qui se lit « f est une fonction de E dans F qui à x associe $f(x)$ ». On note $\mathcal{F}(E,F)$ ou F^E l'ensemble des fonctions de E dans F.
Lorsque E est de la forme $U^n=U\times U\times\ldots\times U$, on dit que la fonction de U^n dans F est une fonction de n variables

$$f : \begin{array}{ccc} U^n & \to & F \\ (u_1, u_2, \ldots, u_n) & \mapsto & f(u_1, u_2, \ldots, u_n) \end{array}$$

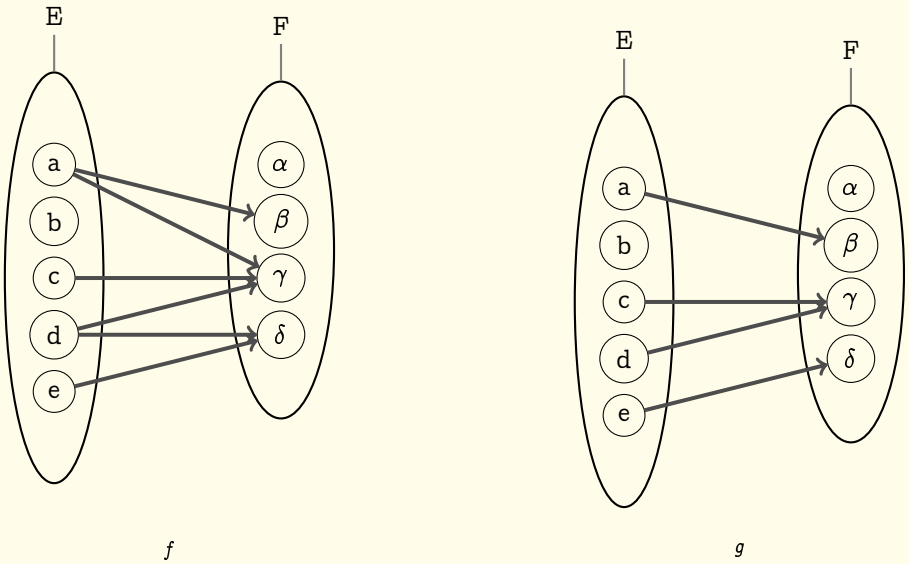
et u_i est appelée la $i^{ème}$ variable de f.

Définition 3 - 13

Si la fonction $f = (E, F, G)$ vérifie $\text{dom}(f) = E$ on dit que f est une **fonction totale** (ou une **application**) de E dans F.

Une *application* en anglais de spécialité se dit *map*, comme un plan de ville.
Pensez à une application comme une fonction qui aux points de la ville de Nantes associe les points correspondant sur le plan de la ville de Nantes. Une bonne carte est une application...

Recherche



f g

Parmi les deux relations représentées, y a-t-il une fonction ? une fonction totale ?

3 2 Fonctions particulières

On est très souvent amené à utiliser des mêmes « types » de fonctions, il est alors naturel de fixer un certain vocabulaire pour éviter à chaque fois de redéfinir ces fonctions.

Définition 3 - 14

Projection canonique
La fonction totale

$$\pi_i : \begin{array}{ccc} E_1 \times E_2 \times \ldots \times E_n & \to & E_i \\ (x_1, x_2, \ldots, x_n) & \mapsto & x_i \end{array}$$

est appelée projection canonique de $E_1 \times E_2 \times \ldots \times E_n$ sur E_i.
Si tous les E_i sont égaux au même ensemble E, π_i est appelée la i^e projection.

Définition 3 - 15

Fonction identité
C'est la fonction totale définie sur E par

$$\text{Id}_E : \begin{array}{ccc} E & \to & E \\ x & \mapsto & x \end{array}$$

Définition 3 - 16

Composition réitérée
Si f est une fonction totale de E dans E, on note f^k la composition

$$\underbrace{f \circ f \circ \cdots \circ f}_{k \text{ fois}}$$

si k est un entier naturel non nul et, par convention, $f^0 = id_E$.

Définition 3 - 17

Fonction caractéristique

$$\mathbb{1}_A : \begin{array}{ccc} E & \to & \{0,1\} \\ x & \mapsto & \begin{cases} 1 \text{ si } x \in A \\ 0 \text{ si } x \notin A \end{cases} \end{array}$$

Définition 3 - 18

Loi de composition

On appelle loi de composition (ou opération) dans E toute fonction de $A \times E$ dans E.
Si $A = E$, on dit que la loi est une loi de composition interne, sinon on parle de loi de composition externe ou loi mixte.

Si f est une telle loi de composition et $(a, x) \in A \times E$, on remplace souvent la **notation préfixée** $f((a, x))$ par une **notation infixée** du style $a \circledcirc x$ ou $a * x$ ou $a \boxplus x$ ou $a \boxdot x$ ou $a + x$ ou...
Sur Haskell, on peut définir de nouveaux opérateurs infixes :

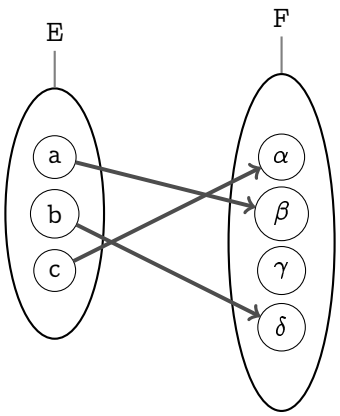
Haskell

```
Prelude> let biplus = \(a,b) -> a + b + b
Prelude> :t biplus
biplus :: (Integer, Integer) -> Integer
Prelude> let (++) a b = biplus (a,b)
Prelude> 1 ++ 2
5
Prelude> biplus (1,2)
5
```

3 3 Fonction injective

Définition 3 - 19

La fonction $f : E \longrightarrow F$ est **injective** (ou est une injection) si, et seulement si, tout élément de F a au plus un antécédent.



Injection

Cette définition équivaut à dire que :

— f^{-1} est une fonction.

— il n'existe pas deux éléments différents de E qui ont la même image dans F. En d'autres termes deux éléments différents ont toujours des images différentes.

— il n'existe pas d'élément de F ayant plus d'un antécédent.

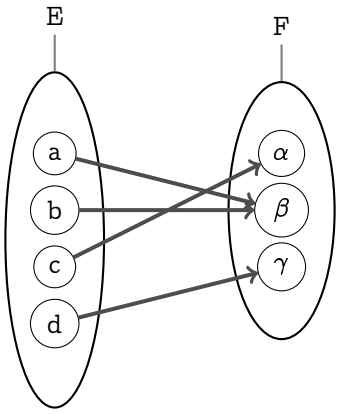
Mathématiquement cela s'écrit :

$$\begin{aligned} f \text{ injective} &\equiv \forall (x, x') \left((x, x') \in E^2 \wedge x \neq x' \to f(x) \neq f(x') \right) \\ f \text{ injective} &\equiv \forall (x, x') \left((x, x') \in E^2 \wedge f(x) = f(x') \to x = x' \right) \end{aligned}$$

3 4 Fonction surjective

Définition 3 - 20

La fonction $f : E \longrightarrow F$ est **surjective** (ou est une surjection de E) sur F si, et seulement si, tout élément de F a au moins un antécédent.



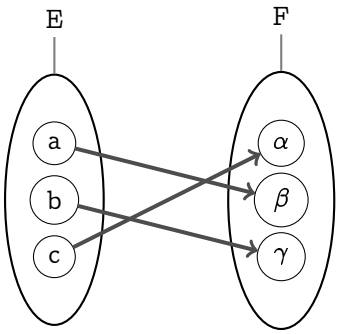
Surjection

Cela équivaut à écrire que $\boldsymbol{Im}(f) = F$ ou que $f(E) = F$ ou encore que la relation réciproque de f est une relation totale.

3 5 Fonction bijective

Définition 3 - 21

La fonction $f : E \longrightarrow F$ est **bijective** ou **biunivoque** ou est une bijection si, et seulement si, elle est à la fois injective et surjective.



Notations

Méthode B

En méthode B on note :

- $D \mapsto C$: fonctions partielles de D vers C ;
- $D \rightarrow C$: fonctions totales de D vers C ;
- $D \rightarrowtail C$: injections partielles de D vers C ;
- $D \rightarrowtail C$: injections totales de D vers C ;
- $D \twoheadrightarrow C$: surjections partielles de D vers C ;
- $D \twoheadrightarrow C$: surjections totales de D vers C ;
- $D \nrightarrow\!\!\!\!\!\twoheadrightarrow C$: bijections partielles de D sur C ;
- $D \rightarrowtail\!\!\!\!\!\twoheadrightarrow C$: bijections totales de D sur C.

En mathématique, on travaillera le plus souvent avec des applications.

4 Relations binaires sur un ensemble

4 1 Au CE1

4 2 Définition

Définition 3 - 22

On appelle **relation binaire sur l'ensemble** E tout triplet du type

$$\mathcal{R} = (E, E, G_{\mathcal{R}})$$

où $G_{\mathcal{R}}$ est une partie de $E \times E$.

Une relation binaire sur l'ensemble E est donc tout simplement une relation « classique » qui a son ensemble de départ égal à son ensemble d'arrivée.

4 3 Représentations

On aurait envie d'utiliser le diagramme sagittal habituel mais il est cependant plus « économique » de n'utiliser qu'une « patate » car nous restons dans E. On symbolise alors les relations du type $a \mathcal{R} a$ à l'aide de *boucles* :

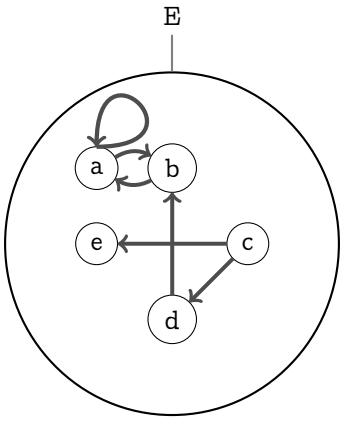


Diagramme sagittal d'une relation sur un ensemble (version 2)

On peut bien sûr utiliser une matrice carrée d'ordre n.
Si $E = \{x_1, x_2, \cdots, x_n\}$, la matrice :

$$M = (m_{i,j}) \text{ avec } \begin{cases} m_{i,j} = 1 \text{ si } x_i \mathcal{R} x_j \\ m_{i,j} = 0 \text{ si } (x_i, x_j) \notin G_{\mathcal{R}} \end{cases}$$

est appelée la *matrice d'adjacence* de $\mathcal{R}$ ou du graphe $\mathcal{R}$. On dit aussi que M est la *matrice booléenne* de la relation $\mathcal{R}$, dans ce cas, les calculs matriciels utilisent le fait que « $1 + 1 = 1$ ».

4 4 Composition

Théorème 3 - 1

— Si la relation $\mathcal{R}$ n'est pas la relation vide, on pose $\mathcal{R}^0 = Id_E$ sinon c'est la relation vide (de E dans E).

— k désignant un entier naturel, une définition récursive de $\mathcal{R}^k$ est la suivante :

$$i \in \mathbb{N}, \mathcal{R}^{i+1} = \mathcal{R}^i \ ; \ \mathcal{R} \circ \mathcal{R}^i$$

Remarque

Il faut bien comprendre que si $x\mathcal{R}^k y$ alors il existe

$$u_1, u_2, \ldots, u_{k-1} \text{ éléments de } E$$

vérifiant

$$x\mathcal{R}u_1 \text{ et } u_1\mathcal{R}u_2 \text{ et } u_{21}\mathcal{R}u_3 \text{ et } \ldots \text{ et } u_{k-1}\mathcal{R}y$$

et l'on dit que y est un *descendant* de x ou que x est un *ascendant* de y.

4 5 Chemin

Nous parlerons l'an prochain de graphes directionnels et de chemin sur ces graphes. Grosso modo, un graphe directionnel, c'est des sommets reliés par des flèches et un chemin c'est un parcours en suivant ces flèches.
Nous pouvons reprendre l'idée pour une relation :

Définition 3 - 23

Soit $\mathcal{R}$ une relation. Il existe un **chemin** de a vers b dans $\mathcal{R}$ s'il existe une suite d'éléments $x_1, x_2, x_3, \ldots, x_{n-1}$ telle que $(a, x_1), (x_1, x_2), (x_2, x_3), \ldots, (x_{n-1}, b)$ appartiennent à $\mathcal{R}$.
Ce chemin est de **longueur** n.

En reprenant la relation représentée sur la figure page précédente, il existe un chemin (et même plusieurs !) de a vers b, passant par exemple par $a, a, a, b, a, a, b, a, a, a, a, b$.
Le théorème suivant nous sera très utile :

Théorème 3 - 2

Soit $\mathcal{R}$ une relation sur un ensemble et n un entier naturel. Il existe un chemin de longueur n de a vers b si, et seulement si, $(a, b) \in \mathcal{R}^n$.

La démonstration s'effectue par récurrence...et est laissée au lecteur attentif.

4 6 Propriétés des relations sur un ensemble

On notera Δ_E la relation $\left(E, E, \left\{(x, x) \mid x \in E\right\}\right)$ appelée également la « diagonale » de $E \times E$.

Définition 3 - 24

Relation réflexive
$\mathcal{R}$ est **réflexive** sur E si, et seulement si, pour tout x de E, on a $x\mathcal{R}x$.

En d'autres termes $\mathcal{R}$ est réflexive sur E si, et seulement si, $G_{\mathcal{R}}$ contient Δ_E car Δ_E est la plus petite relation réflexive sur E : pourquoi ?

Théorème 3 - 3

$\mathcal{R}$ est réflexive si, et seulement si, $\mathcal{R} \cup Id_E = \mathcal{R}$.

Définition 3 - 25

Fermeture réflexive
La **fermeture réflexive** d'une relation quelconque $\mathcal{R}$ sur E est la relation dont le graphe est $G_{\mathcal{R}} \cup \Delta_E$.
C'est la plus petite relation binaire sur E (au sens de l'inclusion) qui contient $\mathcal{R}$.

Définition 3 - 26

Relation irréflexive
$\mathcal{R}$ est irréflexive (on utilise aussi le mot « antiréflexif ») si, et seulement si, il n'existe pas $x \in E$ vérifiant $x\mathcal{R}x$.

Ne pas confondre une relation irréflexive avec une relation non réflexive !
Par exemple, la relation « ... est l'enfant de ... » est irréflexive...

Recherche

Déterminez une caractérisation d'une relation irréflexive à l'aide de Δ_E.

Définition 3 - 27

Relation symétrique
$\mathcal{R}$ est **symétrique** si, et seulement si, à chaque fois que l'on a $x\mathcal{R}y$ on a aussi $y\mathcal{R}x$.

Recherche

Considérons par exemple la relation :
$\text{Train} = \{(\text{Nantes,Paris}),(\text{Paris,Nantes}),(\text{Nantes,Vertou}),(\text{Vertou,Nantes}),$
$(\text{Paris,Vertou}),(\text{Vertou,Paris}),(\text{Paris,Sucé}),(\text{Sucé,Paris})\}$
Est-elle symétrique ?

Théorème 3 - 4

Matriciellement, $\mathcal{R}$ est symétrique si, et seulement si, sa matrice R est symétrique, c'est-à-dire ${}^tR = R$.

Définition 3 - 28

Fermeture symétrique
La fermeture symétrique de la relation $\mathcal{R} = (E, G_{\mathcal{R}})$ est la plus petite relation symétrique dont le graphe contient $G_{\mathcal{R}}$.

Théorème 3 - 5

La fermeture symétrique de $\mathcal{R}$ est $\mathcal{R} \cup \mathcal{R}^{-1}$.

Définition 3 - 29

Relation antisymétrique
$\mathcal{R}$ est antisymétrique si, et seulement si, à chaque fois que l'on a $x\mathcal{R}y$, avec x différent de y, on n'a pas $y\mathcal{R}x$. Cela équivaut à écrire que si on a simultanément $x\mathcal{R}y$ et $y\mathcal{R}x$, on a forcément $x = y$.

Le diagramme sagittal ne possède aucun « aller et retour ».

Définition 3 - 30

Relation transitive
$\mathcal{R}$ est transitive si, et seulement si, à chaque fois que l'on a $x\mathcal{R}y$ et $y\mathcal{R}z$, on a aussi $x\mathcal{R}z$.

Si « on peut aller de x à z en passant par y, on doit pouvoir aller directement de x à z ».

Recherche

Considérons à nouveau la relation :
$\text{Train} = \{(\text{Nantes,Paris}),(\text{Paris,Nantes}),(\text{Nantes,Vertou}),(\text{Vertou,Nantes}),$
$(\text{Paris,Vertou}),(\text{Vertou,Paris}),(\text{Paris,Sucé}),(\text{Sucé,Paris})\}$
Est-elle transitive ?

Théorème 3 - 6

- $\mathcal{R}$ transitive $\Leftrightarrow \mathcal{R}^2 \subseteq \mathcal{R}$.
- $\mathcal{R}$ réflexive et transitive $\Rightarrow \mathcal{R}^2 = \mathcal{R}$.
- $\mathcal{R}$ est transitive si, et seulement si, $\mathcal{R}^n \subseteq \mathcal{R}$ pour tout entier naturel non nul n.

Définition 3 - 31

Fermeture transitive
La fermeture transitive de la relation $\mathcal{R} = (E, G_{\mathcal{R}})$, est la plus petite (au sens de l'inclusion) relation transitive sur E dont le graphe contient $G_{\mathcal{R}}$.

4 7 Fermeture transitive et chemin

En fait, trouver la fermeture transitive d'une relation, c'est déterminer les couples reliés par un chemin.

Définition 3 - 32

Soit $\mathcal{R}$ une relation sur un ensemble. La relation de connexion $\mathcal{R}^+$ contient tous les couples (a,b) tels qu'il existe un chemin sur $\mathcal{R}$ de longueur non nulle allant de a vers b.

Maintenant, souvenons-nous que $\mathcal{R}^n$ est formé des couples (a,b) tels qu'il existe un chemin de longueur n de a vers b. Il en découle que $\mathcal{R}^+$ est en fait la réunion de tous les $\mathcal{R}^n$:

Théorème 3 - 7

$$\mathcal{R}^+ = \bigcup_{n\in\mathbb{N}^*} \mathcal{R}^n$$

Recherche

Par exemple, si $\mathcal{R}$ est la relation sur l'ensemble des département français qui contient (a,b) si les départements a et b ont une frontière commune, quelle est $\mathcal{R}^n$? $\mathcal{R}^+$?

Nous en arrivons au théorème important suivant :

Théorème 3 - 8

La fermeture transitive d'une relation $\mathcal{R}$ est égale à $\mathcal{R}^+$.

Pour les curieux, on peut survoler une petite démonstration de ce résultat.
Il est assez simple de montrer que $\mathcal{R}^+$ est transitive (faites-le...) et que $\mathcal{R}^+$ contient $\mathcal{R}$.
Maintenant, il faudrait montrer que n'importe quelle relation $\mathcal{S}$ transitive qui contient $\mathcal{R}$ contient aussi $\mathcal{R}^+$.
Comme $\mathcal{S}$ est transitive, alors $\mathcal{S}^n$ aussi et $\mathcal{S}^n \subseteq \mathcal{S}$.
Or $\mathcal{S}^+ = \bigcup_{n\in\mathbb{N}^*} \mathcal{S}^n$ donc $\mathcal{S}^+ \subseteq \mathcal{S}$. Or $\mathcal{R} \subseteq \mathcal{S}$ donc $\mathcal{R}^+ \subseteq \mathcal{S}^+$.
Finalement $\mathcal{R}^+ \subseteq \mathcal{S}^+ \subseteq \mathcal{S}$ ce qui achève notre démonstration.
Bon, on sait ce que c'est que la fermeture transitive : il reste à savoir comment la déterminer concrètement...

Aparté

SQL
Il existe sur certains dérivés de SQL d'obtenir la fermeture transitive d'une relation. Sur Oracle, on peut utiliser **`start with`** et **`connect by`**.

Fermeture d'une relation : du concret

La fermeture transitive de la relation « ...est le père ou la mère de... » sur l'ensemble des humains existant ou ayant existé est « ...est un(e)ancêtre de... ».
Vous gérez un réseau informatique avec des centres situés à Vertou, Saint Mars du Désert, Chavagne en Paillé, Guéméné Penfao et des lignes de câbles unidirectionnelles de Vertou vers Saint Mars, de Vertou vers Chavagne, de Chavagne vers Guéméné, de Guéméné vers Saint Mars. Si $\mathcal{R}$ est la relation qui contient le couple (a,b) s'il y a une ligne entre a et b, on voudrait savoir si on peut connecter deux centres donnés. Le chemin de Vertou vers Guéméné existe mais est indirect car il passe par Chavagne : la relation $\mathcal{R}$ n'est donc pas transitive car elle ne contient pas tous les couples pouvant se connecter. La fermeture transitive est en fait la plus petite relation transitive contenant $\mathcal{R}$. C'est concrètement la relation qui comprend tous les couples de centres pouvant être connectés.
On peut dans le même esprit s'occuper des fermetures symétrique, réflexive, réflexive-transitive, etc. Pour les fermetures réflexive et symétrique, c'est assez immédiat. Notre problème sera de trouver des algorithmes pour la fermeture transitive car c'est moins simple.

Recherche

Essayez de déterminer la fermeture transitive de $\mathcal{R} = \{(1,3),(1,4),(2,1),(3,2)\}$ sur l'ensemble $\mathcal{E} = \{1,2,3,4\}$.

5 Un peu d'ordre

5 1 Pré-ordre

Définition 3 - 33

Un pré-ordre sur E est une relation binaire sur E qui est **reflexive** et **transitive**.

Par exemple, la relation « ...divise... » est un pré-ordre sur $\mathbb{Z}$: sauriez-vous le démontrer ?
La fermeture réflexive et transitive d'une relation est un pré-ordre.

Aparté

Un pré-ordre important en informatique est la relation binaire définie sur l'ensemble des *types* « ...est un sous-type de... ».
Cela est bien au-delà de notre propos d'aujourd'hui mais on peut en avoir une idée intuitive.
Par exemple **`Int32`** (les entiers sur 32 bits) **`Int64`** (les entiers sur 64bits) **`Nat`** (les entiers positifs) peuvent être considérés comme des sous-types d'un « super-type » **`Int`**.
Cela peut rendre des services au programmeur qui créera une fonction « polymorphe » sur **`Int`** qui sera valable pour ses sous-types.
Mais il est encore prématuré de parler de tout ceci pour de jeunes Padawans...

5 2 Relations d'équivalence

Définition 3 - 34

Un pré-ordre symétrique est une relation d'équivalence. C'est donc une relation binaire **Réflexive et Symétrique et Transitive (R.S.T.)**

Par exemple, l'égalité des ensembles définie par l'axiome d'extensionalité est une relation d'équivalence.
On peut donc l'ensemble des nombres pairs, l'ensemble des nombres dont le reste par la division par 2 est nul, l'ensemble des nombres dont l'écriture binaire se termine par 0, l'ensemble des entiers privé des nombres impairs, l'ensemble des successeurs des entiers impairs, l'ensemble des doubles d'entiers,... en une seule « classe » :

Définition 3 - 35

x désignant un élément de E, la **classe d'équivalence** modulo $\mathcal{R}$ de x est l'ensemble

$$[x]_{\mathcal{R}} = \{y \mid x\mathcal{R}y\} = \{y \mid y\mathcal{R}x\}$$

On note $E/\mathcal{R}$ l'ensemble des classes d'équivalence sur E modulo $\mathcal{R}$ dont les éléments sont appelés *représentants* de la classe.

Une classe est donc un élément de $\mathcal{P}(E)$ c'est-à-dire une partie (un sous-ensemble) de E.
On va définir en informatique de nombreux types nouveaux. On aura souvent besoin de leur associer une « égalité » qui est en fait un e relation d'équivalence.

Théorème 3 - 9

Soit $\mathcal{R}$ une relation d'équivalence sur E.

- $x\mathcal{R}y \leftrightarrow [x]_{\mathcal{R}} = [y]_{\mathcal{R}}$;
- Chaque classe est non vide ;
- Chaque élément de E appartient à une classe modulo $\mathcal{R}$;
- Chaque classe distincte est disjointe.

À démontrer...

Théorème 3 - 10

Chaque quotient d'un ensemble modulo une relation est une partition de cet ensemble.

Il n'y a pas grand chose à démontrer si vous avez fait le boulot demandé juste au-dessus...

Théorème 3 - 11

Toute partition de E définit une relation d'équivalence sur E.

À vous de trouver une démonstration si vous avez un peu compris ce qui se passe...

5 3 Relation d'ordre - Poset

5 3 1 Définitions

Définition 3 - 36

Un pré-ordre antisymétrique sur un ensemble E est une relation d'ordre sur E. C'est donc une relation binaire **Réflexive et Antisymétrique et Transitive (R.A.T.)**.
L'ensemble E est alors un **poset** (*Partially Ordered Set* in english of speciality...) aussi appelé dans une langue bizarre **ensemble ordonné**.

Soit E un ensemble : vérifiez que $\mathcal{P}(E)$ muni de $\subseteq$ est un ensemble ordonné.
Est-ce que tous les éléments de $PR(E)$ peuvent être comparés ?

Définition 3 - 37

Un ensemble ordonné dont *tous* les éléments peuvent être comparés est un **ensemble TOTALEMENT ordonné** (on parle alors d'**ordre total**).
Sans précision, on parle d'**ordre partiel** : tous les éléments ne sont pas forcément comparables.
Un ensemble totalement ordonné est aussi appelé une **chaîne**.

En informatique, ces notions sont très importantes, notamment dans l'étude des graphes et des arbres en particulier.
En général une relation d'ordre est notée par $\leq$ ou par $\preceq$ ou par $\leqslant$.
La relation transposée de la relation d'ordre $\leqslant$ est notée $\geqslant$: $x \leqslant y \leftrightarrow y \geqslant x$
Est-ce une relation d'ordre ?

Danger

Ordre strict
Une relation $\mathcal{R}$ est une relation d'**ordre strict** si, et seulement si, $\mathcal{R}$ est **Irréflexive et Antisymétrique et Transitive**.
Attention, **ce n'est pas une relation d'ordre** ! ! ! (pourquoi ?...).
Il s'agit par exemple de la relation $<$ sur $\mathbb{N}$.

5 3 2 Diagramme de Hasse

Dresser un diagramme sagittal d'un poset peut s'avérer parfois pénible : il y a trop de flèches dans tous les sens...Nous allons voir un moyen de visualiser plus nettement la situation d'ensembles ordonnés. Nous aurons besoin tout d'abord de préciser les notions de prédécesseur et de successeur.

Définition 3 - 38

On appelle **éléments consécutifs** de l'ensemble ordonné $(E, \leqslant)$ deux éléments a et b qui vérifient

$$\begin{cases} a \leqslant b \wedge a \neq b \\ a \leqslant c \leqslant b \rightarrow a = c \vee b = c \end{cases}$$

a est un prédécesseur (immédiat) de b ou b est un successeur (immédiat) de a.

Tout ça pour dire que si a et b sont consécutifs, il n'y a personne entre eux, ils sont différents et l'un est plus grand que l'autre.
Revenons à nos ensembles ordonnés : pour ne pas avoir trop de flèches, on convient de ne relier x à y que si x est un prédécesseur de y et on place x sous y. On obtient alors un **diagramme de Hasse**.
En fait, il s'agit d'un diagramme sagittal où on élimine les boucles puisqu'on sait que la relation est réflexive. On élimine aussi les « raccourcis » : s'il y a une flèche de a vers b , de b vers c et de a vers c, on élimine la dernière puisqu'on sait que la relation est transitive. Enfin, on n'a pas besoin de flèches car on « pointe vers le haut ».
Considérons par exemple l'ensemble $\mathcal{E} = \{2, 3, 6, 7, 8, 9, 12, 18\}$ et l'ordre partiel $\{(a, b) \mid a \text{ divise } b\}$.
Le diagramme de Hasse correspondant est :

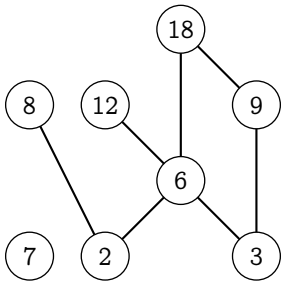


Diagramme de Hasse de $\{\{2,3,6,7,8,9,12,18\},|\}$

Voici un autre exemple pour $\{\mathcal{P}(\{a,b,c\}),\subseteq\}$:

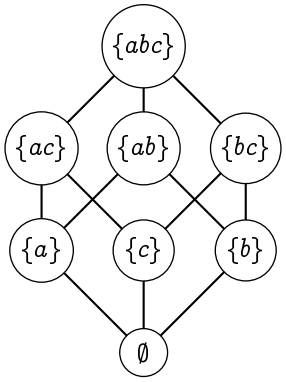


Diagramme de Hasse de $\{\mathcal{P}(\{a,b,c\}),\subseteq\}$

5 3 3 Éléments remarquables

Dans tout ce qui suit $(E,\leqslant)$ est un ensemble ordonné et A est une partie de E.

Définition 3 - 39

On dit que la partie A est **majorée** par $u\in E$ si, et seulement si, on a :

$$\forall x(x\in A\rightarrow x\leqslant u)$$

On dit que la partie A est **minorée** par $v\in E$ si, et seulement si, on a :

$$\forall x(x\in A\rightarrow v\leqslant x)$$

Théorème 3 - 12

On dit que g est le **plus grand élément** (ou élément maximum) de A si, et seulement si,

$$\forall x((g\in A)\wedge(x\in A\rightarrow x\leqslant g))$$

Si g existe, il est unique. On le note $\mathrm{Max}(A)$.
C'est le plus petit majorant de A.

Démontrez ce résultat...

Théorème 3 - 13

On dit que p est le **plus petit élément** (ou élément minimum) de A si, et seulement si,

$$\forall x((p\in A)\wedge(x\in A\rightarrow p\leqslant x))$$

Si p existe, il est unique. On le note $\mathrm{Min}(A)$.
C'est le plus grand minorant de A.

Démontrez ce résultat...

Définition 3 - 40

Considérons $\mathcal{G}$ l'ensemble des majorants de A. Si $\mathcal{G}$ admet un plus petit élément s c'est la **borne supérieure** de A.

Bien remarquer que s n'existe pas toujours et que s'il existe, ce n'est pas forcément un élément de A ; par contre si A admet un plus grand élément c'est sa **borne supérieure** . La borne supérieure de A dans E est notée $\mathrm{Sup}_E(A)$ ou $\mathrm{Sup}(A)$.

Définition 3 - 41

Considérons $\mathcal{P}$ l'ensemble des minorants de A. Si $\mathcal{P}$ admet un plus grand élément i c'est la **borne inférieure** de A.

Bien remarquer que i n'existe pas toujours et que s'il existe, ce n'est pas forcément un élément de A ; par contre si A admet un plus petit élément c'est sa **borne inférieure** . La borne inférieure de A dans E est notée $\mathrm{Inf}_E(A)$ ou $\mathrm{Inf}(A)$.

Définition 3 - 42

$a \in A$, a est **maximal** dans A s'il n'existe pas d'élément $x \in A$ vérifiant $x \neq a$ et $a \leqslant x$.
$\alpha \in A$, α est **minimal** dans A s'il n'existe pas d'élément $x \in A$ vérifiant $x \neq \alpha$ et $x \leqslant \alpha$.

Si A admet un plus grand élément, ce plus grand élément est maximal et c'est le seul.
Si A admet un plus petit élément, ce plus petit élément est minimal et c'est le seul.

À retenir

Lorsque l'on peut représenter l'ensemble ordonné $(E, \leqslant)$ à l'aide d'un diagramme de Hasse, on voit apparaître les éléments minimaux ou maximaux éventuels de E ainsi que les bornes ou éléments extrémaux éventuels de toute partie de E.

Observons par exemple le diagramme de la figure 3.17 page ci-contre. L'ensemble $\mathcal{E}$ ne possède pas de plus grand élément ni de plus petit élément.
Cependant, 8, 12, 18 et 7 sont les éléments maximaux et 2, 3 et 7 sont les éléments minimaux.
La partie $W = \{8, 12\}$ de E n'admet pas de majorant, il n'y a pas dans E d'élément simultanément « plus grand ou égal » à 8 et 12. Il faut, ici, entendre « plus grand ou égal que » comme « être un multiple de » ! W admet un unique minorant dans E, c'est 2, qui divise simultanément 8 et 12 ; 2 étant le « plus grand » des minorants, c'est la borne inférieure de W.
La partie $V = \{2, 3, 6\}$ admet 3 majorants dans E, : ce sont 6, 12 et 18 ; 6 est la borne supérieure de V, c'est aussi le plus grand élément de V.

Observons maintenant le diagramme de Hasse de la figure 3.18 page précédente.
Le plus petit élément est l'ensemble vide car $\emptyset \subseteq A$ pour tout partie A de $\mathcal{E} = \{a, b, c\}$.
Le plus grand élément est $\mathcal{E}$ car toute partie A de $\mathcal{E}$ est incluse dans $\mathcal{E}$.

EXERCICES

Relations binaires

Exercice 3 - 1

$E_1 = \{a, b, c, d\}$, $E_2 = \{0, 1, 2\}$, $E_3 = \{\alpha, \beta, \gamma, \delta, \epsilon\}$. On considère les relations $\mathcal{R} = (E_1, E_2, G_\mathcal{R})$ et $\mathcal{S} = (E_2, E_3, G_\mathcal{S})$ avec

$$\begin{aligned} G_\mathcal{R} &= \{(a,0),(a,1),(c,1),(d,0)\} \\ G_\mathcal{S} &= \{(1,\beta),(2,\delta),(2,\varepsilon)\} \end{aligned}$$

1. Préciser $\mathrm{dom}\,\mathcal{R}$ et $\mathrm{Im}\,\mathcal{R}$.
2. $\mathcal{R}$ est-elle totale ?
3. Reprendre les questions précédentes avec $\mathcal{S}$.
4. Déterminer $\mathcal{S} \circ \mathcal{R}$. Préciser $\mathrm{dom}\,(\mathcal{S} \circ \mathcal{R})$ et $\mathrm{Im}\,(\mathcal{S} \circ \mathcal{R})$.
5. Déterminer les relations ${}^t\mathcal{R}$ et ${}^t\mathcal{S}$.
6. Vrai ou faux ?
   i. $a\mathcal{R}1$
   ii. $\mathcal{R}\,(b, 0)$
   iii. $(1, c) \in \mathcal{G}_{{}^t\mathcal{R}}$
   iv. $(a, \beta) \in \mathcal{G}_{\mathcal{S}\circ\mathcal{R}}$
   v. $(a, \beta) \in \mathcal{G}_{\mathcal{R};\mathcal{S}}$
7. ${}^t\mathcal{R}$ est-elle une fonction ?
8. Déterminer la relation ${}^t\mathcal{S}; {}^t\mathcal{R}$.
9. Déterminer $\mathcal{S}; \mathcal{R}$.

Exercice 3 - 2

$E = \{12, 13, 33, 51, 111, 128\}$, $F = \{2, 3, 4, 5, 6, 7\}$. On considère la relation $\mathcal{R}$ de E vers (ou dans) F définie par : $x \in E, y \in F, x\mathcal{R}y$ si, et seulement si, la somme des chiffres utilisés dans l'écriture de x (on travaille en base 10) est égale à y.

1. Déterminer le domaine et le codomaine de $\mathcal{R}$, son graphe et donner un diagramme sagittal.
2. $\mathcal{R}$ est-elle une fonction ? L'écriture $\mathcal{R}(33)$ est-elle licite ?
3. $\mathcal{R}$ est-elle une fonction totale ?
4. Donner le graphe de ${}^t\mathcal{R}$. L'écriture ${}^t\mathcal{R}(6)$ est-elle licite ?
5. Calculez si cela a un sens :
   i. $\mathcal{R}\,(6)$, $\mathcal{R}\,(12)$, $\mathcal{R}\,(\{12\})$, $\mathcal{R}(E)$.
   ii. ${}^t\mathcal{R}\,(\{6\})$, ${}^t\mathcal{R}(F)$, ${}^t\mathcal{R}(2)$, ${}^t\mathcal{R}\,(\{2\})$
6. Déterminez le graphe de la restriction de $\mathcal{R}$ à $\{33; 51\}$

Exercice 3 - 3

On considère les relations r et s suivantes :

$$\begin{aligned} r &= (E, E, G_r) \\ \text{avec } G_r &= \{(a,a),(a,b),(c,a),(b,c),(d,a)\} \\ s &= (E, F, G_s) \\ \text{avec } G_s &= \{(b,1),(c,1),(d,3)\} \end{aligned}$$

Pour les diagrammes qui suivent, il est exigé que l'ensemble de départ soit à gauche et l'ensemble d'arrivée à droite, si la question n'a pas de sens, dîtes-le en expliquant pourquoi.

1. Donner un diagramme sagittal de la relation $r; r = r^2$

2. Donner un diagramme sagittal de la relation $r ; r^{\sim}$
3. Donner un diagramme sagittal de la relation $r ; s$
4. Donner un diagramme sagittal de la relation $(\{1\} \lhd s^{\sim}) ; (r^{\sim} \rhd \{a, c, d\})$

Exercice 3 - 4

Donner la représentation sagittale d'une fonction totale de l'ensemble fini E dans l'ensemble fini F

1. non injective et non surjective ;
2. non injective mais surjective ;
3. injective mais pas surjective ;
4. bijective.

Exercice 3 - 5

Déterminez si les fonctions de $\mathbb{Z}$ dans $\mathbb{Z}$ suivantes sont injectives ou surjectives ou... ?

1. $f(n) = n - 1$;
2. $f(n) = n^2 + 12$;
3. $f(n) = n^3 + 37$;
4. $f(n) = \lfloor n/2 \rfloor$.

Exercice 3 - 6

Mêmes questions avec les fonctions de $\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ suivantes :

1. $f(m, n) = 2m - n$;
2. $f(m, n) = m^2 - n^2$;
3. $f(m, n) = m^2 + n^2$;
4. $f(m, n) = |m| - |n|$;
5. $f(m, n) = m - n$;
6. $f(m, n) = |n|$;
7. $f(m, n) = m$;
8. $f(m, n) = m^2 - 4$.

Exercice 3 - 7

f est une fonction totale de E dans F, A et B sont deux parties de E et A' et B' sont deux parties de F. Démontrer :

1. $A \subseteq B \to f(A) \subseteq f(B)$
2. $f(A \cup B) = f(A) \cup f(B)$
3. $f(A \cap B) \subseteq f(A) \cap f(B)$, donner un exemple où il n'y a pas égalité.
4. $A \subseteq f^{\sim}(f(A))$
5. $f^{\sim}(A' \cap B') = f^{\sim}(A') \cap f^{\sim}(B')$
6. $f(f^{\sim}(B')) \subseteq B'$
7. $A' \subseteq B' \to f^{\sim}(A') \subseteq f^{\sim}(B')$
8. $f^{\sim}(A' \cap B') = f^{\sim}(A') \cap f^{\sim}(B')$

Exercice 3 - 8

Donner une bijection de $\mathbb{N}$ sur $\mathbb{Z}$.

Relations binaires sur un ensemble

Exercice 3 - 9

Déterminez parmi les relations suivantes celles qui sont réflexives, transitives, symétriques, antisymétriques, irréflexives (on précisera les ensembles) :

1. ...a les mêmes parents que...
2. ...est le frère de...
3. ...n'a pas le même âge que...
4. est au moins aussi grand que...
5. $x + y$ est impair
6. $x + y$ est pair
7. xy est impair
8. $x + xy$ est pair

Exercice 3 - 10

Décrivez les fermetures transitives de :

1. ...a un an de plus que...

2. ...est le double de...
3. ...est le fils de...
4. ...a un lien dynamique vers la page web...
5. ...est relié par le train à...

Exercice 3 - 11

1. Donner la représentation matricielle des relations suivantes définies sur $\{1,2,3,4\}$:
   i. $\{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$;
   ii. $\{(1,1),(1,4),(2,2),(3,3),(4,1)\}$;
   iii. $\{(1,2),(1,3),(1,4),(2,1),(2,3),(2,4),(3,1),(3,2),(3,4),(4,1),(4,2),(4,3)\}$;
   iv. $\{(2,4),(3,1),(3,2),(3,4)\}$
2. Quelles sont, parmi ces relations, celles qui sont réflexives ? irréfléxives ? symétriques ? antisymétriques ? transitives ? Donner des algorithmes qui répondent à ces questions en prenant les relations en arguments et en effectuant des calculs sur les matrices.

Exercice 3 - 12

Combien d'éléments non nuls contient la matrice représentant la relation $\mathcal{R}$ sur l'ensemble $\mathcal{E} = \{1,2,3,\ldots,100\}$ si $\mathcal{R}$ est :

1. $\{(a,b) \mid a > b\}$;
2. $\{(a,b) \mid a \neq b\}$;
3. $\{(a,b) \mid a = b+1\}$;
4. $\{(a,b) \mid a = 0\}$;
5. $\{(a,b) \mid ab = 0\}$;
6. $\{(a,b) \mid a+b = 100\}$

Vous colorierez les « écrans » correspondant et vous écrirez un algorithme qui les dessine.

Exercice 3 - 13

Soit $\mathcal{R} = \{(a,b) \in \mathbb{Z}^2 \mid a \neq b\}$ Quelle est la fermeture réflexive de $\mathcal{R}$?

Exercice 3 - 14

Soit $\mathcal{R} = \{(a,b) \in \mathbb{N}^2 \mid a \text{ divise } b\}$ Quelle est la fermeture symétrique de $\mathcal{R}$?

Exercice 3 - 15

Soit $\mathcal{E} = \{1,2,3,4,5\}$ et $\mathcal{R}$ une relation sur $\mathcal{E}$ qui contient les couples $(1,3)$, $(2,4)$, $(3,1)$, $(3,5)$, $(4,3)$, $(5,1)$, $(5,2)$ et $(5,4)$. Déterminer :

1. $\mathcal{R}^2$
2. $\mathcal{R}^3$
3. $\mathcal{R}^4$
4. $\mathcal{R}^5$
5. $\mathcal{R}^6$
6. $\mathcal{R}^+$

Avez-vous utilisé le calcul matriciel ou des dictionnaires ? Quel outil vous semble le plus efficace ?

Exercice 3 - 16

Soit $\mathcal{R}$ la relation définie sur $\mathbb{N}$ par « ...est le successeur de.. ». Quelle est la fermeture transitive de $\mathcal{R}$? Sa fermeture transitive et réflexive ?

Pré-ordres

Exercice 3 - 17

Soit $E = \{1,2,3,4,5\}$ et $\mathcal{R}$ la relation sur E dont le graphe est :

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

Vérifiez qu'il s'agit d'une relation d'équivalence, déterminez la classe de 2 modulo $\mathcal{R}$ puis le quotient de E modulo $\mathcal{R}$.

Exercice 3 - 18

Sur $\mathbb{Z}$ on définit la relation $\mathcal{R}$: « ...a le même reste dans la division par 3... ».
Vérifiez qu'il s'agit d'une relation d'équivalence, déterminez la classe de 2 modulo $\mathcal{R}$ puis le quotient de E modulo $\mathcal{R}$.

Exercice 3 - 19

On considère l'ensemble de la population française et la relation « ...a un ascendant commun avec... ». Est-ce une relation d'équivalence ?

Exercice 3 - 20

On considère deux relations d'équivalence $\mathcal{R}$ et $\mathcal{S}$ définies sur un même ensemble E. Est-ce que leur réunion et leur intersection sont encore des relations d'équivalence ?

Exercice 3 - 21

On définit la relation $\mathcal{R}$ sur $\mathbb{R} \times \mathbb{R}$ par : $(x, y)\mathcal{R}(a, b) \leftrightarrow 2x - y = 2a - b$
Vérifiez qu'il s'agit d'une relation d'équivalence, déterminez la classe de $(0, 0)$, de $(1, -1)$ modulo $\mathcal{R}$ puis le quotient de $\mathbb{R} \times \mathbb{R}$ modulo $\mathcal{R}$.

Exercice 3 - 22

Soit $\mathcal{R}$ un pré-ordre sur E. Soit $\equiv$ la relation définie sur E par : $x \equiv y \leftrightarrow (x\mathcal{R}y) \wedge (y\mathcal{R}x)$
Est-ce que $\equiv$ est une relation d'équivalence ?

Exercice 3 - 23

E est un ensemble non vide et A est une partie non vide fixée de E. On considère la relation binaire $\mathcal{R}$ sur $\mathcal{P}(E)$ définie par : $X\mathcal{R}Y \Leftrightarrow X \cap A = Y \cap A$

1. Démontrer que $\mathcal{R}$ est une relation d'équivalence.
2. On suppose ici que $E = \{a, b, c, d\}$ et $A = \{a, d\}$. Déterminer toutes les classes d'équivalence.

Exercice 3 - 24

Soit $\mathcal{E}$ l'ensemble des fonctions dérivables de $\mathbb{R}$ dans $\mathbb{R}$.
On considère la relation $\mathcal{R}$ définie sur $\mathcal{E}$ qui contient toutes les couples (f, g) tels que $f'(x) = g'(x)$ pour tout réel x.

1. Est-ce que $\mathcal{R}$ est une relation d'équivalence ?
2. Décrire la classe de $f : x \mapsto x^2$.

Exercice 3 - 25

E est un ensemble totalement ordonné par $\leqslant$. On considère la relation binaire sur E^2 définie par :

$$(x, y) \preceq (x', y') \leftrightarrow \begin{cases} x < x' \text{ ou} \\ x = x' \text{ et } y \leqslant y' \end{cases}$$

Démontrer que c'est une relation d'ordre total. Cet ordre porte le nom d'ordre lexicographique, expliquer pourquoi ?

Exercice 3 - 26

Dans $\mathbb{N}^*$ on considère la relation notée « $\mid$ » qui est la relation « divise », $a \mid b$ se lit « a divise b » ou encore « b est un multiple de a » ; la définition mathématique de cette relation étant

$$a \mid b \text{ si, et seulement si, il existe } k \in \mathbb{N}^* \text{ vérifiant } b = ka$$

1. Démontrer que c'est une relation d'ordre sur $\mathbb{N}^*$. Cet ordre est-il total ? Donner des éléments comparables et non comparables.
2. Démontrer que c'est une relation d'ordre sur toute partie A de N^*.
3. On considère la relation divise dans $E = \{2, 4, 6, 8, 10, 12\}$.
   - i. Construire le diagramme sagittal de cette relation.
   - ii. Construire le diagramme de Hasse de cette relation.
   - iii. E admet-il un élément minimum ?
   - iv. E admet-il un élément maximum ?
   - v. E admet-il des éléments minimaux ?
   - vi. E admet-il des éléments maximaux ?
   - vii. $V = \{2, 4\}$, donner trois majorants de V dans E.
   - viii. $T = \{8, 10\}$ admet-il des majorants dans E ? Donner des majorants de T dans $\mathbb{N}^*$.
   - ix. Donner des minorants de T dans E.
   - x. Donner tous les minorants de T dans $\mathbb{N}^*$.
   - xi. $U = \{4, 6, 8\}$ admet-il une borne supérieure dans E ?
   - xii. $U = \{4, 6, 8\}$ admet-il une borne supérieure dans $\mathbb{N}^*$?

xiii. $U = \{4, 6, 8\}$ admet-il une borne inférieure dans E ?

xiv. $U = \{4, 6, 8\}$ admet-il une borne inférieure dans $\mathbb{N}^*$?

Exercice 3 - 27

$(E, \leqslant)$ est un ensemble ordonné. Représenter les différents diagrammes de Hasse pour un ensemble E de trois éléments. On précisera, pour chaque diagramme si E est totalement ordonné. Même question pour un ensemble de 4 éléments.

Exercice 3 - 28

$\mathbb{R}$ est muni de sa relation d'ordre habituelle et $\mathbb{R}^2$ est ordonné comme produit direct des ensembles ordonnés $(\mathbb{R}, \leqslant)$ et $(\mathbb{R}, \leqslant)$. Le plan P est rapporté à un repère orthonormé $\left(O, \vec{i}, \vec{j}\right)$ et on pourra confondre tout point M du plan, qui a pour coordonnées x et y dans ce repère, avec le couple (x, y) de $\mathbb{R}^2$. Cela permet d'ordonner P par la relation :

$$M_1 \preceq M_2 \Leftrightarrow (x_1, y_1) \leqslant (x_2, y_2)$$

$\mathcal{C}$ désigne le cercle de centre O et de rayon 1, $\mathcal{D}$ désigne le disque de centre O et de rayon 1.

1. Donner 5 majorants et 5 minorants de $\mathcal{C}$ et de $\mathcal{D}$.
2. $\mathcal{C}$ et $\mathcal{D}$ admettent-ils une borne sup ? une borne inf ? un plus petit élément ? un plus grand élément ?
3. On note $\mathcal{D}^+$ l'ensemble des points de $\mathcal{D}$ qui ont leurs deux coordonnées positives ou nulles, $\mathcal{D}^+$ admet-il un plus petit élément ?

Exercice 3 - 29

Vérifier que $(\mathcal{P}(E), \subseteq)$ est bien un ensemble ordonné. Soit $\{A, B\}$ une partie de $\mathcal{P}(E)$, déterminer sa borne sup et sa borne inf dans $\mathcal{P}(E)$.

Exercice 3 - 30

$\mathbb{N}_{10}$ est ordonné par la relation de divisibilité (rappel : $\mathbb{N}_{10} = \{1, 2, \cdots, 10\}$).

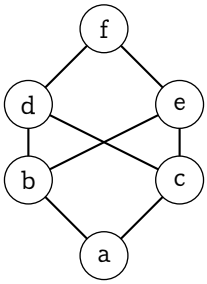
1. Tracer le diagramme sagittal de cette relation puis son diagramme de Hasse.
2. $\mathbb{N}_{10}$ admet-il un plus petit élément, un plus grand élément ?
3. étudier les éléments remarquables (plus petit élément, majorant, borne sup, $\cdots$) des parties de $\mathbb{N}_{10}$ suivantes :
   i. $A = \{1, 3, 6\}$ **ii.** $B = \{2, 3\}$ **iii.** $C = \{2, 3, 4\}$ **iv.** $D = \{2, 4\}$

Exercice 3 - 31

Soit $\mathcal{S}$ une relation réflexive et symétrique définie sur un ensemble A.
Un chemin est une suite finie a_1,...,a_n d'éléments de A telle que pour chaque $i \in [\![1, n-1]\!]$, on a $a_i \mathcal{S} a_{i+1}$.
On dit que le chemin joint a_1 et a_n.
On suppose que pour tout couple (a, b) de A^2, il existe un unique chemin joignant a à b.
Soit r un élément de A fixé.
On définit la relation $\mathcal{R}$ sur A par : $a \mathcal{R} b$ si, et seulement si, a est un des éléments du chemin joignant r à b.
Montrez que $\mathcal{R}$ est une relation d'ordre, que pour tout a dans A, on a $r \leqslant a$.
Soit $X_a = \{x \in A \mid x \mathcal{R} a\}$. Montrez que X_a est fini et totalement ordonné par $\mathcal{R}$.
Une structure $(A, \mathcal{R}, r)$ est appelée un **arbre**...

Exercice 3 - 32

Un **treillis** (en anglais *lattice*) est une poset dans lequel chaque couple d'éléments admet une borne supérieure et une borne inférieure.
Est-ce que le diagramme suivant correspond à un treillis ?



CHAPITRE 4

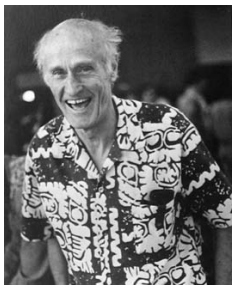
Programmation fonctionnelle et Haskell for dummies

Haskell Brooks Curry (1900-1982) est un mathématicien américain dont l'influence en informatique a été telle que trois langages de programmation portent son nom : Haskell, Brooks et Curry...
Il a fait partie de cette génération qui a lancé les fondements de la logique et de l'informatique : Alonzo Church, Alan Turing, Steven Kleene.
Ses travaux sur la logique combinatoire sont les fondements du langage de programmation fonctionnel Haskell créé dans les années 1990 et qui nous servira de support à nos travaux informatiques qui vous permettront tout au long des deux années à venir de vous familiariser avec la programmation fonctionnelle dont le rôle est de plus en plus important dans le monde informatique.

1 Rapide introduction

Alan Turing

Les langages « à la Von Neumann » sont basés sur les concepts de variable et d'instruction. Ces concepts sont assez intuitifs car ils correspondent à de nombreuses situations « de la vie de tous les jours » : nous agissons sur et en fonction de notre environnement immédiat. Ce modèle dérive des travaux de Turing et de sa machine universelle dans les années 1930. D'autres mathématiciens ont proposés des modèles différents à la même époque : le λ-calcul d'Alonzo Church, les fonctions récursives de Stephen Kleene, la logique combinatoire de Haskell Curry. Ces trois derniers modèles n'utilisent pas le concept de machine et de mémoire de Turing mais uniquement de valeur, de calcul d'autres valeurs à partir de fonctions qui sont elles-mêmes des valeurs.



Stephen Kleene

Mmmm..., il nous faudrait beaucoup de temps pour rentrer dans les détails. Nous ne disposons que de quelques minutes alors donnons plutôt un exemple.
Nous voulons calculer la somme des entiers de a jusqu'à b.
En C en programmant en impératif :

C

```c
#include <stdio.h>

int somme(int,int);

void main(void)
{ int s,a,b;

  printf("Entrez la valeur du plus petit entier: ");
  scanf("%d" , &a);
  printf("Entrez la valeur du plus grand entier: ");
  scanf("%d" , &b);
  s = somme(a,b);
  printf("La somme des entiers de  %d à %d est %d \n", a,b,s);
}

int somme(int a, int b)
{ int tmp; int som;
  tmp = a;
  som = a;
  while (tmp != b)
   {
     tmp = tmp + 1;
     som = som + tmp;
   }
  return som;
}
```

Puis on compile et on exécute :

Shell

```
$ gcc -o somme poly_haskell.c
$ ./somme
Entrez la valeur du plus petit entier: 1
Entrez la valeur du plus grand entier: 10
La somme des entiers de  1 à 10 est 55
```

Pour comprendre ce qui se passe ici, on est obligé d'introduire des suite (t_i) et (s_i) des valeurs prises par les variables **tmp** et **som** avec t_i la valeur prise par la variable **tmp** au bout de la i^e itération si celle-ci a eu lieu.
Ensuite il reste à démontrer par récurrence que la fonction calcule bien la somme de a et b.

On travaille bien sur le temps (nombre d'itérations) et la mémoire (contenu des mémoires référencées par **tmp** et **som**).
En programmation fonctionnelle, on ne peut pas travailler sur des variables. On travaille sur des fonctions. En Haskell, par exemple, cela donne :

Haskell
```
somme a b = foldl (+) 0 [a..b]
```

et dans l'interpréteur :

Haskell
```
*Main> somme 1 10
55
```

Quelle est donc cette fonction **foldl** ? Nous allons le voir bientôt. Elle permet de « plier » la liste des entiers de a à b et à chaque « pliage » d'additionner à un accumulateur l'entier plié en partant d'un accumulateur nul.
En fait, comme on a souvent besoin d'additionner les éléments d'une collection, il existe une fonction toute faite pour ça : **sum** :

Haskell
```
*Main> sum [1..10]
55
```

On peut prendre des fonctions en argument. Par exemple, pour calculer la somme des images par une fonction f des entiers de a à b :

Haskell
```
somme_fonction f a b = foldl (\accu x -> accu + f(x)) 0 [a..b]
```

Alors, pour avoir la somme des carrés des entiers de 1 à 10 :

Haskell
```
*Main> somme_fonction (\x -> x^2) 1 10
385
```

Supposons qu'on veuille à présent faire la somme des carrés impairs inférieurs à 1000.
Écrivez le programme en C...
Voici ce que cela peut donner en Haskell :

Haskell
```
sum (takeWhile (<1000) (filter odd (map (^2) [1..])))
```

Quand on a fait un peu d'anglais de spécialité, ça se comprend...
Haskell Curry a aussi donné son nom au fait de n'utiliser que des fonctions qui n'ont qu'un seul argument. Une **fonction curryfiée** est une fonction de plusieurs variables transformée en une fonction d'une seule variable renvoyant une fonction ayant une variable de moins...
Par exemple, créons une fonction curryfiée qui semble renvoyer la somme de deux entiers :

Haskell
```
*Main> let plus x y = x + y
```

Notons $\mathcal{F}(D, A)$ l'ensemble des fonctions de D vers A.
Alors `plus` $\in \mathcal{F}\big(\mathbb{N}, \mathcal{F}(\mathbb{N}, \mathbb{N})\big)$.
En effet, **plus x** est elle-même une fonction de $\mathbb{N}$ dans $\mathbb{N}$ qui à un entier n associe l'entier $n + x$. Par exemple, avec 3 :

Haskell
```
*Main> let plus_trois = plus 3
*Main> plus_trois 5
8
*Main> :t plus_trois
plus_trois :: Integer -> Integer
```

Cette possibilité existe aussi dans des langages non spécifiquement fonctionnels comme Python qui dispose de la fonction **`lambda`** qui renvoie une fonction « anonyme » mais on y a accès de manière moins naturelle :

Python

```
>>> def plus(x): return lambda y: x+y
...
>>> plus(2)(3)
5
>>> def plus_trois(t): return plus(3)(t)
...
>>> plus_trois(5)
8
```

À retenir

Les idées principales de la programmation fonctionnelle sont :

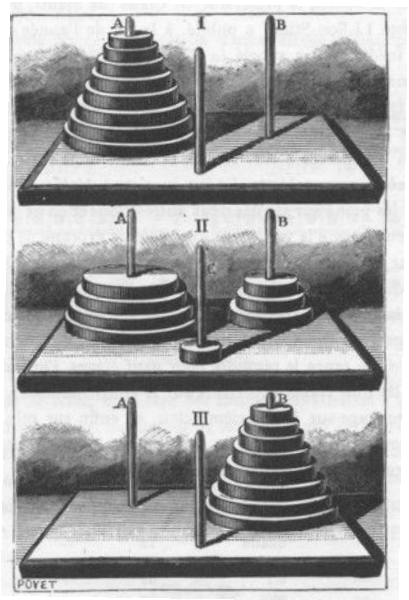
- utiliser des fonctions qui sont comme protégées dans une capsule inviolable : elles feront toujours ce que vous avez prévu qu'elles fassent, maintenant ou dans mille ans, ici ou sur Mars, quelque soit l'état du système de fichier ou de la base de données. C'est ce que l'on appelle des « fonctions pures », très proches de celles qu'on connaît en mathématiques ;
- décomposer un problème complexe en petits problèmes simples ;
- abstraire au maximum pour pouvoir utiliser les fonctions dans de nombreux contextes différents donc ne pas dépendre de l'environnement pour comprendre le code qui devient alors beaucoup plus lisible et contrôlable ;
- rendre possible la preuve a priori du programme pour ne plus avoir de bug...

Pour vous convaincre du rôle essentiel de la programmation fonctionnelle, lisez l'article révolutionnaire de John Backus [1978], le père du pourtant (ou justement...) très impératif Fortran. Enfin, il faudra vous plonger dans le merveilleux tutoriel Haskell de Lipovača et Robert [2013] qui est drôle et clair et commence par ces lignes :

> *J'ai échoué dans mon apprentissage de Haskell à peu près deux fois avant d'arriver à le saisir, car cela me semblait trop bizarre et je ne comprenais pas. Mais alors, j'ai eu le déclic, et après avoir passé cet obstacle initial, le reste est venu plutôt facilement. Ce que j'essaie de vous dire : Haskell est génial, et si vous êtes intéressés par la programmation, vous devriez l'apprendre même si cela semble bizarre au début. Apprendre Haskell est très similaire à l'apprentissage de la programmation la première fois - c'est fun ! Ça vous force à penser différemment.*

2 Un jeu

Mathémator : Cet été, lors de la visite d'un musée, j'ai, pour une fois, impressionné mes enfants en résolvant devant eux très rapidement le problème suivant :



Ce casse-tête a été posé par le mathématicien français Édouard Lucas en 1883.
Le jeu consiste en une plaquette de bois où sont plantés trois piquets. Au début du jeu, 4 disques de diamètres croissant de bas en haut sont placés sur le piquet de gauche. Le but du jeu est de mettre ces disques dans le même ordre sur le piquet de droite en respectant les règles suivantes :

— on ne déplace qu'un disque à la fois ;

— on ne peut poser un disque que sur un disque de diamètre supérieur.

Je vous propose donc de jouer vous aussi avec moi...

Hihutix (à part): *pauvres enfants, passer des vacances avec un prof de maths, vois le cauchemar* **(tout haut)** Oh, comme je suis content !

Mathémator : Essayez d'abord avec 2 puis 3 disques.

Hihutix: Bon, avec deux, je mets le disque là puis l'autre là-bas et ensuite celui-ci ici. Ça marche. Avec trois, je bouge celui-là, et puis l'autre, celui-ci ici, l'autre là, puis celui-là là et l'autre là puis le dernier ici. Ah, c'est pas le bon piquet mais ça marche à peu près. Facile votre jeu.

Mathémator : Alors essayez avec quatre, cinq, autant de disques que vous voulez.

Quelques minutes plus tard

Hihutix: Pénible votre jeu !

Mathémator : Les choses se compliquent. Il est temps de parler de la légende rapportée par Lucas dans ses « *récréations mathématiques* » :

> *N. Claus de Siam a vu, dans ses voyages pour la publication des écrits de l'illustre Fer-Fer-Tam-Tam, dans le grand temple de Bénarès, au-dessous du dôme qui marque le centre du monde, trois aiguilles de diamant, plantées dans une dalle d'airain, hautes d'une coudée et grosses comme le corps d'une abeille. Sur une de ces aiguilles, Dieu enfila au commencement des siècles, 64 disques d'or pur, le plus large reposant sur l'airain, et les autres, de plus en plus étroits, superposés jusqu'au sommet. C'est la tour sacrée du Brahmâ. Nuit et jour, les prêtres se succèdent sur les marches de l'autel, occupés à transporter la tour de la première aiguille sur la troisième, sans s'écarter des règles fixes que nous venons d'indiquer, et qui ont été imposées par Brahma. Quand tout sera fini, la tour et les brahmes tomberont, et ce sera la fin des mondes !*

C'est un problème posé par un mathématicien mais nous allons l'étudier en informathicien. Nous voudrions répondre à plusieurs questions :

— peut-on résoudre le problème des 64 disques ?

— si oui, en combien de mouvements ?

— peut-on trouver une tactique la plus efficace possible ?

— est-ce que la fin du monde est pour bientôt ?

Au lieu d'étudier des cas séparés, nous allons généraliser un peu et considérer une tour avec n disques, n étant un entier naturel non nul.

Hihutix: Cela va compliquer les choses. Pourquoi ne pas s'en tenir à un cas précis ? C'est une perversion de mathématicien.

Mathémator : Détrompez-vous ! Au lieu d'étudier un à un des cas isolés, il est bon pour un informaticien aussi de voir qu'un programme plus général pourra traiter tous les cas. Cependant, il sera utile d'étudier des cas simples d'abord pour se donner une idée.

Après une rapide exploration avec papier et crayon, il est temps pour un informathicien d'introduire des notations adéquates.

Notons M_n le nombre minimum de mouvements pour transférer n disques. Pouvez-vous donner les premières valeurs de M_n ?

Hihutix: Bon, s'il n'y a qu'un disque, on ne pourra pas faire mieux qu'un déplacement : $M_1 = 1$. Avec deux disques, hop, hop, hop : trois mouvements ; $M_2 = 3$. Pour trois, j'avais trouvé sept déplacements mais je ne suis pas sûr que ça soit la meilleure solution.

Mathémator : C'est ce que nous allons prouver mais tout d'abord, en bon mathématicien, rajoutons un cas extrême : s'il y a zéro disque, il faudra zéro mouvement.

Hihutix: C'est un peu tiré par les cheveux.

Mathémator : Mais considérer les cas les plus triviaux permet souvent de simplifier le cas général. Il est temps d'ailleurs de voir grand mais il nous faudrait une idée. Essayons de trouver des points communs aux déplacements de deux et trois disques respectivement.

Hihutix: Ben, je mets le petit sur le piquet du milieu, le grand sur celui d'arrivée puis le petit sur ce même piquet d'arrivée.

Mathémator : Pour trois c'est pareil : je mets les deux petits sur celui du milieu, le grand sur celui d'arrivée puis les deux petits par dessus le grand.

Hihutix: Vous trichez ! Vous avez bougé deux disques en même temps.

Mathémator : En fait, j'ai vu que je savais bouger deux disques d'un piquet vers un autre. Cela me demande trois étapes qui se cachent derrière le « *je mets les deux petits sur celui du milieu* ».

Cela nous donne en fait la solution : avec $n+1$ disques, je bouge les n plus petits sur le piquet du milieu (M_n mouvements), je bouge le plus grand sur le piquet d'arrivée (1 mouvement) puis je redéplace les n petits sur le grand (M_n mouvements). Ainsi :

$$M_{n+1} \leqslant 2M_n + 1$$

Hihutix: Pourquoi avoir utilisé $\leqslant$ et non pas $=$? Et puis, on ne connaît pas M_n.

Mathémator : Vous touchez du doigt deux problèmes essentiels qui vont guider bon nombre de nos raisonnements.

Pour le $\leqslant$, il est essentiel de comprendre que nous avons prouvé que $2M_n+1$ mouvements étaient **suffisants** mais nous ne savons toujours pas si ces $2M_n+1$ mouvements sont **nécessaires**. C'est très important de comprendre la différence.

Par exemple, pour acheter un chocolat chaud à la cafétéria et l'offrir à son professeur adoré, il est **suffisant** d'avoir 10 euros dans sa poche mais ce n'est pas nécessaire. Inversement, il est **nécessaire** d'avoir au moins 20 centimes mais ce n'est malheureusement pas suffisant.

Nous avons étudié cela en détail dans un précédent chapitre sur la logique.

À retenir

On considère P et Q deux propositions (des énoncés, des faits, des formules).

— Q est une **condition nécessaire** pour avoir P si, dès que P est vraie, alors nécessairement, forcément, obligatoirement, Q est vraie. On note souvent $P \to Q$.

— Q est une **condition suffisante** pour avoir P s'il suffit que Q soit vraie pour que P soit vraie. On note souvent $P \to Q$.

— Lorsque P est à la fois condition nécessaire et condition suffisante de Q, on dit que P est une **condition nécessaire et suffisante** de Q ou encore que P est vraie **si, et seulement si**, Q est vraie. On note souvent $P \leftrightarrow Q$.

Lorsque qu'une « affirmation » du type $P \leftrightarrow Q$ ou $P \leftrightarrow Q$ est vraie, on dit que c'est un **théorème**.

Nous avons développé des méthodes de preuve dans un chapitre précédent.
Revenons tout d'abord à nos disques. Peut-on faire mieux que la solution proposée ?
À un moment donné, il *faut* bien bouger le plus grand disque puisqu'il est sur le mauvais piquet. Il *faut* donc qu'il soit tout seul et il *faut* donc avoir bougé auparavant les n autres disques plus petits. Cela *nécessite au minimum* M_n mouvements. Après cela, on peut bouger le grand disque autant de fois qu'on le désire mais à un moment donné, il *faudra* le placer sur le piquet d'arrivée et déplacer les n autres disques sur le piquet d'arrivée ce qui *nécessite* à nouveau M_n mouvements.
Ainsi, il *faudra au minimum* effectuer $2M_n + 1$ mouvements.
On a donc prouvé que :

$$M_{n+1} \geqslant 2M_n + 1$$

Il est donc **nécessaire** d'effectuer $2M_n + 1$ mouvements pour déplacer les $n + 1$ disques. Or nous avons montré auparavant que ces $2M_n + 1$ mouvements étaient **suffisants**. On en déduit donc que :

$$\begin{cases} M_0 = 0 \\ M_n = 2M_{n-1} + 1 \text{ pour tout } n > 0 \end{cases}$$

Est-ce que cette formule marche bien avec nos premiers exemples ?

Hihutix: $M_1 = 2 \cdot 0 + 1 = 1$: OK. $M_2 = 2 \cdot 1 + 1 = 3$: OK. $M_3 = 2 \cdot 3 + 1 = 7$: OK.

Mathémator : Ça ne prouve pas grand chose mais ça nous rassure. De toute façon, nous avons prouvé que cette formule est bonne. Que vaut M_{64} ?

Hihutix: Ben $M_{64} = 2M_{63} + 1$. Le problème, c'est qu'on ne connaît pas M_{63}.

Mathémator : Et oui : on définit la *suite* **M** à partir d'elle-même, ce qui n'est pas très pratique. On dit qu'on a définit la suite par **récurrence**. Vous verrez cette année en informatique que ce mode de calcul n'est pas réservé aux suites que vous avez peut-être étudiées au lycée. Par exemple, en langage formel, on travaille avec les lettres d'un alphabet. On peut alors définir un mot comme étant soit le vide, soit une lettre suivi d'un mot.

Hihutix: C'est un peu idiot : en fait, vous me dites que pour savoir si un truc est un mot, faut déjà savoir si c'est un mot.

Mathémator : Vous n'êtes pas assez précis. Prenons notre alphabet et la chaîne de caractères « math ». C'est le vide suivi de la chaîne « math » qui est la lettre « m » suivie de la chaîne « ath » qui est la lettre « a » suivie de la chaîne « th » qui est la lettre « t » suivie de la lettre « h » qui est un mot puisque c'est une lettre. En remontant, on obtient que « th » est un mot puis que « ath » aussi et enfin « math » est donc un mot.

Hihutix (à part): *Ben on le savait avant. Si c'est ça c'qu'on apprend à l'IUT d'info, il est peut-être encore temps de m'inscrire en GEA.* **(tout haut)** Fabuleux ! Math est un mot ! J'aurais pas cru, comme ça, a priori.

Mathémator : Ah, comme quoi ce que nous faisons est utile.

Hihutix (à part): *Complètement gâteux le gars* **(tout haut)** Je n'en doute pas.

Mathémator : Et la fin du monde dans tout ça ? Pour connaître M_{64}, il semblerait qu'il faille connaître tous les M_k précédents.

Voyons voir, calculez jusqu'à M_6.

Hihutix: Si vous voulez :

— $M_3 = 2 \cdot 3 + 1 = 7$;

— $M_4 = 2 \cdot 7 + 1 = 15$;

— $M_5 = 2 \cdot 15 + 1 = 31$;

— $M_6 = 2 \cdot 31 + 1 = 63$;

Ouais, bof. Je ne vois pas trop où ça nous mène.

Mathémator : « *Ils ont des yeux et ils ne voient pas* ». Et pourtant, quand j'étais jeune, un enfant de six ans aurait reconnu la suite :

$$2^3 - 1,\ 2^4 - 1,\ 2^5 - 1,\ 2^6 - 1, \ldots$$

Hihutix (à part): *Ben tiens ! Les puissances à 6 ans et pourquoi pas les logarithmes à 7.*

Mathémator : Vous dites ?

Hihutix: Je réfléchissais.

Mathémator : Ah. Alors, vous avez sûrement envie de dire que tout prête à croire que :

$$M_n = 2^n - 1, \text{ pour tout entier strictement positif } n$$

Hihutix: Ben oui, c'est vrai au début donc y a pas de raison pour que ça s'arrête.

Mathémator : Encore faudrait-il le prouver. Par exemple, est-ce que la proposition suivante est un théorème :

Les entiers impairs supérieurs à 3 sont tous des nombres premiers.

Hihutix: 3 est premier, 5 est premier, 7 est premier...

Mathémator : ...donc c'est vrai !

Hihutix: Ben non. 9 est impair mais n'est pas premier...Ouais, OK, j'ai compris.

Mathémator : « Vrai pour les premiers, vrai pour tous » n'est pas un théorème ! Vous retiendrez également que...

À retenir

Pour prouver qu'une proposition n'est pas un théorème, il suffit d'exhiber un **contre-exemple**.

Nous en avons déjà parlé. Revenons à notre récurrence ou induction.

À retenir

Pour prouver qu'une propriété $\mathcal{P}_n$ dépendant uniquement d'un paramètre n est vraie pour tout $n \geqslant n_0$, il faut vérifier que :

— $\mathcal{P}_{n_0}$ est vraie (on parle parfois d'initialisation) ;

— pour tout $n \geqslant n_0$, $\mathcal{P}_n \to \mathcal{P}_{n+1}$ (on parle parfois d'hérédité).

Cela peut par exemple se prouver en *raisonnant par l'absurde* comme nous l'avons déjà vu.
Ici, pour tout entier naturel non nul n, notons $\mathcal{P}_n$ la proposition : $M_n = 2^n - 1$.
Cette proposition est vraie pour $n = 1$ puisque $M_1 = 1 = 2^1 - 1$.
Il existe donc au moins un entier naturel non nul k tel que $\mathcal{P}_k$ soit vraie.
Alors $M_{k+1} = 2M_k + 1 = 2(2^k - 1) + 1 = 2 \cdot 2^k - 2 + 1 = 2^{k+1} - 1$.
Ainsi $\mathcal{P}_k \to \mathcal{P}_{k+1}$ pour tout entier naturel non nul k.
Le tour est joué. On peut donc affirmer que $M_n = 2^n - 1$ pour tout entier naturel non nul n.

Hihutix: Donc $M_{64} = 2^{64} - 1$. Je prends ma calculatrice...$M_{64} = 18\,446\,744\,073\,709\,551\,615$. Ça ne m'arrange pas tellement pour connaître la date de la fin du monde.

Mathémator : C'est pourtant simple. Disons que les moines sont très bien entraînés et se relaient efficacement. Ils déplacent alors un disque par seconde. Cela nous donne une durée de jeu de...

Hihutix: Ah, ça je sais : $18\,446\,744\,073\,709\,551\,615/3600/24/365,25/100$ ce qui fait en gros 5.8 milliards de siècles...

Mathémator : Et comme l'Univers a grosso modo 140 millions de siècles d'existence, cela nous laisse de la marge.

Hihutix: Ouf ! Mais cela ne nous indique pas comment déplacer les huit disques du jeu initial. On sait juste qu'en étant aussi efficaces que les moines, cela nous prendra $M_8 = 2^8 - 1 = 255$ secondes.

Mathémator : Je vous laisse donc cinq minutes pour le faire.

Cinq minutes plus tard

Hihutix: Il faut se rendre à l'évidence : je ne suis pas fait pour être moine. Faut le faire pour chaque n jusqu'à 8 en fait et noter comment on a fait. Pfff... C'est pénible.

Mathémator : Et quand un informathicien a bien réfléchi sur le papier, il peut laisser faire le sale boulot à la machine. On a en effet une méthode qu'on sait être correcte. Mais il est très difficile de l'appliquer pour un grand n donné. On sait que ça marche mais on ne sait pas vraiment comment ça marche. Et c'est là qu'une programmation bien pensée devient merveilleusement efficace. Si on a à notre disposition un langage sachant traiter la récursion, il va suffire de lui donner un minimum d'instruction.
Rappelons ici notre méthode : avec $n+1$ disques, je bouge les n plus petits sur le piquet du milieu (M_n mouvements), je bouge le plus grand sur le piquet d'arrivée (1 mouvement) puis je redéplace les n petits sur le grand (M_n mouvements).
Illustrons notre propos avec le langage Haskell qui aime les récursions.
L'opérateur **++** concatène des listes donc aussi des chaînes qui sont des listes de caractères :

Haskell

```
*Main> :t (++)
(++) :: [a] -> [a] -> [a]
*Main> [1,2,3] ++ [4]
[1,2,3,4]
*Main> "Raahhh" ++ " lovely"
"Raahhh lovely"
```

On traite les deux cas, un ou n disques :

Haskell

```
hanoi ::  String -> String -> String -> Int -> String
-- On nomme les trois piquets sous forme de chaînes.
-- On obtient la liste des mouvements sous forme d'une chaîne.
-- On déplace n disques de dep vers but en se servant de piv comme pivot.
hanoi dep but piv 1 =
  -- 1 disque : renvoie la chaîne décrivant le mvt de dep vers but
  "De " ++ dep ++ " vers " ++ but ++ "\n"
hanoi dep but piv n =
  -- n disques :
  (hanoi dep piv but (n - 1)) ++ --  on bouge n-1 disques de dep vers piv
  (hanoi dep but piv 1) ++ -- on bouge le gros disque de dep vers but
  (hanoi piv but dep (n - 1)) -- on bouge le n-1 disques du piv vers but
```

On affiche joliment le résultat :

Haskell

```
-- on affiche le résultat
tours_hanoi :: Int -> IO ()
tours_hanoi n = putStrLn ( hanoi "Tige 1" "Tige 2" "Tige 3" n)
```

Ce qui donne pour trois disques :

Haskell

```
*Main> tours_hanoi 3
De Tige 1 vers Tige 2
De Tige 1 vers Tige 3
De Tige 2 vers Tige 3
De Tige 1 vers Tige 2
De Tige 3 vers Tige 1
```

```
De Tige 3 vers Tige 2
De Tige 1 vers Tige 2
```

et si l'on veut obtenir le nombre de mouvements pour déplacer de 1 à 20 disques, on change un peu le code :

Haskell

```haskell
nb_hanoi :: Int -> Int
nb_hanoi 1 = 1
nb_hanoi n = 2 * (nb_hanoi (n-1)) + 1

*Main> [nb_hanoi k | k <- [1..20]]
[1,3,7,15,31,63,127,255,511,1023,2047,4095,8191,16383,32767,65535,131071,262143,524287,1048575]
```

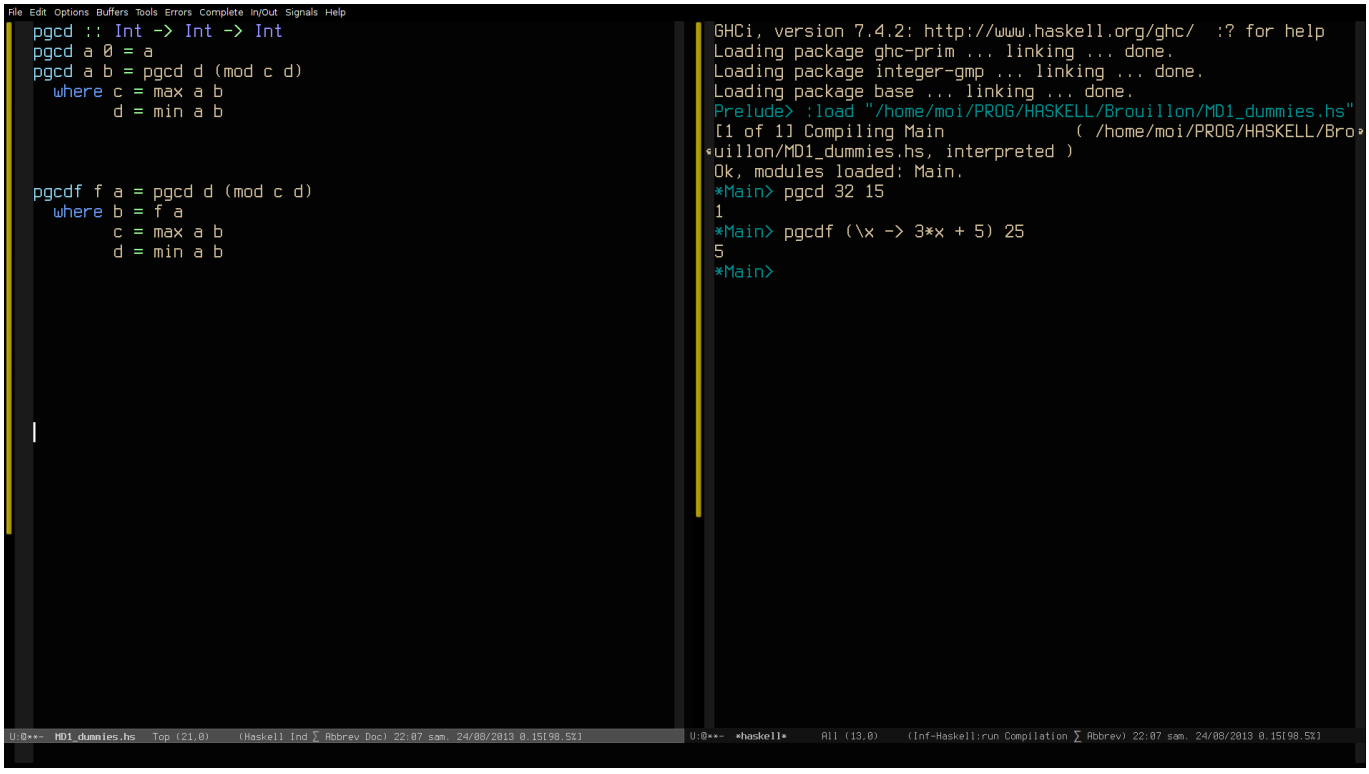
Hihutix: Waouh ! Ça je préfère.

Mathémator : Je n'en doute pas, mais pour y arriver, il faudra passer par le stade « réflexion mathématiquement assistée avec papier-crayon »

3 Une programmation désaffectée

3.1 Environnement

Nous travaillerons avec emacs en mode haskell et nous partagerons notre écran en deux tampons, l'un contenant le code et l'autre l'interpréteur du compilateur GHC :



Pour cela, il faut ouvrir un petit shell depuis emacs en tapant **`M-x shell`**, créer un répertoire

Shell

```
$ mkdir ~/Maths/
$ mkdir ~/Maths/INFO1/
$ mkdir ~/Maths/INFO1/MD/
```

pour y placer votre Fichier d'extension **`hs`** : **`C-x C-f ~/Maths/INFO1/MD/dummy.hs`**
et lancer **GHCI** avec **`C-c C-b`**.
Pour les principaux raccourcis clavier d'emacs, voir l'annexe A page 120.

3 2 Les fonctions

L'outil de base est donc la fonction *pure*, au sens mathématique du terme, contrairement aux « fonctions » du paradigme impératif : c'est une relation entre deux types, qui à chaque élément du type de départ (domaine) fait correspondre un et un seul élément du type d'arrivée (image ou codomaine).
Il y a trois étapes à suivre pour construire une fonction :

SPÉCIFIER la fonction : c'est-à-dire parfois (on peut effectivement s'en passer au besoin) la nommer, donner son *type*, i.e. son domaine et son codomaine : on parle de *signature de type* et on explique ce qu'elle fait. Par exemple pour le double d'un entier sur Haskell :

Haskell
```
double :: Int -> Int
-- calcule le double d'un entier : le résultat est un entier
```

Sur Haskell, donner la signature est optionnel car le compilateur l'*infère* mais c'est une bonne pratique de l'annoncer à l'avance pour plus de sécurité. Si le type inféré par le compilateur est différent de celui spécifié, c'est qu'il faut revoir sa copie...Cela rend de plus le code plus lisible.

On peut obtenir le type d'une expression avec la commande `:t` :

Haskell
```
*Main> :t double
double :: Int -> Int
```

RÉALISER la fonction c'est associer à la fonction spécifiée une expression. Dans notre exemple, il s'agit de multiplier l'argument de la fonction par 2. Pour cela, on utilise des *paramètres formels* qui font *abstraction* de la valeur particulière du nombre dont on veut connaître le double. Ici :

Haskell
```
double n = 2 * n
```

UTILISER la fonction dans une expression en lui donnant des *paramètres effectifs* (arguments), c'est-à-dire liés à l'application particulière de la fonction :

Haskell
```
*Main> (double 3) + (double 7)
20
```

Il faut bien noter que les étapes de réalisation et d'utilisation sont en quelque sorte indépendantes, contrairement au paradigme impératif.
La réalisation *fait abstraction* de l'utilisation : l'expression fournie réalisera la spécification quelque soit le contexte.
L'utilisation *fait abstraction* de la réalisation : si on suit la spécification, on aura ce qu'on veut dans tous les cas.

3 3 Un exemple

Nous allons reprendre un exemple donné pour découvrir Scheme dans Scholl et coll. [1993].

Exemple

Décrire une fonction qui étant donnés quatre flottants leur associe la moyenne des deux nombres parmi les quatre qui ne sont ni le plus grand ni le plus petit. Par exemple, la moyenne olympique de 10, 8, 12 et 14 est 11 et celle de 12, 12, 12 et 12 est 12.

spécification On donne la signature de la fonction et on dit rapidement ce qu'elle doit faire :

Haskell
```
moyenne_olympique4 :: Float -> Float -> Float -> Float -> Float
-- renvoie la moyenne olympique de 4 flottants sous forme d'un flottant
```

Les types (ici **Float**) commencent par une majuscule.

Pour nommer les fonctions, même si c'est tentant, évitez des noms comme « bourre_pif_44 » ou même « f » car il vaut mieux avoir des noms les plus significatifs possibles pour rendre l'utilisation des fonctions plus limpide.

réalisation Il existe bien sûr de nombreuses pistes pour réaliser cette spécification. Proposez-en au moins trois. Nous allons par exemple additionner les quatre nombres, retirer le plus grand et le plus petit puis diviser par 2.

On voit comment additionner quatre nombres mais comment obtenir le plus grand et le plus petit des quatre ?

On dispose sur Haskell des fonctions **min** et **max**. La signature de **min** est :

Haskell
```
*Main> :t min
min :: Ord a => a -> a -> a
```

Ord est la *classe de type* des types ordonnés. Comme il y en a plusieurs (les « nombres », les caractères, les chaînes de caractères, les booléens, et tous ceux que nous créerons...), on les regroupe sous la *variable de type* **a** qui peut être en fait n'importe quel type.

Recherche Est-ce que **Ord** peut s'appliquer à n'importe quel type ?

Haskell
```
*Main> "Argh" > "Ah!"
True
*Main> compare "Argh" "Ah!"
GT
*Main> compare True False
GT
*Main> compare 3 4.0
LT
*Main> compare 3 "quatre"

<interactive>:68:9:
    No instance for (Num [Char])
      arising from the literal '3'
    Possible fix: add an instance declaration for (Num [Char])
    In the first argument of 'compare', namely '3'
    In the expression: compare 3 "quatre"
    In an equation for 'it': it = compare 3 "quatre"
```

La fonction **min** est donc *polymorphe* :

Haskell
```
*Main> min 'a' 'b'
'a'
*Main> min 5 3
3
*Main> min "Tralala" "Pouet Pouet"
"Pouet Pouet"
*Main> min True False
False
```

Bon, mais ça ne permet de trouver que le plus petit entre DEUX nombres et nous voudrions le minimum de QUATRE nombres...

Que fait-on naturellement quand on a une liste de nombres ? On les compare deux par deux : les deux premiers, hop avec le troisième et hop avec le quatrième.

Bref, quelque chose comme ça : **min a (min b (min c d))**.

Bon, ben, y a plus qu'à l'écrire comme on l'a pensé :

Haskell
```
moyenne_olympique4 a b c d =
  (s - mini - maxi) / 2 -- l'expression correspondant à la méthode choisie
  where s = a + b + c + d -- la somme des 4 nombres
        mini = min a (min  b (min c d) ) -- le plus petit des 4
        maxi = max a (max  b (max c d) ) -- le plus grand des 4
```

utilisation Reprenons un des exemples proposés au départ :

Haskell
```
*Main> moyenne_olympique4 10 8 12 14
11.0
```

Vous aurez remarqué que 10 par exemple est un **`Int`** et pourtant il est traité comme un **`Float`**...

En fait, 10 n'est pas un **`Int`**, c'est un **`Num`** qui regroupe les différents types numériques : **`Int`**, **`integer`**, **`Float`**, **`Double`**.

Haskell
```
*Main> :t 10
10 :: Num a => a
```

4 Polymorphisme - abstraction - récursion - listes

Nous avons donc mis au point une fonction qui calcule la moyenne olympique de quatre nombres mais s'il y en a plus ? On veut donc abstraire la taille de l'argument de la fonction.

Exemple

Décrire une fonction qui étant donnée une liste d'au moins quatre nombres leur associe la moyenne des nombres parmi ceux de la liste qui ne sont ni le plus grand ni le plus petit. Par exemple, la moyenne olympique de 15, 7, 10, 8, 12 et 14 est 11.

Reprenons notre minimum de quatre nombres : **`min a (min b (min c d) )`**.
Bon, pour cinq : **`min a (min b (min c (min d e)))`**.
Pour six....pour 20...ça doit marcher pareillement...On voit le principe.
La signature sera :

Haskell
```
min_liste :: (Ord a) => [a] -> a
-- renvoie le mini d'une liste d'éléments ordonnables
```

et pour le maximum aussi :

Haskell
```
max_liste :: (Ord a) => [a] -> a
-- renvoie le maxi d'une liste d'éléments ordonnables
```

Alors on pourrait regrouper ça en une seule fonction qui prendrait le type d'extremum en argument, quelque chose comme ça :

Haskell
```
extr_liste :: (Ord a) => (a -> a -> a) -> [a] -> a
-- renvoie le mini ou le maxi (on choisit en mettant la nature de l'extremum en
    paramètre) d'une liste d'éléments ordonnables
```

Ouh la la...ralentissons un peu...

4 1 Récursion, induction, récurrence

GNU : Gnu's Not Unix

Définition : si vous ne comprenez toujours pas le sens du mot « récursion », relisez cette phrase.

...des mots qui tournent autour du même thème mais qu'il faut savoir distinguer.
L'*induction* consiste à conclure à partir d'observations préalables : cela correspond à la démarche expérimentale classique. Sans précautions, cela peut conduire à certaines contre-vérités. Par exemple 3, 5 et 7 sont premiers donc on pourrait en déduire, par induction, que tous les nombres impairs à partir de 3 sont premiers et pourtant...
L'*induction mathématique* ou *raisonnement par récurrence* corrige ce défaut. On part toujours d'un résultat observé mais on *démontre* qu'il est vrai pour tous les éléments d'un ensemble donné, éventuellement infini. C'est l'induction expérimentale corrigée par la rigueur du raisonnement mathématique.
En informatique, une *fonction récursive* (ou un *type récursif*) est une fonction (ou un type) qui fait référence à elle(lui)-même dans sa définition. Ce mécanisme est très puissant et permet de condenser l'écriture d'un programme.
Nous avons déjà parlé de l'induction mathématique. Nous allons donc nous occuper dans un premier temps des fonctions puis des types récursifs.

4 1 1 Fonction récursive

Nous avons parlé des tours de Hanoï. Voyons un exemple plus simple, la factorielle :

$$n! = 1 \times 2 \times 3 \times \cdots \times n$$

Les trois petits points, c'est de la récursion cachée. Débarrassons-nous en :

$$n! = \begin{cases} 1 \text{ si n=0} \\ n \times (n-1)! \text{ si } n \geqslant 1 \end{cases}$$

ou encore `fac(n): si n = 0 alors 1 sinon n * (fac (n - 1))`.

À retenir

L'idée est de résoudre un problème portant sur une donnée d'une certaine « taille » et de tenter de le décomposer pour se ramener au *même problème* mais sur une donnée de *taille plus petite* et de savoir le résoudre directement dans un cas basique le plus petit possible.

Pour la factorielle, la valeur de la fonction en n se calcule au moyen de la valeur de la fonction en $n-1$. De plus, on sait calculer la factorielle pour $n = 0$ ce qui va permettre d'arrêter le processus.

Recherche

Pour la factorielle, on prouve par récurrence que `fac(n)` calcule bien $n!$ et que le nombre d'appels de la fonction `fac` engendrés par le calcul de `fac(n)` est égal à $n - 1$: faites-le !

Attention ! Toute formule n'entraîne pas un calcul fini. Par exemple, on aurait pu définir `fac` de la sorte :
`fac(n): si n = 0 alors 1 sinon (fac (n + 1)) / (n + 1)`
Quel est le problème ?
Voici une première manière d'implémenter la factorielle avec Haskell :

Haskell

```haskell
fac1 :: Integer -> Integer
fac1 0 = 1
fac1 n = n * (fac (n-1))
```

Vous noterez l'emploi d'**`Integer`** au lieu de **`Int`** car ce dernier correspond à un entier machine donc, comme vous l'avez vu en archi, il est borné.
Sur une machine 64 bits, on prend 1 bit pour le signe donc on dispose de 63 bits.
On peut donc coder des entiers entre -2^{63} et $2^{63} - 1$. On vérifie avec la fonction constante **`maxBound`** :

Haskell
```
*Main> maxBound :: Int
9223372036854775807
*Main> 2^63 - 1
9223372036854775807
*Main> minBound :: Int
-9223372036854775808
*Main> - 2^63
-9223372036854775808
```

Le type **Integer** correspond lui à des entiers de longueur quelconque. Alors pourquoi utiliser **Int** ? Parce que c'est beaucoup plus rapide et moins gourmand en mémoire. On n'utilisera **Integer** qu'en cas de besoin comme ici pour les factorielles qui font exploser la taille des nombres :

Haskell
```
*Main> fac1 100
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000
```

Il y a bien d'autres façons d'écrire la factorielle en Haskell...

— En recréant une boucle :

Haskell
```
fac2 n = fac_boucle n 1
  where
    fac_boucle 0 accu = accu
    fac_boucle k accu = fac_boucle (k - 1) (k * accu)
```

Il y a une boucle là-dedans ?

— En utilisant un *pliage* comme **foldl** qui plie une liste par la gauche en lui appliquant une fonction et en commençant par une valeur donnée.

Haskell
```
fac3 :: Integer -> Integer
fac3 n = foldl (*) 1 [1..n]
```

— En utilisant **product** qui calcule le produit des éléments d'une liste...

```
fac4 :: Integer -> Integer
fac4 n = product [1..n]
```

Il est difficile au début de comprendre les dessous de l'évaluation en Haskell. Nous n'en parlerons pas (tout de suite).

Disons que l'emploi de fonctions que nous allons étudier bientôt comme **fold**, **map** sera notre priorité. Ensuite, on essaiera une récursion.

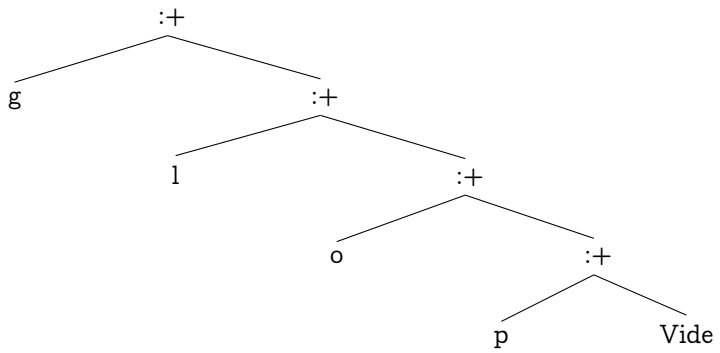
4 1 2 Définition récursive d'un type

On ne dispose souvent que de quelques types de base : entiers, flottants, caractères, booléens.

On a pourtant besoin de types plus sophistiqués.

Par exemple, qu'est-ce qu'un mot ? Disons que c'est soit le vide, soit un caractère suivi d'un mot. Cela nous donne une règle de *construction* de proche en proche : on ajoute une à une dans l'ordre les lettres du mot pour le construire.

La structure attachée est celle d'un arbre. Voici celui correspondant au mot « glop ».



On choisit un *opérateur infixe* **:+** pour noter le *constructeur* par ajout d'un caractère à gauche. Comme il faut bien partir de quelque chose, on crée aussi le constructeur **Vide** qui construit un mot vide.
Sur Haskell, ce genre de type se crée simplement avec **data** après avoir créé un opérateur infixe avec la fonction **infixr** :

Haskell
```haskell
infixr 1 :+

data Mot =
  Vide
  | Char :+ Mot
```

Le « r » après **infix** indique que l'opérateur est associatif à droite :
'a' :+ ('b' :+ ('c' :+ Vide)) == 'a' :+ 'b' :+ 'c' :+ Vide
Le **1** indique le niveau de priorité sur les autres opérateurs. Par exemple, le niveau de ***** est 7 alors que celui de **+** est 6.
Le **|** indique l'alternative.
On va rajouter un petit sucre dont on reparlera plus tard : la dérivation. Pour pouvoir afficher nos objets dans l'interpréteur, nous allons faire en sorte que notre type **Mot** fasse partie de la classe **Show** :

Haskell
```haskell
data Mot =
  Vide
  | Char :+ Mot
  deriving (Show)
```

On peut maintenant créer un mot :

Haskell
```haskell
*Main> let m = 'a' :+ 'b' :+ Vide
*Main> m
'a' :+ ('b' :+ Vide)
*Main> :t m
m :: Mot
```

Pour pouvoir travailler sur ces « Mots », en plus des constructeurs, on a besoin de *sélecteurs* qui vont permettre de désigner les éléments qui ont permis de construire le mot. Les sélecteurs permettent en fait de *découper* les objets construits.
Ici, on construit des arbres par la gauche : nous avons besoin de distinguer la *tête* du mot qui est son premier caractère, de sa *queue* qui est le mot privé de sa tête. Attention ! Un mot vide n'a ni queue ni tête...
Par exemple, pour la tête :

Haskell
```haskell
tete :: Mot -> Char
-- renvoie le premier caractère d'un mot avec un filtrage par motif
tete Vide = error "Mot vide !"
-- un mot vide n'a pas de tête
tete (t :+ q) = t
```

Recherche

Déterminez une fonction **`queue :: Mot -> Mot`** qui renvoie la queue d'un mot.

Par exemple :

Haskell

```
*Main> tete m
'a'
```

On pourrait avoir besoin de *testeurs* qui nous permet de savoir comment a été construit l'objet. Ici, ce serait un test **`estVide`** mais on peut s'en passer grâce au *filtrage par motif*.
En ce qui concerne Haskell, on peut donc avoir à comparer un mot au mot **`Vide`** : il faut donc dériver notre type de la classe **`Eq`** :

Haskell

```
data Mot =
  Vide
  | Char :+ Mot
  deriving (Show,Eq)

estVide :: Mot -> Bool
estVide m = m == Vide

*Main> estVide Vide
True

*Main> estVide m
False
```

Sélecteurs et testeurs sont alors liés par la relation :

$$\neg(\text{estVide } m) \leftrightarrow \big(m = (\text{tete } m) :+ (\text{queue } m)\big)$$

Cherchons maintenant à définir une fonction sur ce type **`Mot`**. Par exemple, on voudrait compter le nombre de « a » dans un mot.
Commençons par la signature :

Haskell

```
nba :: Mot -> Int
-- calcule le nombre de 'a' dans un mot
```

Nous allons étudier ensuite tous les cas de construction d'un mot. Il y en a deux.

— Si le mot est vide, alors son nombre de 'a' est 0.

— Sinon, le mot est construit par **`t :+ q`**.
Si `t` $= 'a'$, alors `nba mot` $= 1 +$ `nba q`, sinon, `nba mot` $=$ `nba q`.

Il n'y a plus qu'à transcrire cela en Haskell :

Haskell

```
nba Vide     = 0
nba (t :+ q) = (nba q) + (if t == 'a' then 1 else 0)
```

Autre syntaxe possible en filtrant sur le motif de **`t`** : :

Haskell

```
nba2 Vide = 0
nba2 (t :+ q)
  | t == 'a' = (nba2 q) + 1
  |otherwise =  nba2 q
```

Autre syntaxe en utilisant **`case`** :

Haskell

```
nba3 mot = case mot of Vide     -> 0
                       (t :+ q) -> (nba3 q) + (if t == 'a' then 1 else 0)
```

Une manière plus fonctionnelle de procéder est d'utiliser un pliage mais la fonction **foldl** est de signature **foldl :: (a -> b -> a) -> a -> [b] -> a** donc ne peut plier que des listes.
Il faudrait soit créer une fonction de pliage de mots, soit créer une fonction qui à un mot associe la liste de ses caractères : voici de quoi occuper vos loisirs :)
Comme je suis très gentil, je vous propose une solution pour le pliage (pour vous aider d'ailleurs à mieux comprendre le pliage gauche...) :

Haskell

```haskell
pliage :: (a -> Char -> a) -> a -> Mot -> a
-- adapte foldl aux mots
pliage fonc ini Vide     = ini
pliage fonc ini (t :+ q) = pliage fonc (fonc ini t) q


nba4 :: Mot -> Int
-- calcule le nombre de 'a' dans un mot par pliage
nba4 mot =
  pliage (\accu t -> accu + (if t == 'a' then 1 else 0)) 0 mot
```

Recherche

1. Déterminez une fonction **longueur :: Mot -> Int** qui renvoie le nombre de caractères constituant un mot.
2. Déterminez une fonction **renverse :: Mot -> Mot** qui inverse l'ordre d'un mot.
3. Déterminez une fonction **colle :: Mot -> Mot -> Mot** qui concatène deux mots.
4. Déterminez une fonction **liste_of_Mot :: Mot -> [Char]** qui à un mot associe la liste de ses caractères.

4.2 Zoom sur les listes

Vous étudierez en algo2 les listes d'un point de vue « bas niveau », c'est-à-dire en mettant les mains dans le cambouis de la machine.
En mathématique, nous restons propres et proches de la nature. Les listes ne seront pas des parties d'un moteur polluant mais de beaux arbres bons pour la planète. Malgré tout, ce seront des arbres un peu spéciaux car ils seront plats...
Ce type pré-existe bien sûr en Haskell. Dans un but pédagogique, nous allons cependant le reconstruire à la main.

4.2.1 Le constructeur liste

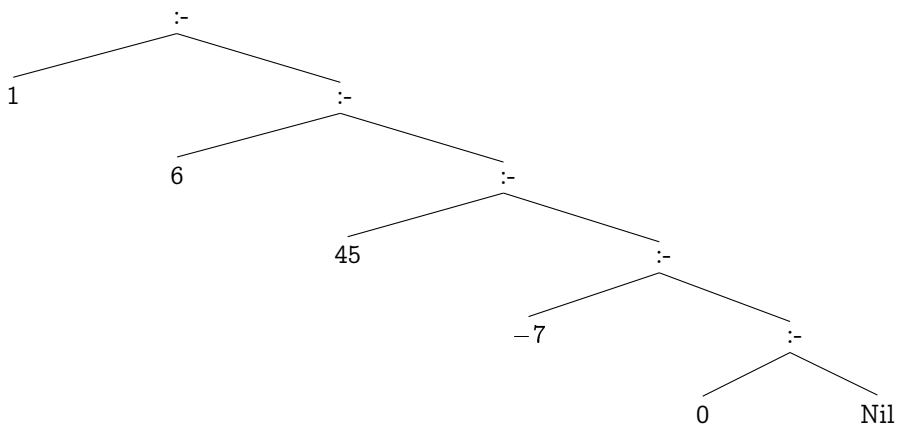
Bon, nous avons vu les mots dans la section précédente donc vous devriez pouvoir créer sans souci un type **Liste**...il faudrait juste préciser ce que nous entendons par liste !
C'est un ensemble d'éléments de *même type* placés dans un *ordre donné*.
Conventionnellement, on note les listes en plaçant leurs élément séparés par des virgules entre crochets. Par exemple, voici une liste d'entiers : $[1, 6, 45, -7, 0]$.
Nos mots étaient en fait des listes de caractères, mais on veut aussi pouvoir créer des listes d'entiers, de booléens, de schmurx, etc.
Nous allons donc créer un type *polymorphe*...

Haskell
```
infixr 1 :-

data Liste a =
  Nil
  | a :- (Liste a)
  deriving (Show,Eq,Ord)
```

Nous choisissons cependant de choisir nos éléments dans un ensemble ordonné pour rendre nos listes plus riches (...et surtout « triables »).
Par exemple :

Haskell
```
*Main> let l1 = 1 :- 6 :- 45 :- (-7) :- 0 :- Nil
*Main> l1
1 :- (6 :- (45 :- (-7 :- (0 :- Nil))))
*Main> :t l1
l1 :: Liste Integer
```

La machine (et vous...) voit en fait ça :



On peut créer des listes de booléens si ça nous chante :

Haskell
```
*Main> let l2 = True :- False :- True :- True :- Nil
*Main> :t l2
l2 :: Liste Bool
```

et même des listes de listes !

Haskell
```
*Main> let a_1 = 1 :- 2 :- Nil
*Main> let a_2 = 3 :- 4 :- Nil
*Main> let a_3 = 5 :- 6 :- Nil
*Main> let big_a = a_1 :- a_2 :- a_3 :- Nil
*Main> :t big_a
big_a :: Liste (Liste Integer)
```

Recherche

Déterminez les sélecteurs.

4 2 2 Les fonctions indispensables

Voici les outils minimums tels qu'ils sont décrits dans la documentation du module **`Data.List`** de Haskell. Adaptez-les à notre type **`Liste`** et donnez-leur un nom français pour éviter les conflits avec les fonctions de Haskell.

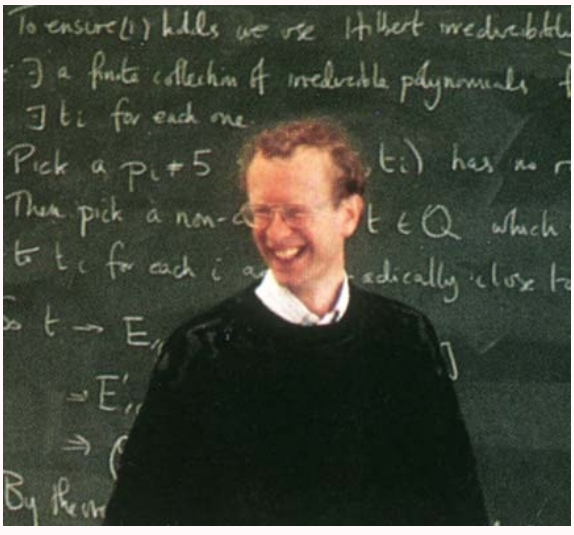
Extrait de la documentation de Data.List
```
  (++) :: [a] -> [a] -> [a]
```

```
Append two lists, i.e.,
 [x1, ..., xm] ++ [y1, ..., yn] == [x1, ..., xm, y1, ..., yn]
 [x1, ..., xm] ++ [y1, ...] == [x1, ..., xm, y1, ...]
If the first list is not finite, the result is the first list.

head :: [a] -> a
Extract the first element of a list, which must be non-empty.

last :: [a] -> a
Extract the last element of a list, which must be finite and non-empty.

tail :: [a] -> [a]
Extract the elements after the head of a list, which must be non-empty.

init :: [a] -> [a]
Return all the elements of a list except the last one. The list must be non-empty.

null :: [a] -> Bool
Test whether a list is empty.

length :: [a] -> Int
length returns the length of a finite list as an Int. It is an instance of the more
    general Data.List.genericLength, the result type of which may be any kind of
    number.

map :: (a -> b) -> [a] -> [b]   Source
map f xs is the list obtained by applying f to each element of xs, i.e.,
 map f [x1, x2, ..., xn] == [f x1, f x2, ..., f xn]
 map f [x1, x2, ...] == [f x1, f x2, ...]

reverse :: [a] -> [a]   Source
reverse xs returns the elements of xs in reverse order. xs must be finite.

foldl :: (a -> b -> a) -> a -> [b] -> a Source
foldl, applied to a binary operator, a starting value (typically the left-identity
    of the operator), and a list, reduces the list using the binary operator, from
    left to right:
 foldl f z [x1, x2, ..., xn] == (...((z `f` x1) `f` x2) `f`...) `f` xn
The list must be finite.

any :: (a -> Bool) -> [a] -> Bool       Source
Applied to a predicate and a list, any determines if any element of the list
    satisfies the predicate.

all :: (a -> Bool) -> [a] -> Bool       Source
Applied to a predicate and a list, all determines if all elements of the list
    satisfy the predicate.

sum :: Num a => [a] -> a        Source
The sum function computes the sum of a finite list of numbers.

product :: Num a => [a] -> a    Source
The product function computes the product of a finite list of numbers.

maximum :: Ord a => [a] -> a    Source
maximum returns the maximum value from a list, which must be non-empty, finite, and
    of an ordered type. It is a special case of maximumBy, which allows the
    programmer to supply their own comparison function.

minimum :: Ord a => [a] -> a    Source
minimum returns the minimum value from a list, which must be non-empty, finite, and
    of an ordered type. It is a special case of minimumBy, which allows the
    programmer to supply their own comparison function.

takeWhile :: (a -> Bool) -> [a] -> [a]  Source
takeWhile, applied to a predicate p and a list xs, returns the longest prefix
```

```
    (possibly empty) of xs of elements that satisfy p:
 takeWhile (< 3) [1,2,3,4,1,2,3,4] == [1,2]
 takeWhile (< 9) [1,2,3] == [1,2,3]
 takeWhile (< 0) [1,2,3] == []

elem :: Eq a => a -> [a] -> Bool        Source
elem is the list membership predicate, usually written in infix form, e.g., x `elem`
    xs.

find :: (a -> Bool) -> [a] -> Maybe a   Source
The find function takes a predicate and a list and returns the first element in the
    list matching the predicate, or Nothing if there is no such element.

filter :: (a -> Bool) -> [a] -> [a]     Source
filter, applied to a predicate and a list, returns the list of those elements that
    satisfy the predicate; i.e.,
 filter p xs = [ x | x <- xs, p x]

partition :: (a -> Bool) -> [a] -> ([a], [a])   Source
The partition function takes a predicate a list and returns the pair of lists of
    elements which do and do not satisfy the predicate, respectively; i.e.,
 partition p xs == (filter p xs, filter (not . p) xs)

zip :: [a] -> [b] -> [(a, b)]   Source
zip takes two lists and returns a list of corresponding pairs. If one input list is
    short, excess elements of the longer list are discarded.
```

CHAPITRE 5

Arithmétique : un premier survol



En 1994, le mathématicien britannique Andrew Wiles fit sensation en démontrant le grand théorème de Fermat. Ce vieux problème d'arithmétique avait tenu les mathématiciens en haleine pendant trois siècles. Pour le vaincre, Wiles travailla sur les courbes elliptiques dont l'étude mêle arithmétique, structures algébriques, géométrie, analyse. Ces travaux et leurs dérivés ont permis d'assurer la sécurité des cartes à puces, des échanges sur internet : la recherche fondamentale en mathématiques sur des nombres entiers fit faire un bon en avant à l'informatique. Quoi de plus normal après tout : un ordinateur n'est rien de plus qu'une machine calculant avec des zéros et des uns...

1 Structures mères

1 1 Lois de composition interne

La notion de loi de composition interne est primordiale, tant en mathématiques qu'en informatique : la notion de type en informatique impose un certain respect de la loi...

Haskell

```
*Main> let un = 1 :: Int
*Main> :t un
un :: Int
*Main> pi
3.141592653589793
*Main> :t pi
pi :: Floating a => a
*Main> un + pi

<interactive>:166:6:
    No instance for (Floating Int)
      arising from a use of ‘pi’
    Possible fix: add an instance declaration for (Floating Int)
    In the second argument of ‘(+)’, namely ‘pi’
    In the expression: un + pi
    In an equation for ‘it’: it = un + pi
```

On n'additionne pas n'importe qui avec n'importe quoi.
Formalisons la chose :

Définition 5 - 1

Loi de composition interne
Un loi de composition interne $\star$ définie sur un ensemble E est une application de $E \times E$ vers E. On dit alors que $(E, \star)$ est un **magma**.

Le plus souvent, si $\star$ est la loi, on note $x \star y$ au lieu de $\star(x, y)$.
Par exemple $(\mathbb{N}, +)$, $(\mathcal{P}(E), \cap)$, un langage L muni de la concaténation, sont trois magmas mais $(\mathbb{N}, -)$ n'est pas un magma.

Définition 5 - 2

Élément neutre
C'est un élément de E qui vérifie, pour tout élément x de E,

$$e \star x = x \star e = x$$

Quels sont les éléments neutres des magmas cités plus haut ? Si un magma admet un élément neutre, celui-ci est unique : pouvez-vous le démontrer ?

Définition 5 - 3

Monoïde
On appelle monoïde tout couple $(E, \star)$ tel que la loi $\star$ soit *associative*. Si en plus le monoïde admet un élément neutre, on dit que le monoïde est **libre** (ou **unitaire**).

Les magmas précédemment cités sont-ils des monoïdes ? Des monoïdes libres ?

Théorème 5 - 1

Sous-monoïde
Soit $(E, \star)$ un monoïde et $M \subseteq E$ tel que M soit *stable* par $\star$. Alors $(M, \star)$ est un monoïde : c'est un **sous-monoïde** de $(E, \star)$.

Enfin, le définition suivante introduit une notion fondamentale en arithmétique et en algèbre en général :

Définition 5 - 4

Élément inversible

Soit $(E, \star)$ un monoïde unitaire d'élément neutre e. Un élément x de E est inversible (ou symétrisable) par rapport à $\star$ s'il existe y dans E tel que

$$x \star y = y \star x = e$$

Recherche

Montrez que si a admet un symétrique, celui-ci est unique.

1 2 Les groupes

É.GALOIS (1811-1832)

Mort dans un duel à l'âge de vingt ans, Évariste GALOIS a tout de même eu le temps de révolutionner les mathématiques et ses travaux ont eu une très grande influence sur le développement récent de la cryptographie depuis ces trente dernières années.
La notion de groupe a d'abord été liée à la recherche des solutions d'une équation : peut-on trouver un nombre entier naturel n tel que $n + 2 = 0$?
Comme ce genre de problème a besoin d'être résolu (j'ai monté deux étages et je suis arrivé au rez-de-chaussée : de quel étage suis-je parti ?), la structure de groupe a été étudiée de près.
Vous me direz, ce genre de problème n'est pas vraiment révolutionnaire ni utile. Pourtant, nous verrons en TD que cette structure permet d'assurer la sécurité des cartes à puces et des échanges sur la toile...
Donnons-en tout d'abord une définition :

Définition 5 - 5

Groupe

Un groupe est un monoïde unitaire tel que tout élément est inversible.

Les lois de groupe sont souvent notées $\times$ ou $+$.
On note alors :

$$\underbrace{x + x + \cdots + x}_{k \text{ fois}} = k \cdot x \qquad \text{ou} \qquad \underbrace{x \times x \times \cdots \times x}_{k \text{ fois}} = x^k$$

Que peut-on dire de $\mathbb{Z}$ muni de l'addition ?
Nous travaillerons en général avec des groupes ayant seulement un nombre fini d'éléments :

Définition 5 - 6

Groupe fini

Soit $(G, +)$ un groupe d'élément neutre 0_E ayant un nombre fini d'éléments. Le cardinal de G est appelé l'**ordre du groupe** G.
Soit x un élément de G. L'**ordre de l'élément** x est le plus petit entier k tel que $k \cdot x = 0$.

Étudions maintenant le groupe qui nous occupera le restant de notre chapitre...

1 3 Le groupe (Z/nZ,+)

1 3 1 Congruence

Définition 5 - 7

Congruence des entiers

Soit n un entier naturel non nul.
Deux éléments a et b du groupe $(\mathbb{Z}, +)$ sont **congrus modulo** n si, et seulement si, ils ont le même reste dans la division par n. On note

$$a \equiv b \pmod{n}$$

et on lit « *a est congru à b modulo n* ».

Cette notion de « modulo » est entrée dans le langage courant du geek :

> *« Well, LISP seems to work okay now, modulo that GC bug. »*
> *« I feel fine today modulo a slight headache. »*
>
> in « The New Hacker's Dictionnary »
> http://outpost9.com/reference/jargon/jargon_28.html#SEC35

Vérifiez tout d'abord que la relation de congruence est une *relation d'équivalence*. On peut donc définir un ensemble quotient, c'est-à-dire les relations d'équivalence.
Prenons un exemple innocent : travaillons modulo 2. Combien y a-t-il de classes d'équivalence ? Quel est l'ensemble quotient ?
On a pris l'habitude de noter $\mathbb{Z}/n\mathbb{Z}$ l'ensemble quotient de la relation de congruence modulo n.
On notera dans ce cours $\overline{k}^n$ la classe de k modulo n.

Attention !
La plupart des langages possèdent une fonction **mod** ou quelque chose d'équivalent. Ainsi **a mod n** renvoie parfois l'élément de la classe d'équivalence de **a** compris entre 0 et $n-1$, d'autre fois entre $-n+1$ et $n-1$ selon le signe de a mais dans tous les cas le nombre qui vérifie **a = (a/n)*n + x mod n**.
Avec Caml :

OCaml
```
        Objective Caml version 3.12.0

# 17 mod 3 ;;
- : int = 2
# -17 mod 3 ;;
- : int = -2
# -17/3;;
- : int = -5
# (-17 / 3)*3 + (-17 mod 3) ;;
- : int = -17
```

Avec Python :

Python
```
Python 2.7.2+ (default, Oct 4 2011, 20:06:09)
>>> 17 % 3
2
>>> -17 % 3
1
>>> -17/3
-6
>>> (-17 // 3)*3 + (-17 % 3)
-17
```

Avec Haskell, on a les deux :)

Haskell
```
*Main> mod 17 3
2
*Main> mod (-17) 3
1
*Main> div (-17) 3
-6
*Main> 3 * (div (-17) 3) + (mod (-17) 3)
-17
*Main> rem (-17) 3
-2
*Main> quot (-17) 3
-5
*Main> 3 * (quot (-17) 3) + (rem (-17) 3)
-17
*Main> 3 * (quot (-17) 3) + (mod (-17) 3)
```

```
-14
*Main> 3 * (div (-17) 3) + (rem (-17) 3)
-20
```

1 3 2 Division euclidienne

Pour éviter ce genre d'ambiguïté, nous allons nous mettre d'accord grâce au théorème suivant :

Théorème 5 - 2 **Division euclidienne**

Soit a un entier relatif et b un entier naturel non nul.
Il existe un unique couple d'entiers (q, r) tels que

$$a = bq + r \quad \text{avec} \quad 0 \leqslant r < b$$

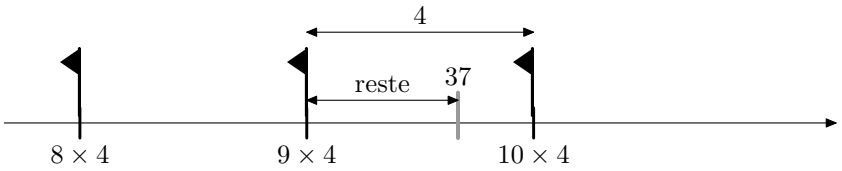
Déterminer q et r, c'est effectuer la division euclidienne de a par b.
On appelle a le dividende, b le diviseur, q le quotient et r le reste.

Remarque

Si l'on remplace la condition sur le reste par $0 \leqslant |r| < b$, il n'y a plus unicité (c'est d'ailleurs la formulation habituelle qui permet de travailler dans les mêmes conditions dans des structures plus vastes qu'on appelle *anneaux euclidiens*) ce qui explique que Python et Ocaml ne sont pas d'accord...

Papyrus d'une copie des éléments d'Euclide datant du premier siècle après JC, quatre siècles après la mort du mathématicien grec.

Qui dit théorème dit démonstration. Le principe de celle qui va suivre est le schéma suivant qui traduit la division de 37 par 4 :



qui traduit qu'un entier a est soit un multiple de b, soit est encadré par deux multiples consécutifs de b.
L'énoncé contient deux termes importants : « il *existe* un *unique* couple ». Il va donc falloir prouver deux choses :
- qu'un tel couple existe
- qu'il est unique.

Traitons d'abord le cas particulier où $a \in \mathbb{N}$.

Existence Le monde des multiples de b se sépare en deux catégories : ceux qui sont inférieurs (ou égaux) à a et les autres.

En d'autres termes, appelons $\mathcal{M}_i$ l'ensemble des multiples de b inférieurs ou égaux à a. Cet ensemble $\mathcal{M}_i$ est non vide car il contient au moins 0. De plus $\mathcal{M}_i$ est majoré par a puisqu'on l'a défini comme ça.

L'ensemble $\mathcal{M}_i$ est donc une partie de $\mathbb{N}$ non vide et majorée : $\mathcal{M}_i$ admet donc un plus grand élément d'après une propriété que vous avez démontrée.

Appelons μ ce plus grand élément. C'est un multiple de b, donc il existe un entier q tel que $\mu = qb$.

Le multiple suivant est $\mu + b = qb + b = (q + 1)b$ qui n'appartient pas à $\mathcal{M}_i$ car il est strictement supérieur au maximum μ.

On en déduit que

$$bq \leqslant a < bq + b$$

C'est ce que nous « montrait » le dessin. Il nous indique aussi que le reste correspond en fait à la différence $a - bq$.

Posons donc $r = a - bq$: on a alors $a = bq + r$ et donc $bq \leqslant bq + r < bq + b$, c'est à dire $0 \leqslant r < b$

Nous avons donc démontré l'existence de deux entiers q et r tels que $a = bq + r$ avec $0 \leqslant r < b$, *inspirés* que nous étions par un joli dessin.
Il reste à démontrer l'unicité d'un tel couple d'entiers.

unicité Une méthode habituelle pour démontrer une unicité est *supposer* l'existence d'un second couple.
Supposons donc qu'il existe deux couples d'entiers (b, q) et (b', q') vérifiant

$$a = bq + r \quad \text{avec} \quad 0 \leqslant r < b$$
$$a = bq' + r' \quad \text{avec} \quad 0 \leqslant r' < b$$

Effectuons la différence membre à membre de ces égalités. On obtient

$$0 = b(q - q') + r - r' \quad \text{avec} \quad -b < r - r' < b$$

Vous en déduisez comme moi que $r - r'$ est un multiple de b, et que ce multiple est strictement compris entre $-b$ et b.
Le seul multiple qui convient est 0, donc $r - r' = 0$ et par suite $q - q' = 0$, c'est à dire que $r = r'$ et $q = q'$. Dès lors il n'existe qu'un seul couple solution.

Définition 5 - 8

Diviseur
Un entier non nul d divise (ou est un diviseur de) l'entier a si, et seulement si, le reste de la division euclidienne de a par d est nul.

Recherche

Pourquoi dit-on habituellement qu'on ne peut pas diviser par zéro ? D'ailleurs, dans de nombreux cas, le debugger de votre langage préféré renvoie des messages du type `*** Exception: divide by zero`...
Vous traiterez ensuite le cas où a est strictement négatif connaissant bien sûr le résultat pour $a \in \mathbb{N}$.

En général, on prend comme *représentant principal* de la classe de a modulo n le reste de la division de a par n. Par exemple, on notera :

$$\mathbb{Z}/8\mathbb{Z} = \{\overline{0}^8, \overline{1}^8, \overline{2}^8, \overline{3}^8, \overline{4}^8, \overline{5}^8, \overline{6}^8, \overline{7}^8\}$$

Danger

Notez bien que $\overline{0}^8$, $\overline{1}^8$, etc. sont des ensembles d'entiers !...
Ici, $\overline{0}^8 = \{\ldots, -16, -8, 0, 8, 16, 24, 32, \ldots\}$
Par **ABUS DE NOTATION**, on écrira le plus souvent k pour désigner $\overline{k}^n$ ce qui nous amènera à écrire par exemple que $8 = 0$, voire $3 \times 5 = 7 = -1$ comme nous le verrons plus loin...

Donnons quelques théorèmes importants que vous démontrerez en TD :

Théorème 5 - 3

Soit a et b deux entiers et n un entier naturel.
a est congru à b modulo n si et seulement si $a - b$ est un multiple de n

Cette propriété nous permet en fait d'exploiter autrement les congruences. En effet, si par exemple $x \equiv 5 (\mathrm{mod}\, 32)$, cela signifie qu'il existe un entier k tel que $x - 5 = 32k$, soit encore que $x = 5 + 32k$.
Voyons une autre formulation utile :

Théorème 5 - 4

$$a \equiv b (\mathrm{mod}\, n) \iff \exists k \in \mathbb{Z} \mid a = b + kn$$

Ça ressemble à la division euclidienne de a par n mais ça peut ne pas l'être : n'oubliez pas que le reste doit obligatoirement être positif et strictement inférieur au diviseur.
Par exemple on a bien $33 \equiv 97 (\mathrm{mod}\, 32)$ mais 97 n'est certes pas le reste de la division de 33 par 32.
Il faudrait maintenant pouvoir définir des opérations sur les classes de congruences. On aurait envie de dire que la somme des classes de a et de b modulo n est en fait la classe de la somme $a + b$ modulo n et pareil pour la multiplication mais est-ce licite ?

1 3 3 Premières propriétés

Nous aurons besoin des propriétés suivantes que vous allez prouver en TD :

Théorème 5 - 5

1. $\text{Card}\left(\mathbb{Z}/_{n\mathbb{Z}}\right) = n$
2. $a \equiv a \pmod{n}$
3. Si $a \equiv b \pmod{n}$, alors $b \equiv a \pmod{n}$
4. Si $a \equiv b \pmod{n}$ et $b \equiv c \pmod{n}$, alors $a \equiv c \pmod{n}$
5. Si $a \equiv b \pmod{n}$ et $a' \equiv b' \pmod{n}$, alors
$$a + a' \equiv b + b' \pmod{n} \quad a - a' \equiv b - b' \pmod{n} \quad aa' \equiv bb' \pmod{n}$$
6. Si $a \equiv b \pmod{n}$, alors, pour tout $p \in \mathbb{N}$, $a^p \equiv b^p \pmod{n}$

En fait, il faut retenir qu'on peut additionner, soustraire, multiplier des congruences **de même module**.
Nous avons oublié la division et pour cause : c'est une opération à haut risque quand on travaille avec des entiers !
Par exemple $12 \equiv 0 \pmod{6}$, mais $\frac{12}{3} \not\equiv \frac{0}{3} \pmod{6}$.

1 3 4 Loi de groupe

La propriété 5 du théorème 5 - 5 nous permet de définir une loi d'addition et une loi de multiplication sur $\mathbb{Z}/_{n\mathbb{Z}}$:

Théorème 5 - 6

$$\forall n \in \mathbb{N}^*,\ \forall (x,y) \in \mathbb{Z}^2,\quad \overline{x}^n + \overline{y}^n = \overline{x+y}^n,\quad \overline{x}^n \cdot \overline{y}^n = \overline{x \cdot y}^n$$

Écrivez par exemple les lois d'addition et de multiplication de $\mathbb{Z}/_{7\mathbb{Z}}$ et $\mathbb{Z}/_{8\mathbb{Z}}$: des remarques ?
Est-ce que $(\mathbb{Z}/_{n\mathbb{Z}}, +)$ et $(\mathbb{Z}/_{n\mathbb{Z}}, \cdot)$ sont des groupes ?
Dire que p est **inversible modulo** n signifie qu'il existe un entier p' tel que $p \cdot p' = 1 \pmod{n}$, c'est-à-dire qu'il existe un entier k tel que $pp' = 1 + kn$. Nous venons de voir pour certaines valeurs de n, cette recherche semble possible et qu'elle pose problème dans d'autres cas. Nous allons avoir besoin de quelques théorèmes de plus pour conclure à coup sûr, cette notion d'inverse modulaire étant centrale en cryptographie donc en sécurité informatique.

1 3 5 Calcul modulaire en Haskell

On peut créer un type enregistrement pour travailler modulo un entier :

Haskell
```haskell
data Classe =
  Zero_c
  |Un_c
  |Classe {modulo :: Int, representant :: Int}
```

On fait de **`Classe`** une instance de **`Eq`** pour gérer les égalités :

Haskell
```haskell
instance Eq Classe where
  a == Zero_c = (mod (representant a) (modulo a) == 0)
  a == Un_c   = (mod (representant a) (modulo a) == 1)
  a == b      = (mod (representant a) (modulo a) == mod (representant b) (modulo b))
                && ((modulo a) == (modulo b))
```

On fait de **`Classe`** une instance de **`Show`** pour gérer l'affichage

Haskell
```haskell
instance Show Classe where
  show Zero_c = "0%"
  show Un_c = "1%"
  show a = (show (representant a)) ++ "%" ++ (show (modulo a))
```

Pour plus de commodité, on va créer un opérateur infixe correspondant à notre $\cdots \equiv \cdots [\cdots]$:

Haskell
```
(%) :: Int -> Int -> Classe
(%) a n = Classe {representant = (mod a n), modulo = n}
```

On obtient alors :

Haskell
```
*Main> 13 % 3
1%3
```

On peut ensuite créer des lois induites :

Haskell
```
- somme induite
(%+) :: Classe -> Classe -> Classe
(%+) Zero_c a = a
(%+) a Zero_c = a
(%+) Un_c   a = Classe {representant = (representant a) + 1, modulo = modulo a}
(%+) a Un_c   = Classe {representant = (representant a) + 1, modulo = modulo a}
(%+) a b      =
  Classe{
    modulo =
       if (modulo a) == (modulo b)
       then modulo a
       else error "Classes incompatibles" ,
    representant = mod ( (representant a) + (representant b)) (modulo a)
    }

-- produit induit
(%*) :: Classe -> Classe -> Classe
(%*) Zero_c Un_c = Zero_c
(%*) Zero_c a    = 0%(modulo a)
(%*) a Zero_c    = 0%(modulo a)
(%*) Un_c a      = a
(%*) a Un_c      = a
(%*) a b         =
  Classe{
    modulo =
       if (modulo a) == (modulo b)
       then modulo a
       else error "Classes incompatibles" ,
    representant = mod ( (representant a) * (representant b)) (modulo a)
    }
```

On fait alors de **`Classe`** une instance de **`Num`** pour surcharger les opérateurs **`+`** et **`*`**, gérer les opposés :

Haskell
```
instance Num Classe where
  (+)           = (%+)
  (*)           = (%*)
  negate Zero_c = Zero_c
  negate b      = Classe{representant = mod (- (representant b)) (modulo b), modulo
      = (modulo b)}
  fromInteger 0 = Zero_c
  fromInteger 1 = Un_c
  abs a         = a
  signum a      = 1
```

Alors, pour obtenir $9 + 7$ modulo 5 :

Haskell
```
*Main> 9%5 + 7%5
1%5
```

2 Divisibilité dans Z

2 1 Propriété fondamentale de N

Nous admettrons la propriété suivante qui nous sera très utile par la suite :

Théorème 5 - 7

Toute partie non vide de $\mathbb{N}$ possède un plus petit élément.

Est-ce que cette propriété est encore vraie dans $\mathbb{Z}$? Dans $\mathbb{R}$?

2 2 PGCD

Considérons un entier relatif a. On notera $\mathcal{D}(a)$ l'ensemble de ses diviseurs dans $\mathbb{Z}$.
Par exemple, $\mathcal{D}(0) = \mathbb{Z}$, $\mathcal{D}(1) = \{-1, 1\}$, $\mathcal{D}(12) = \{-12, -6, -4, -3, -2, -1, 1, 2, 3, 4, 6, 12\}$ et plus généralement

$$\mathcal{D}(a) = \{-|a|, \dots, -1, 1, \dots, |a|\}$$

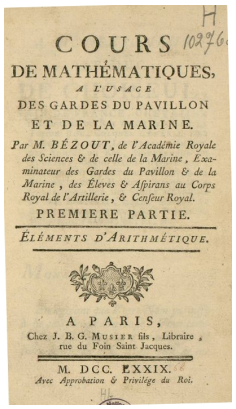
Définition 5 - 9

PGCD
Soit a et b deux entiers. On appelle PGCD de a et b et on note $a \wedge b$ l'entier défini de la manière suivante :

- $0 \wedge 0 = 0$
- Si a et b ne sont pas simultanément nuls, $a \wedge b$ est le plus grand entier naturel qui divise simultanément a et b.

Le PGCD de a et b est donc le plus grand élément de $\mathcal{D}(a) \cap \mathcal{D}(b)$: la notation le rappelle. Comme toute partie non vide et majorée de $\mathbb{N}$ admet un plus grand élément, cette définition en est bien une car $\mathcal{D}(a) \cap \mathcal{D}(b)$ contient au moins 1.

2 3 Égalité de Bézout


COURS
DE MATHÉMATIQUES,
A L'USAGE
DES GARDES DU PAVILLON
ET DE LA MARINE.
Par M. BÉZOUT, de l'Académie Royale des Sciences & de celle de la Marine, Examinateur des Gardes du Pavillon & de la Marine, des Élèves & Aspirans au Corps Royal de l'Artillerie, & Censeur Royal.
PREMIERE PARTIE.
ÉLÉMENTS D'ARITHMÉTIQUE.
A PARIS,
Chez J. B. G. MUSIER fils, Libraire, rue du Foin Saint Jacques.
M. DCC. XXXIX.
Avec Approbation & Privilège du Roi.

Première page du cours écrit par Étienne Bézout (1730-1783)

Voici la clé de notre section : nous allons montrer que notre PGCD est en fait une combinaison linéaire de a et de b. Considérons d'abord le cas où a et b sont non nuls et notons

$$S = \{au + bv \mid (u, v) \in \mathbb{Z}^2 \text{ et } au + bv > 0\}$$

L'ensemble S est constitué d'entiers naturels et est non vide car il contient au moins $a^2 + b^2$: il admet donc un unique **plus petit élément** que nous noterons d qui s'écrit donc $d = au_0 + bv_0$.
Notre mission va bien sûr consister à prouver que d est en fait le PGCD de a et b.
Nous allons pour cela utiliser une ficelle classique : introduire le reste de la division euclidienne de a par d et utiliser le fait que d est le plus petit élément de S puis raisonner par l'absurde.
La division de a par d s'écrit :

$$a = dq + r \quad \text{avec} \quad 0 \leqslant r < d$$

Supposons que $r > 0$, alors

$$r = a - dq = a - q(au_0 + bv_0) = a(1 - qu_0) + b(-qv_0) > 0$$

et donc $r \in S$ or $r < d$. C'est impossible car d est le plus petit élément de S : **contradiction**.
Notre supposition de départ est donc fausse et $r = 0$. Nous en déduisons que d divise a.
On montre de manière similaire que d divise b. Ainsi

$$d \text{ est un diviseur commun de } a \text{ et } b$$

Il nous reste à montrer que c'est le plus grand. Soit δ un diviseur commun à a et b quelconque. On montre facilement qu'alors δ divise toute combinaison linéaire de a et de b donc en particulier δ divise $au_0 + bv_0$ donc d. Alors $c \leqslant |d| = d$, donc d est bien le plus grand des diviseurs.

Il faut encore vérifier que le PGCD est **unique**. La méthode habituelle est de supposer qu'il existe un deuxième PGCD, disons d'. Alors $d \leqslant d'$ car d' est le plus grand diviseur puis $d' \leqslant d$ car d aussi et finalement $d = d'$ ce qui assure l'unicité.

Comme $|a| = \pm 1 \times a + 0 \times 0$, l'égalité tient toujours si b (ou a) est nul, donc

Théorème 5 - 8

Égalité de Bézout

Soit a et b deux entiers relatifs. Si $d = a \wedge b$, alors il existe deux entiers u et v tels que :

$$au + bv = d$$

2 4 Algorithme des différences

Voyons maintenant une propriété qui va être à la base de la construction algorithmique du PGCD :

Théorème 5 - 9

Si $ab \neq 0$ et k un entier quelconque

$$a \wedge (b + ka) = a \wedge b$$

et en particulier

$$a \wedge b = a \wedge (b - a) = a \wedge (a - b)$$

Notons $a \wedge b = d$ et $a \wedge (b + ka) = d'$. Alors

- d divisant a et b, d divise $(ka + b)$ et a donc divise d' d'après une propriété précédente.
- d'autre part, d' divise a et $b + ka$, donc divise $b + ka - ka$, donc b, donc d' est un diviseur commun à a et b donc d' divise d.

Comme $d|d'$ et $d'|d$, on a donc $d = d'$.

Cette propriété va nous permettre de fabriquer un algorithme de calcul de $a \wedge b$. En effet, si $a \geqslant b$, et comme $a \wedge b = b \wedge (a - b)$, on calcule le PGCD de deux entiers plus petits. Et on réitère. Par exemple :

$$15 \wedge 12 = 12 \wedge 3 = 9 \wedge 3 = 6 \wedge 3 = 3 \wedge 3 = 3 \wedge 0 = 3$$

Comment se persuader que cette descente aboutit ? Nous pouvons aisément nous convaincre sans se perdre dans trop de formalisme que :

Théorème 5 - 10

Suite strictement décroissante d'entiers naturels

Toute suite strictement décroissante d'entiers naturels est constante à partir d'un certain rang.

Et cette constante ne pouvant être par construction que 0, le PGCD de a et b sera la dernière valeur non nulle de cette suite car $k \wedge 0 = k$ pour tout entier k.

Ce procédé semble assez long : beaucoup d'opérations sont inutiles. Malgré tout il n'utilise que des sommes et des différences qu'un ordinateur traite extrêmement rapidement.

2 5 Algorithme d'Euclide I

C'est le même principe que ce que nous venons de faire, mais en plus rapide car nous remarquons que, en notant

$$a = bq_0 + r_0 \quad \text{avec } 0 \leqslant r_0 < b$$

la division euclidienne de a par b, on obtient que

$$a \wedge b = b \wedge (a - bq_0) = b \wedge r_0$$

Puis en notant

$$b = r_0 q_1 + r_1 \quad \text{avec } 0 \leqslant r_1 < r_0$$

on obtient de même que

$$a \wedge b = b \wedge r_0 = r_0 \wedge r_1$$

et on continue ainsi à fabriquer une suite strictement décroissante d'entiers naturels : elle est donc finie et son dernier terme $r_p = 0$.

Alors

$$a \wedge b = r_{p-1} \wedge r_p = r_{p-1} \wedge 0 = r_{p-1} = \text{ le dernier reste non nul}$$

Soit, à la mode spaghetti :

```
Fonction euclide( a, b : entiers ) : entier
Si b = 0 Alors
    Retourner a
Sinon
    Retourner euclide(b,(mod a b))
FinSi
```

26 Algorithme d'Euclide II

L'algorithme d'Euclide nous assure que le PGCD de a et b est le dernier reste non nul de la suite (r_k) construite au paragraphe précédent. L'égalité de Bézout nous assure de l'existence de deux entiers u et v tels que $au + bv = a \wedge b$.
Nous allons essayer de combiner les deux en construisant deux suites (u_k) et (v_k) telles que

$$r_k = au_k + bv_k$$

Voici le départ

- $r_0 = a = 1 \times a + 0 \times b$
- $r_1 = b = 0 \times a + 1 \times b$
- $r_2 = \textbf{mod}\ (r_0, r_1)$

  $\vdots$
- $r_{i+1} = \textbf{mod}\ (r_{i-1}, r_i)$

  $\vdots$
- $r_{p-1} = a \wedge b$
- $r_p = 0$

Or par définition, comme $r_2 = \textbf{mod}\ (r_0, r_1)$, on a $r_0 = q_2 r_1 + r_2$, donc $r_2 = 1 \times r_0 - q_2 \times r_1$, c'est à dire

$$u_2 = 1 \quad \text{et} \quad v_2 = -q_2$$

On réitère le mécanisme. Supposons que $r_{k-1} = u_{k-1} \times a + v_{k-1} \times b$ et $r_k = u_k \times a + v_k \times b$
Comme $r_{k+1} = \textbf{mod}\ (r_k, r_{k-1})$, on a $r_{k-1} = q_{k+1} r_k + r_{k+1}$, donc $r_{k+1} = 1 \times r_{k-1} - q_{k+1} \times r_k$, c'est à dire

$$r_{k+1} = 1 \times (u_{k-1} \times a + v_{k-1} \times b) - q_{k+1} \times (u_k \times a + v_k \times b)$$

Finalement

$$u_{k+1} = u_{k-1} - u_k \times q_{k+1} \quad \text{et} \quad v_{k+1} = v_{k-1} - v_k \times q_{k+1}$$

Ça a l'air un peu brut comme ça, au premier coup d'œil, mais en fait il y a une disposition pratique qui permet de mieux voir ce qui se passe. D'abord, dans algorithme d'Euclide étendu il y a algorithme d'Euclide, donc on commence par rechercher le PGCD de a et b par l'algorithme d'Euclide en n'oubliant pas cette fois de noter les quotients en remplissant le tableau suivant :

k	u_k	v_k	r_k	q_k
0	1	0	$r_0 = a$	/
1	0	1	$r_1 = b$	q_1
2	$u_0 - u_1 q_1$	$v_0 - v_1 q_1$	$r_2 = r_0 - r_1 q_1$	q_2
3	$u_1 - u_2 q_2$	$v_1 - v_2 q_2$	$r_3 = r_1 - r_2 q_2$	q_3
$\vdots$			$\vdots$	$\vdots$
$p-1$			$r_{p-1} = a \wedge b$	q_{p-1}
p			$r_p = 0$	

Et le secret tient dans le schéma

—

(×)

=

et pareil pour les v_k et les r_k.
Par exemple, pour 19 et 15 :

k	u_k	v_k	r_k	q_k	
0				/	L_0
1					L_1
2	1	-1	4	3	$L_2 \leftarrow L_0 - \times L_1$
3	-3	4	3	1	$L_3 \leftarrow L_1 - 3 \times L_2$
4	4	-5	1	3	$L_4 \leftarrow L_2 - 1 \times L_3$
5			0		$L_5 \leftarrow L_3 - 3 \times L_4$

Le dernier reste non nul est 1 donc $19 \wedge 15 = 1$ et $4 \times 19 - 5 \times 15 = 1$
En version récursive, on peut procéder en deux étapes :

```
Fonction euc(u,v,r,u',v',r' : entiers) : liste d'entiers
Si r' = 0 Alors
    Retourner [u,v,r]
Sinon
    Retourner euc(u',v',r',u-div(r,r')*u',v-div(r,r')*v',r-div(r,r')*r')
FinSi
```

```
Fonction EUC(a,b : entiers) : liste d'entiers
Retourner euc(1,0,a,0,1,b)
```

2 7 Nombres premiers entre eux

Définition 5 - 10

Deux entiers a et b sont premiers entre eux si et seulement si

$$a \wedge b = 1$$

L'égalité de Bézout vue précédemment nous permet donc dénoncer le théorème suivant :

Théorème 5 - 11

Théorème de Bézout
Les entiers a et b sont premiers entre eux si et seulement s'il existe deux entiers u et v tels que $au + bv = 1$

$$a \wedge b = 1 \iff \text{il existe } (u, v) \in \mathbb{Z}^2 \text{ tel que } au + bv = 1$$

Nous avons donc en main tous les éléments pour étudier les éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$ mais avant, une petite digression...

3 À la recherche des nombres premiers

3 1 Définition

Quoi de plus simple qu'un nombre premier :

Définition 5 - 11

Un entier naturel est dit premier s'il est supérieur (i.e. supérieur ou égal) à 2 et n'est divisible que par 1 et lui-même.

et pourtant, ils renferment tant de mystères que les plus grands esprits depuis des siècles n'ont toujours pas réussi à en percer tous les secrets et ce malgré les énormes progrès technologiques et les investissements colossaux consentis par les pouvoirs tant civils que militaires pour assurer ou percer la confidentialité des transmissions de toutes natures qui continuent de dépendre d'une meilleure connaissance des nombres premiers, ce que nous verrons un peu plus loin. Qui sont-ils ? Combien sont-ils ? Où sont-ils ? À quoi servent-ils ? Nous essaierons de donner quelques éléments de réponses à ces questions.
Mais tout d'abord, pourquoi jouent-ils un rôle si important ? Fondamentalement, il existe deux manières d'engendrer $\mathbb{N}$:

— si on veut engendrer $\mathbb{N}$ en utilisant l'addition, on s'aperçoit que le nombre 1 nous suffit : on « fabrique » 2 en additionnant 1 avec lui-même ; 3 en additionnant 1 avec 2, etc.

— si on veut engendrer $\mathbb{N}$ en utilisant la multiplication, là, les choses se compliquent. Pour « fabriquer » 2, il faut le créer ; même problème pour 3. On fabrique 4 en multipliant 2 avec lui-même, mais il faut créer 5. On fabrique 6 en multipliant 3 avec 2. On crée 7. On fabrique 8 à partir de 2. On fabrique 9 à partir de 3. On fabrique 10 à partir de 2 et 5, etc.

  Les nombres que l'on est obligés de créer sont les briques nécessaires à fabriquer tous les autres. C'est bien plus compliqué que l'addition me direz-vous, mais la multiplication est plus « puissante » et nous permet d'aller bien plus vite et plus loin.

Les nombres premiers sont donc ces éléments qui nous permettent de fabriquer tous les autres. Un des premiers problèmes étudiés à été de savoir s'ils peuvent tenir dans une boîte. Euclide à répondu à cette question il y a vingt-trois siècles et la réponse est non.
Pour le prouver, nous aurons besoin d'un résultat intermédiaire :

Théorème 5 - 12

Tout entier naturel admet au moins un diviseur premier.

Si ce nombre, appelons-le n, est premier, tout va bien.
Sinon, l'ensemble de ses diviseurs étant non vide (n est dedans) et borné (par 1 et lui-même), il admet un plus petit élément $p \neq 1$. Ce nombre n'est pas divisible par un autre, sinon ce nombre plus petit que p divisant p diviserait n et alors p ne serait plus le plus petit diviseur de n. Le nombre p est donc premier et voilà.

Théorème 5 - 13

Il y a une infinité de nombres premiers.

Raisonnons par l'absurde et supposons qu'il existe exactement n nombres premiers qu'on nommera $p_1, p_2, \cdots, p_n$ et appelons N le nombre

$$N = p_1 p_2 \cdots p_n + 1$$

Il est plus grand que tous les p_i, donc il n'est pas premier d'après notre hypothèse, donc il admet un diviseur premier p qui est donc un des p_i, puisqu'il n'y a qu'eux. Soit i_0 tel que $p = p_{i_0}$. Alors p divise $p_1 p_2 \cdots p_n$. Or il divise N, donc il divise leur différence $N - p_1 p_2 \cdots p_n$, c'est à dire 1, donc $p = 1$ ce qui est absurde puisqu'il est premier. Ainsi, il n'existe pas de plus grand nombre premier.

Il y en a donc une infinité et l'aventure ne fait que commencer.

3 2 Comment vérifier qu'un nombre est premier ?

Observons les diviseurs de 1321 par exemple : si aucune des 1319 divisions ne « tombe juste », alors on pourra dire que 1321 premier. Et puis d'abord, il est impair, il ne semble pas être divisible par 3, alors pourquoi pas...

diviseur	2	3	4	5	6	7	8	9	10	11	12	13
quotient	660,5	440,3	330,3	264,2	220,2	188,7	165,1	146,8	132,1	120,1	110,1	101,6
diviseur	14	15	16	17	18	19	20	21	22	23	24	25
quotient	94,36	88,07	82,56	77,71	73,39	69,53	66,05	62,90	60,05	57,43	55,04	52,84
diviseur	26	27	28	29	30	31	32	33	34	35	36	37
quotient	50,81	48,93	47,18	45,55	44,03	42,61	41,28	40,03	38,85	37,74	36,69	35,70

Le tableau est incomplet : il reste encore à essayer les quotients de 38 à 1320, mais cela aurait été mauvais pour la planète.

Tout d'abord, nous n'avons pas trouvé de quotient entier en ce début d'enquête. Nous observons de plus que si la suite des diviseurs est croissante, celle des quotients est décroissante. Vous avez sûrement remarqué qu'à partir de 37, les quotients sont inférieurs aux diviseurs donc nous pouvons nous arrêter là : si pour un diviseur supérieur à 37, on trouvait un quotient entier q, alors en divisant 1321 par q, qui est inférieur à 37, on devrait se trouver au début de notre liste de diviseurs et ce diviseur serait entier. Le problème, c'est qu'aucune division par un entier inférieur à 37 n'a donné de quotient entier donc on peut s'épargner les 1283 divisions restantes.

Mine de rien, nous venons de franchir un grand pas dans la théorie des nombres : pour tester si l'entier 1321 est premier, il nous a suffit d'effectuer 36 divisions[a]. Deux problèmes se posent maintenant : pourquoi 36 et pouvons-nous généraliser ce résultat aux autres entiers ?

Un petit Joker : $\sqrt{1321} \approx 36,3...$

Reprenons la propriété 5 - 12 page ci-contre. Tout naturel n admet un plus petit diviseur p qui est premier. Écartons le cas où n est premier et donnons un nom aux nombres qui restent :

Définition 5 - 12 Un entier naturel autre que 1 qui n'est pas premier est dit **composé**.

L'entier n étant composé, il s'écrit donc $n = pq$, avec q un entier supérieur à p (car p est le plus petit diviseur). Ainsi

$$p \leqslant q \quad \text{et donc} \quad p^2 \leqslant pq = n \quad \text{c'est à dire} \quad p \leqslant \sqrt{n}$$

Nous pouvons donc généraliser l'observation précédente

Théorème 5 - 14 Si un entier est composé, alors il admet un diviseur premier inférieur à sa racine carrée.

3 3 Tests et cribles

La théorie des nombres (i.e. la partie des mathématiques qui étudie les nombres premiers) est actuellement intimement liée à l'informatique et vice versa. Même à notre petit niveau, nous ne pouvons donc pas passer à côté...

3 3 1 Tests naïfs

On teste tous les entiers de 2 à $\sqrt{n}$ en passant par une petite fonction locale récursive :

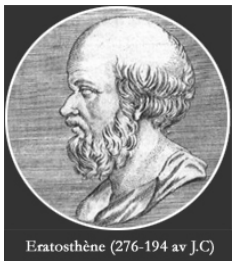
a. Et encore, nous aurions pu faire mieux comme nous allons le voir tout de suite après.

```
Fonction test1_tmp(t,n : entiers) : booléen
Si t * t > n Alors
    Retourner Vrai
Sinon
    Si rem(n,t) = 0 Alors
        Retourner Faux
    Sinon
        Retourner test1_tmp(t+1,n)
    FinSi
FinSi
```

```
Fonction test1(n : entier) : booléen
Retourner test1_local(2,n)
```

On pourrait déjà diviser facilement le nombre de tests par 2, voire plus...

3 3 2 Crible d'Ératosthène



Né 276 années avant JiCé, directeur de la Bibilothèque d'Alexandrie, on lui doit aussi une approximation du diamètre de la Terre. Devenu aveugle, il s'est laissé mourir de faim...

Autre problème crucial : comment obtenir une liste des entiers inférieurs à un petit nombre donné n ?
Le principe est ancien puisqu'il est attribué au grec Ératosthène
On écrit les entiers de 2 à n puis on barre les multiples des nombres premiers inférieurs à $\sqrt{n}$.
Les entiers restant sont premiers.
Voici une première solution extrêmement concise avec Haskell (mais peu efficace en fait...)

Haskell
```
crible n = n : filter (\k -> mod k n /= 0) (crible (n + 1))

premiers k = take k (crible 2)

*Main> premiers 25
[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97]
```

3 4 Décomposition des entiers en produit de nombres premiers

Avant d'aller plus loin dans notre exploration, nous avons dit en introduction que les nombres premiers étaient les briques qui nous permettaient de construire tous les autres : il serait bon de le vérifier.

Théorème 5 - 15 Tout entier n supérieur à 2 se décompose en produit fini de nombres premiers.

La démonstration la plus simple (mais pas la plus intéressante) consiste à raisonner par récurrence sur n.
Nous savons que 2 est premier.

Supposons que tout entier inférieur ou égal à n se décompose en produit de facteurs premiers. L'entier suivant $n+1$ admet au moins un diviseur premier p d'après la propriété 5 - 12 page 102. Soit q le quotient de $n+1$ par p.
Si $q = 1$, alors $n+1 = p$ et donc $n+1$ est premier.
Si $q > 1$, nous appliquons l'hypothèse de récurrence à q : q se décompose en produit fini de nombres premiers, et par suite $n+1$ aussi, car $n+1 = p \times q$.

Par exemple, $50 = 2 \times 5^2$: les facteurs premiers ne sont pas forcément distincts. On a donc l'habitude d'écrire la décomposition sous la forme

$$n = p_1^{\alpha_1} \times p_2^{\alpha_2} \times \cdots \times p_r^{\alpha_r}$$

Une petite chose reste à vérifier : cette décomposition est-elle unique ? Elle a l'air de l'être, c'est pourquoi...nous l'admettrons :

Théorème 5 - 16

Tout entier n supérieur à 2 admet une et une seule (à l'ordre près des termes) décomposition en produit fini de nombres premiers

Bon, nous en savons assez pour revenir au problème de recherche de l'inverse modulaire d'un entier qui nous ouvrira les portes de la cryptographie.

4 Éléments inversibles de Z/nZ

4 1 Anneaux et corps

Définition 5 - 13

Anneau
Soit A un ensemble muni de deux lois $\cdot$ et $+$ qu'on nommera addition et multiplication.
On dit que $(A, +, \cdot)$ est un anneau si, et seulement si :

- $(A, +)$ est un groupe commutatif ;
- la multiplication est associative ;
- la multiplication est distributive sur l'addition.

Si la multiplication est commutative, alors l'anneau est dit *commutatif*.
Si la multiplication admet un élément neutre, l'anneau est dit *unitaire*.
Les éléments d'un anneau unitaire qui admettent un symétrique par la multiplication sont dits *inversibles*. On note A^* l'ensemble des éléments inversibles de A.

Avez-vous déjà rencontré un anneau dans ce cours ?
Dans un anneau, est-ce que tout élément est inversible ?
Vous démontrerez facilement en TD le théoremm suivant :

Théorème 5 - 17

$\left((\mathbb{Z}/n\mathbb{Z})^*, \cdot\right)$ est un groupe.

Définition 5 - 14

Corps
Un corps est un anneau commutatif unitaire dans lequel tout élément non nul est inversible.

En anglais, le terme mathématique pour désigner les corps est *field*.
Voici un bon théorème que vous pourrez démontrer en TD :

Théorème 5 - 18

Deux théorèmes en un...

- Les seuls éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$ sont les éléments premiers avec n.
- Si n est premier, $\mathbb{Z}/n\mathbb{Z}$ est un corps et on le note dans ce cas $\mathbb{F}_n$.

Définition 5 - 15

Fonction indicatrice d'Euler

On note $\varphi(n)$ le nombre d'éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$. C'est donc aussi le cardinal (on dit aussi l'*ordre*) du groupe $\left((\mathbb{Z}/n\mathbb{Z})^*, \cdot\right)$.

Cette fonction joue un rôle important en cryptographie, notamment dans le système RSA.

4 2 Comment calculer l'inverse modulaire d'un entier ?

C'est un problème important qui intervient souvent en cryptographie.
Avec les notations habituelles, le but est de trouver $y \in \{0, 1, \ldots, n-1\}$ tel que $x \cdot y \equiv 1 (\mathrm{mod}\, n)$.

$$\begin{aligned}
\overline{x}^n \text{ est inversible dans } \mathbb{Z}/n\mathbb{Z} &\Leftrightarrow \exists \overline{y}^n \in \mathbb{Z}/n\mathbb{Z} \quad | \quad \overline{x}^n \cdot \overline{y}^n = \overline{1}^n \\
&\Leftrightarrow \exists y \in \{0, 1, \ldots, n-1\} \quad | \quad x \cdot y \equiv 1 (\mathrm{mod}\, n) \\
&\Leftrightarrow \exists (y, k) \in \{0, 1, \ldots, n-1\} \times \mathbb{Z} \quad | \quad xy = 1 + kn \\
&\Leftrightarrow \exists (y, k) \in \{0, 1, \ldots, n-1\} \times \mathbb{Z} \quad | \quad xy - kn = 1 \\
&\Leftrightarrow x \wedge n = 1
\end{aligned}$$

Pouvez-vous justifier cette série d'équivalences ? Dans quelle mesure nous donne-t-elle un moyen de déterminer y ? Nous reparlerons de tout cela en TD...
Nous démontrerons également le théorème suivant :

Théorème 5 - 19

Théorème d'Euler

$$a \wedge n = 1 \Longrightarrow a^{\varphi(n)} \equiv 1 (\mathrm{mod}\, n)$$

En quoi peut-il nous aider à déterminer l'inverse modulaire d'un entier ? Le problème, c'est que le calcul de $\varphi(n)$ peut s'avérer compliqué...
Les propriétés suivantes vont nous aider :

Théorème 5 - 20

Propriétés de $\varphi(n)$

- Si p est premier, alors $\varphi(p) = p - 1$;
- si $m \wedge n = 1$ alors $\varphi(mn) = \varphi(m)\varphi(n)$ (admis) ;
- Si p est premier, $\varphi(p^n) = p^n - p^{n-1} = p^n \left(1 - \frac{1}{p}\right)$;
- $\varphi(n) = n \prod_{p \in \mathcal{P}_n} \left(1 - \frac{1}{p}\right)$ avec $\mathcal{P}_n$ l'ensemble des diviseurs premiers de n.

C'est la dernière propriété qui nous permettra de calculer $\varphi(n)$ avec Haskell.

4 3 Petit théorème de Fermat

Pierre de Fermat (1601-1665)

Nous avons mis au point avec le modèle du crible d'Ératosthène une méthode permettant de tester si un entier est premier ou non. Cela marche assez bien pour des petits nombres, mais cette méthode devient impraticable s'il s'agit de tester un entier d'une centaine de chiffres. Nous allons nous occuper dans cette section de deux problèmes imbriqués : d'une part trouver des méthodes permettant de vérifier rapidement si un nombre est premier et d'autre part réfléchir à d'éventuelles méthodes permettant de « fabriquer » des nombres premiers aussi grands que l'on veut.
Nous avons besoin pour cela d'un dernier résultat : le fameux « petit théorème de Fermat ».S'il est qualifié de petit, c'est qu'une conjecture célèbre du même Fermat, restée indémontrée pendant des siècles, s'est accaparée le titre de *Grand Théorème de* Fermat.

$$x^n + y^n = z^n \quad \text{n'a pas de solution dans } \mathbb{N}^3 \quad \text{pour } n > 2$$

Dans la marge d'un manuscrit, Fermat prétendait en avoir trouvé la démonstration mais manquer de place pour l'écrire. Il a pourtant fallu attendre 1994 pour qu'Andrew Wiles le démontre

en utilisant des outils surpuissants : l'entêtement d'une multitude de chercheurs a abouti à la démonstration de ce théorème, mais surtout a permis de développer d'importants outils en cryptographie donc en sécurité informatique.
Voici celui qui nous sera utile qui n'est en fait qu'une conséquence du théorème d'EULER...

Théorème 5 - 21

Petit théorème de Fermat

— Si p est premier et ne divise pas a alors $a^{p-1} \equiv 1 (\mathrm{mod}\, p)$;

— Si p est premier, $a^p \equiv a (\mathrm{mod}\, p)$.

5 EXERCICES

5 1 Structures mères

Exercice 5 - 1

Donnez des exemples de couples $(E, \star)$ avec $\star$ une application définie sur $E \times E$ telle que $(E, \star)$ ne soit pas un magma.

Exercice 5 - 2

Les tables ci-dessous définies sur $E = \{a, b, c\}$ définissent-elles des monoïdes ? Unitaires ?

$\star$	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

$\perp$	a	b	c
a	a	c	c
b	b	c	a
c	c	a	a

Notons $A = \{a, b, c\}$. Montrez que $(A, \star)$ a une structure de groupe. Comparer ce groupe avec $(\mathbb{Z}/3\mathbb{Z}, \overline{+})$ et $\left(\{1, e^{2\pi/3}, e^{-2\pi/3}\}, \times\right)$.

Exercice 5 - 3

Soit $A = \{a_1, a_2, \ldots, a_p\}$ un alphabet. Nous avons vu au chapitre précédent que $A^\star$ muni de la concaténation des chaînes a une structure de monoïde (unitaire ?). Les ensembles suivants sont-ils des sous-monoïdes de $A^\star$ muni de la concaténation ? Unitaires ?

1. L'ensemble des chaînes de longueur paire ;
2. L'ensemble des chaînes de longueur impaire ;
3. $\{(a_1 a_2)^n \mid n \in \mathbb{N}\}$;
4. $\{a_1^n a_2^n \mid n \in \mathbb{N}\}$;
5. L'ensemble des chaînes ne contenant que a_1 et a_2, ceux-ci apparaissant un nombre égal de fois.

5 2 Division euclidienne

Exercice 5 - 4 Nombre de chiffres

Déterminez une fonction **nb_chiffres n** qui renvoie le nombre de chiffres de l'écriture décimale de n.

Exercice 5 - 5 Décomposition en base quelconque

Déterminez une fonction **base b n** qui renvoie la liste des chiffres de l'écriture normalisée de n en base b.

Haskell

```
*Main> base 2 97
[1, 1, 0, 0, 0, 0, 1]
```

Exercice 5 - 6 Persistance d'un entier

Déterminez une fonction récursive qui renvoie la liste des chiffres de l'écriture décimale d'un nombre.
Un entier naturel n étant donné, on calcule le produit **prod n** de ses chiffres dans son écriture en base 10 puis le produit des chiffres de **prod n** et on recommence ainsi l'application de **prod** jusqu'à obtenir un chiffre entre 0 et 9. Le nombre minimal de fois où on applique **prod** pour transformer n en un chiffre entre 0 et 9 est appelé la *persistance* de n. Par exemple, la persistance de 9 est égale à 0, celle de 97 est égale à 3 car **prod**$(97) = 9 \cdot 7 = 63$, **prod**$(63) = 6 \cdot 3 = 18$, **prod**$(18) = 1 \cdot 8 = 8$. et celle de 9575 est égale à 5.
On voudrait connaître le plus petit entier naturel de persistance 5.

Exercice 5 - 7 Écriture littérale

Déterminez une fonction qui renvoie l'écriture littérale d'un nombre dont l'écriture décimale est comprise entre 1 et 999.

Exercice 5 - 8 Problème de calendrier

Connaissant une date, à quel jour de la semaine correspond-elle ? L'allemand ZELLER a proposé une formule en 1885.
On note m le numéro du mois à partir de janvier, s le numéro du siècle, a le numéro de l'année dans le siècle s, j le numéro du jour. Pour 1492, $s = 14$ et $a = 92$.
En fait, m doit être modifié selon le tableau suivant :

Mois	Jan.	Fév.	Mars	Avril	Mai	Juin	Juil.	Août	Sep.	Oct.	Nov.	Déc.
Rang	13*	14*	3	4	5	6	7	8	9	10	11	12

Les mois marqués d'une astérisque comptent pour l'année précédente.
Le nom du jour (0 correspondant à samedi) est alors déterminé en prenant le reste (positif !) dans la division par 7 de nj donné par la formule :

$$nj = j + a + \text{quo}(a, 4) + \text{quo}(26(m + 1), 10) + \text{quo}(s, 4) - 2s$$

Ceci est valable à partir du 15 octobre 1582, date à laquelle le pape Grégoire XIII a modifié le calendrier : on est passé du jeudi 4 octobre 1582 au vendredi 15 octobre 1582 (en Grande-Bretagne il a fallu attendre 1752, au Japon en 1873, en Russie 1918, en Grèce 1924).
L'année est alors de 365 jours, sauf quand elle est bissextile, i.e., divisible par 4, sauf les années séculaires (divisibles par 100), qui ne sont bissextiles que si divisibles par 400.
Avant le changement, il faut remplacer $\text{quo}(s, 4) - 2s$ par $5 - s$.
Pour les démonstrations, voir l'article original de ZELLER disponible (en allemand...) à l'adresse : http://www.merlyn.demon.co.uk/zel-86px.htm
Quel jour est né le petit Jésus ? Quel jour a été prise la Bastille ?

Exercice 5 - 9 Critères de divisibilité

En travaillant modulo le bon entier, démontrer les critères de divisibilité usuels par 2, 3, 4, 5, 9, 10 et 11.
On rappelle que l'écriture en base 10 d'un nombre n est définie par la donnée de l'unique famille $(a_0, a_1, \ldots, a_k)$ vérifiant :

$$n = a_k 10^k + a_{k-1} 10^{k-1} + \cdots + a_2 10^2 + a_1 10 + a_0$$

avec $a_k \neq 0$ et $0 \leqslant a_1 < 10$ pour tout $i \in \{0, 1, .., k\}$.

Exercice 5 - 10 Nombre de diviseurs

Déterminer une fonction qui calcule le nombre de diviseurs positifs d'un entier naturel donné.

Exercice 5 - 11 Liste des diviseurs

Déterminer une fonction récursive qui calcule la liste des diviseurs positifs d'un entier naturel donné.

*

Exercice 5 - 12 Nombres parfaits

Un nombre parfait est un nombre entier n strictement supérieur à 1 qui est égal à la somme de ses diviseurs (sauf n bien sûr) ou, de manière équivalente, c'est un nombre tel que son double est égal à la somme de ses diviseurs, lui-même compris.
Donner la liste des entiers parfaits inférieurs à 100 000.

On connaît une quarantaine de nombres parfaits actuellement, tous pairs. On ne sait toujours pas s'il en existe des impairs.
Cependant, EUCLIDE avait déjà découvert la propriété suivante :
« Lorsque la somme d'une suite de nombres doubles les uns des autres est un nombre premier, il suffit de multiplier ce nombre par le dernier terme de cette somme pour obtenir un nombre parfait. »
Qu'en pensez-vous ? Quel est la rapport avec $2^p - 1$ et $2^{p-1}(2^p - 1)$? Écrivez les nombres parfaits obtenus en binaire : joli, non ?

5 3 PGCD

Exercice 5 - 13 PGCD

Programmez sur Haskell l'algorithme récursif d'EUCLIDE I.

Exercice 5 - 14 Nombres premiers entre eux

Déterminez une fonction qui teste si deux entiers sont premiers entre eux.

Exercice 5 - 15 Bézout

Programmez sur Haskell l'algorithme récursif d'EUCLIDE II qui calcule des coefficients de BÉZOUT et le PGCD de deux nombres.

Exercice 5 - 16 Inverse modulaire

Déterminez une fonction qui calcule l'inverse d'un entier a modulo un entier n. Vous ferez attention aux exceptions.

5 4 Nombres premiers

Exercice 5 - 17 Test naïf

Programmez le test naïf de primalité vu en cours. Modifiez votre programme pour réduire le nombre de tests effectués sachant qu'un entier pair ou multiple de 3 autre que 2 et 3 n'est pas premier...

Exercice 5 - 18 Décomposition

Créez une fonction **`decompo n`** qui renvoie la liste des diviseurs premiers de n.

5 5 Complexité et relation de domination

Voici un autre standard du vocabulaire geek :

> *Brooks's Law [prov.]*
> *« Adding manpower to a late software project makes it later » – a result of the fact that the expected advantage from splitting work among N programmers is $O(N)$, but the complexity and communications cost associated with coordinating and then merging their work is $O(N^2)$*
>
> in « The New Hacker's Dictionnary »
> http://outpost9.com/reference/jargon/jargon_17.html#SEC24

Les notations de LANDAU(1877-1938) ont en fait été crées par Paul BACHMANN(1837-1920) en 1894, mais bon, ce sont tous deux des mathématiciens allemands.
Par exemple, si l'on considère l'expression :

$$f(n) = n + 1 + \frac{1}{n} + \frac{75}{n^2} - \frac{765}{n^3} + \frac{\cos(12)}{n^{37}} - \frac{\sqrt{765\,481}}{n^{412}}$$

Quand n est « grand », disons 10 000, alors on obtient :

$$f(10\,000) = 10\,000 + 1 + 0,0001 + 0,00000000075 - 0,0000000000000765 + \text{ peanuts}$$

Tous les termes après n comptent pour du beurre quand n est « grand ». Donnons une définition pour plus de clarté :

Définition 5 - 16

« Grand O »
Soit f et g deux fonctions de $\mathbb{N}$ dans $\mathbb{R}$. On dit que f est un « grand O » de g et on note $f = O(g)$ ou $f(n) = O(g(n))$ si, et seulement si, il existe une constante strictement positive C telle que $|f(n)| \leqslant C|g(n)|$ pour tout $n \in \mathbb{N}$.

Dans l'exemple précédent, $\frac{1}{n} \leqslant \frac{1}{1} \times 1$ pour tout entier n supérieur à 1 donc $\frac{1}{n} = O(1)$.
De même, $\frac{75}{n^2} \leqslant \frac{75}{1^2} \times 1$ donc $\frac{75}{n^2} = O(1)$ mais on peut dire mieux : $\frac{75}{n^2} \leqslant \frac{75}{1} \times \frac{1}{n}$ et ainsi on prouve que $\frac{75}{n^2} = O\left(\frac{1}{n}\right)$.
En fait, un grand O de g est une fonction qui est au maximum majorée par un multiple de g.
Point de vue algorithmique, cela nous donne un renseignement sur la complexité au pire.
On peut cependant faire mieux si l'on a aussi une minoration.
C'est le moment d'introduire une nouvelle définition :

Définition 5 - 17

« Grand Oméga »
Soit f et g deux fonctions de $\mathbb{R}$ dans lui-même. On dit que f est un « grand Oméga » de g et on note $f = \Omega(g)$ ou $f(n) = \Omega(g(n))$ si, et seulement si, il existe une constante strictement positive C telle que $|f(n)| \geqslant C|g(n)|$ pour tout $n \in \mathbb{N}^*$.

Remarque

Comme Ω est une lettre grecque, on peut, par esprit d'unification, parler de « grand omicron » au lieu de « grand O »...

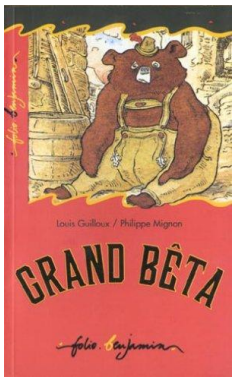
Remarque

$f = \Omega(g) \iff g = O(f)$...

Si l'on a à la fois une minoration et une majoration, c'est encore plus précis et nous incite à introduire une nouvelle définition :

Définition 5 - 18

« Grand Théta »
$f = \Theta(g) \iff f = O(g) \quad \wedge \quad f = \Omega(g)$



Voici maintenant une petite table pour illustrer les différentes classes de complexité rencontrées habituellement :

coût \ n	100	1000	10^6	10^9
$\log_2(n)$	≈ 7	≈ 10	≈ 20	≈ 30
$n \log_2(n)$	≈ 665	$\approx 10\,000$	$\approx 2 \cdot 10^7$	$\approx 3 \cdot 10^{10}$
n^2	10^4	10^6	10^{12}	10^{18}
n^3	10^6	10^9	10^{18}	10^{27}
2^n	$\approx 10^{30}$	$> 10^{300}$	$> 10^{10^5}$	$> 10^{10^8}$

Gardez en tête que l'âge de l'Univers est environ de 10^{18} secondes...

5 6 Exercices « papier-crayon »

Exercice 5 - 19 Complexité de l'algorithme d'Euclide et nombre d'or...

On suppose dans toute la suite que a et b sont deux entiers naturels tels que $a \geqslant b$.

1. Rappeler le principe de l'algorithme et de sa preuve.
2. Soit $r_n = a \wedge b$ avec les notations habituelles : $r_0 = b$ puis $r_1 = a \bmod b$, $r_2 = r_1 \bmod r_0$ et plus généralement :
$$r_{k+1} = r_k q_k + r_{k-1}, \quad 0 \leqslant r_{k-1} < r_k$$
Montrez que pour tout entier $k \in \{1, 2, \ldots, n\}$, on a $q_k \geqslant 1$ puis que $q_n \geqslant 2$.

3. Soit $\Phi = \frac{1+\sqrt{5}}{2}$ le fameux nombre d'or. Comparez Φ et $1 + \frac{1}{\Phi}$.
4. Montrez par récurrence que $r_k \geqslant \Phi^{n-k}$ pour tout entier $k \in \{0, 1, ..., n\}$.
5. Déduisez-en que $b \geqslant \Phi^n$ puis que le nombre d'appels récursifs de l'algorithme d'Euclide est au maximum de $\frac{\ln b}{\ln \Phi}$.

Exercice 5 - 20 Nombre de diviseurs

Déterminez le nombre de diviseurs de 2^n.

Exercice 5 - 21 Opérations sur les O

On suppose que f, g, F, G sont des fonctions de $\mathbb{N}$ dans $\mathbb{R}_+$ et que $f = O(F)$ et $g = O(G)$. Rappelez la définition des O et trouvez φ et ψ telles que $f + g = O(\phi)$ et $fg = O(\psi)$.

Exercice 5 - 22 Bases

Écrire 355 en base 2 et en base 16.

Exercice 5 - 23 Bibinaire

Boby Lapointe, célèbre chanteur français, était aussi mathématicien à ses heures. Ayant trouvé le code binaire trop compliqué à utiliser, il inventa le code... bibinaire. Il suffit de remplacer les chiffres par des lettres. On commence par couper le nombre écrit en binaire en paquets de 2. S'il y a un nombre impair de chiffres, on rajoute un zéro à gauche, ce qui ne modifie pas la valeur de nombre. On commence par le premier groupe de deux chiffres le plus à droite. On remplace 00 par O, 01 par A, 10 par E, 11 par I.
Puis on prend le paquet de deux chiffres suivants en se déplaçant de droite à gauche. On remplace 00 par H, 01 par B, 10 par K, 11 par D.
Pour le paquet suivant, on recommence avec les voyelles. S'il y a encore un groupe, on remplace par une consonne, etc.
Écrivez les nombres de 0 à 25 en bibinaire. Pourquoi Boby a-t-il finement choisi le H ?
Écrivez la table de 3 en bibinaire.
Écrivez les nombres de 255 à 257 en bibinaire. Quel est la base du bibinaire ? Écrivez 355 en bibinaire.
Écrivez KEKIDI et HEHOBOBI en base 10.
Pourquoi est-il pratique d'écrire les nombres en base 2 avec un nombre de chiffres multiple de 4 avant de les traduire en bibinaire ?

Exercice 5 - 24 Groupe

Soit $A = \frac{1}{3}\begin{bmatrix} 0 & -2 & -2 \\ 2 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$.

1. Calculer A^2, A^3 et A^4.
2. Montrez que $\{A, A^2, A^3, A^4\}$ est un groupe pour le produit matriciel de $\mathcal{M}_3(\mathbb{R})$.
3. (INFO 2) On appelle $GL_3(\mathbb{R})$ l'ensemble des matrices de taille 3 à coefficients réels inversibles. Est-ce que $(GL_3(\mathbb{R}), \times)$ est un groupe ? Est-ce que le groupe étudié précédemment est un sous-groupe de $(GL_3(\mathbb{R}), \times)$?

Exercice 5 - 25 Puissance

Calculez à la main en un minimum d'opérations les deux derniers chiffres de l'écriture en base 10 de 7^{73} puis de 6^{73}

Exercice 5 - 26 Indicatrice d'Euler

Un vieux théorème publié en 1247 par le mathématicien chinois Qin Jiushao mais utilisé dès le troisième siècle après JC et appelé communément *théorème chinois* en référence au « Sunzi suanjing », permet de montrer un résultat primordial en cryptographie :

$$\textit{Si } m \textit{ et } n \textit{ sont premiers entre eux, alors } \varphi(mn) = \varphi(m)\varphi(n) \ .$$

Nous admettrons ce résultat.

1. Soit p un nombre premier et n un entier naturel non nul.
   Prouvez que $\varphi(p^n) = p^n - p^{n-1}$ puis que, pour tout entier naturel non nul k,
   $$\varphi(k) = k \prod_{p|k} \left(1 - \frac{1}{p}\right)$$
2. Calculez $\varphi(200)$ et $\varphi(1000)$.
   Trouvez, DE TÊTE, les trois derniers chiffres de :
   $$789587546378637837837834545364653897897 7^{20858271459145855145513513513514745409825825 8001}$$
3. Comment utiliser le théorème d'EULER pour calculer plus simplement $m^e \pmod n$ lorsque m et n sont premiers entre eux ?
4. Cela est très important pour le RSA. Soit un couple (p, q) d'entiers premiers. On note $n = pq$ leur produit. Soit m un nombre plus petit que p et q.
   Comment calculer plus simplement $m^e \pmod n$? Quelle condition doit remplir e pour qu'il existe un entier d tel que $(m^e)^d \equiv m \pmod n$?

Exercice 5 - 27 Lemme de Gauß

Démontrez le lemme suivant :

Théorème 5 - 22

Lemme de Gauß

$$\forall (a, b, c) \in \mathbb{Z}^3, (a|bc \wedge \text{PGCD}(a, b) = 1) \Rightarrow a|c$$

Exercice 5 - 28 Restes chinois

Voici une ré-écriture du théorème chinois :

Théorème 5 - 23

Théorème du reste chinois

Soit $m_1, m_2, ..., m_n$ des entiers positifs premiers entre eux deux à deux. Le système :

$$\begin{array}{rcl} x & \equiv & a_1[x_1] \\ x & \equiv & a_2[x_2] \\ \vdots & \vdots & \vdots \\ x & \equiv & a_n[x_n] \end{array}$$

admet une unique solution unique x modulo $m = m_1 \times m_2 \times \cdots \times m_n$.

Occupons-nous de démontrer qu'il existe bien une solution x du problème modulo m. Il restera à démontrer son unicité.
Considérons les entiers $M_k = m/m_k$: montrez qu'il existe un entier y_k tel que $M_k y_k \equiv 1[m_k]$.
Montrez ensuite que $x = a_1 M_1 y_1 + \cdots + a_n M_n y_n$ est une solution du problème.

Exercice 5 - 29 Pirates chinois

15 pirates chinois se partagent un butin constitué de pièces d'or. Mais une fois le partage (équitable) effectué, il reste 3 pièces. Que va-t-on en faire ? La discussion s'anime. Bilan : 8 morts. Les 7 survivants recommencent le partage, et il reste cette fois-ci 2 pièces ! Nouvelle bagarre à l'issue de laquelle il ne reste que 4 pirates. Heureusement, ils peuvent cette fois-ci se partager les pièces sans qu'il n'en reste aucune. Sachant que 32 Tsing-Tao (bière chinoise) coûtent une pièce d'or, combien (au minimum) de Tsing-Tao pourra boire chaque survivant ?

Exercice 5 - 30 Haskell chinois

Créez une fonction **chinois** de signature :

Haskell
```
chinois :: [Classe] -> Classe
```

qui résout le problème des restes chinois.
Par exemple :

Haskell
```
*Main> chinois [3%15 , 2%7 , 0%4]
408%420
```

Exercice 5 - 31 Opération sur les grands nombres version 1

Soit $(x_1, x_2, ..., x_n)$ une liste de n nombres entiers premiers entiers deux à deux et soit m leur produit.
Démontrez que tout entier a positif inférieur à m peut être représenté de manière unique par un n-uplet :

Haskell
```
[a%m_1 ; a%m_2 ; ... ; a%m_n]
```

avec les notations Haskell habituelles.
Soit $b = [99; 98; 97; 95]$. Décomposez 123684 et 413456 dans cette « base » puis effectuez la somme des quadruplets obtenus. Que vous inspire cette méthode ?
Proposez des fonctions Haskell effectuant ces calculs automatiquement.

Exercice 5 - 32 Ordre d'un élément dans un groupe fini

1. Comment justifier que pour tout élément g d'un groupe fini G, il existe un plus petit entier n tel que $g^n = 1_G$? On appelle n l'ordre de g.
2. Déterminez l'ordre des éléments de $(\mathbb{Z}/9\mathbb{Z})^*$.
3. Calculez $2^{120000000000300000000060000000200004000020} \pmod 7$.

Exercice 5 - 33 Permutations

Définition 5 - 19

Permutation
Soit E un ensemble. Une permutation de E est une application bijective de E sur E. On note $\mathfrak{S}(E)$ l'ensemble des permutations de E.

Généralement, on note les permutations sous forme d'une matrice où la première ligne correspond aux éléments de E et où la deuxième ligne correspond aux images des éléments de E.
Par exemple :

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 2 & 5 & 4 \end{pmatrix}$$

est une permutation de $E = \{1, 2, 3, 4, 5\}$.

Définition 5 - 20

Soit $n \in \mathbb{N}$, alors $\mathfrak{S}_n$ désigne l'ensemble des permutations de $\{1, 2, 3, ..., n\}$.

Théorème 5 - 24

Groupe des permutations
L'ensemble $\mathfrak{S}(E)$ des permutations sur un ensemble E, muni de la loi $\circ$ de composition des applications, a une structure de groupe.

Théorème 5 - 25

Nombre de permutations
Le groupe $\mathfrak{S}_n$ est d'ordre $n!$.

1. Démontrez le théorème 5 - 24 page ci-contre.
2. Démontrez le théorème 5 - 25. Vous pourrez procéder par récurrence. Pour l'hérédité, distinguez les permutations qui envoient 1 sur un élément quelconque x.
3. Décrivez explicitement les groupes $\mathfrak{S}_2$ et $\mathfrak{S}_3$.
4. En cryptographie, on travaillera sur l'alphabet $E = \{0,1\}^n$ et on effectuera souvent des *permutations de bits* : seuls les positions des bits sont permutées.
   On formalise ces permutations ainsi :
   $$f : \begin{array}{ccc} \{0,1\}^n & \to & \{0,1\}^n \\ b_1 b_2 \cdots b_n & \mapsto & b_{\pi(1)} b_{\pi(2)} \cdots b_{\pi(n)} \end{array}$$
   avec $\pi \in \mathfrak{S}_n$.
   Une *permutation circulaire gauche* « décale » les éléments d'un nombre fixé de « rangs ». Par exemple, voici une permutation circulaire gauche :
   $$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 1 & 2 \end{pmatrix}$$
   Décrire $\pi(k)$ dans le cas d'une permutation circulaire de bits de i rangs vers la gauche.
   On définit de même des permutations circulaires droite.
   Combien y a-t-il de permutations circulaires dans $\mathfrak{S}_n$?

Exercice 5 - 34 Chiffrement par blocs

Commençons par une définition :

Définition 5 - 21

Cryptosystème (ou système de chiffrement ou chiffre)
Un cryptosystème est un quintuplet $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ tel que :
- L'ensemble $\mathcal{P}$ est l'espace des messages en clair (*plaintext* en anglais) ;
- L'ensemble $\mathcal{C}$ est l'espace des messages chiffrés (*cyphertext* en anglais) ;
- L'ensemble $\mathcal{K}$ est l'espace des clés (*key*) ;
- $\mathcal{E}$ est la famille des fonctions de chiffrement, qui vont de $\mathcal{P}$ dans $\mathcal{C}$;
- $\mathcal{D}$ est la famille des fonctions de déchiffrement, qui vont de $\mathcal{C}$ dans $\mathcal{P}$;

Pour chaque élément $e \in \mathcal{K}$, il existe un élément $d \in \mathcal{K}$ tel que, pour tout message clair m de $\mathcal{P}$, il existe $D_d \in \mathcal{D}$ et $E_e \in \mathcal{E}$ telles que $D_d(E_e(m)) = m$. Les fonctions D_d et E_e sont des applications injectives. Il est entendu que d doit rester secret...

1. On travaille sur des mots formés des 26 minuscules non accentuées de notre alphabet latin et d'une espace qui sont modélisés par les entiers de $E = \{0, 1, 2, ..., 26\}$.
   Dans ce contexte, on remplace une « lettre » x par $kx \pmod{27}$ avec $k \in E$. Définit-on ainsi un cryptosystème ?
2. Un cryptosystème est un *chiffrement par blocs* si $\mathcal{P} = \mathcal{C} = A_n$ avec A_n l'ensemble des mots de l'alphabet A de longueur n. Est-ce que le chiffrement ROT-13 est un chiffrement par blocs ?
3. Pourquoi les fonctions de chiffrement d'un chiffrement par blocs sont des permutations ?

Depuis 2001, le chiffrement par blocs « officiel » est l'AES qui opère par blocs de 128 bits. Une présentation de l'algorithme de chiffrement associé, RIJNDAEL, et de sa résistance aux attaques est présentée ici : http://www.cryptis.fr/assets/files/Canteaut-25ans-Cryptis.pdf.

Exercice 5 - 35 Mode ECB

ECB : *Electronic CodeBook*. C'est le plus simple...et le plus vulnérable des modes de chiffrement par blocs.
Voyons sur un exemple. On considère une chaîne de bits quelconque que l'on découpe en blocs de longueur fixe, par exemple 7 (plus pratique pour ensuite utiliser l'ASCII : pourquoi ?). On rajoute éventuellement des zéros en bout de chaîne.
Considérons $m = 1001001111011$. On rajoute un 0 en bout de chaîne :

$$m' = 1001001\ 1110110 = \beta_1\beta_2$$

avec $\beta_1 = 1001001$ et $\beta_2 = 1110110$.

On choisit une clé de chiffrement dans $\mathfrak{S}_7$ car $\mathcal{K} = \mathfrak{S}_7$: $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 1 & 4 & 2 & 6 & 5 & 7 \end{pmatrix}$

Alors $E_\pi(\beta_1) = 0110001$ et $E_\pi(\beta_2) = 1101110$.

1. Quelle est la principale faiblesse du mode ECB ?
2. Chiffrez « maman » avec ce cryptosystème en transformant la chaine de lettres en chaînes de bits constituée des codes ASCII de ses caractères. Le tableau ASCII standard est-il satisfaisant ? Comment y remédier ?
3. Quelle est la fonction de déchiffrement associée à l'exemple précédent ?
4. Déchiffrer 10110111001111 sachant que la clé de chiffrement est $k = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 1 & 7 & 6 & 5 & 4 \end{pmatrix}$
5. Déchiffrez « papa » sachant que c'est le cryptosystème de la question 2. qui a été employé.

Exercice 5 - 36 Mode CBC

CBC : *Cipher-Block Chaining*. Ce mode a été inventé par IBM en 1976. Pour éviter la faiblesse de l'ECB, les blocs sont maintenant chaînés : chaque bloc en clair est « XORé » ou « OUEXé » avec le bloc crypté précédent avant d'être crypté lui-même.
Pour le premier bloc, on utilise un *vecteur d'initialisation* public. Avec les notations habituelles, on a donc

$$c_i = E_k(m_i \oplus c_{i-1}), \quad c_0 = \overrightarrow{v_0}$$

1. Quel est le lien entre OUEX et $\mathbb{F}_2$?
2. Cryptez l' exemple traité avec ECB en prenant $\overrightarrow{v_0} = 1010101$.
3. Faites de même avec « Maman ».
4. Donnez une formule de déchiffrement. Déchiffrer le cryptogramme 101101110011111 sachant que la clé est $k = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$ et que $\overrightarrow{v_0} = 001$.
5. Déchiffrez « Papa » sachant que c'est le cryptosystème de la question 2. qui a été employé.

Ce mode est efficace mais nécessite l'utilisation de fonctions de chiffrement et de déchiffrement différentes ce qui peut ralentir la procédure.

Exercice 5 - 37 Mode CFB

CFB : *Cipher-FeedBack*. C'est un mode dérivé de CFB qui est utilisé par OpenPGP, format pour l'échange sécurisé de données (paiements sécurisés par exemple) largement utilisé actuellement mais est susceptible d'être attaqué, même s'il est très difficile de mettre en œuvre concrètement cette attaque :

http://www.cert-ist.com/fra/ressources/Publications_ArticlesBulletins/Autres/FailledanslechiffrementCFBdOpenPGP/

Le mode CFB utilise un registre de décalage de taille r inférieure à celle de la clé.
On a besoin d'un vecteur d'initialisation $\overrightarrow{v_0}$. Le message clair est découpé en blocs de longueur r. Ensuite on procède comme suit, sachant que les m_j sont les blocs en clair de longueur r :

— $t_0 = \overrightarrow{v_0}$;

— $s_j = E_k(t_j)$;
— g_j est la chaîne constituée des r bits les plus à gauche de s_j. Quelle est l'opération arithmétique associée ?
— $c_j = m_j \oplus g_j$;
— $t_j = (2^r t_{j-1} + c_{j-1}) \bmod 2^n$ (attention ! Il faut travailler en base 2).

Le déchiffrement fonctionne de manière identique en échangeant les rôles de m_j et c_j à la quatrième étape : démontrez-le !

1. Chiffrez « *Papa is cool* » sachant que la clé est $k = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 3 & 2 \end{pmatrix}$, que $\overrightarrow{v_0} = 0101$ et que $r = 3$.

2. Déchiffrer le cryptogramme 101101110011111 sachant que la clé est $k = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$, que $r = 2$ et que $\overrightarrow{v_0} = 001$.

5.7 Exponentiation rapide

Exercice 5 - 38 Complexité

Voici deux algorithmes :

```
Fonction algo1( a,n : entiers naturels ) : entier naturel
Début
    Si n=0 Alors
        Retourner 1
    Sinon
        Retourner a*algo1(a,n-1)
    FinSi
Fin
```

```
Fonction algo2( a,n : entiers naturels ) : entier naturel
Variable
    i,r : entiers
Début
    r ← 1
    Pour i variantDe 1 Jusque n Faire
        r ← a*r
    FinPour
    Retourner r
Fin
```

Que calculent-ils ? Quelle est leur complexité ?
En voici un troisième :

```
Fonction algo3( a,n : entiers naturels ) : entier naturel
Début
    Si n=0 Alors
        Retourner 1
    Sinon
        Si n=1 Alors
            Retourner a
        Sinon
            Si n est pair Alors
                Retourner algo3(a*a,n/2)
            Sinon
                Retourner a*algo3(a*a,(n-1)/2)
            FinSi
        FinSi
    FinSi
Fin
```

Que calcule-t-il ? Quelle est sa complexité ?
Comparez le temps de calcul de algo3(1 000 000 000) et de algo1(1 000 000 000) sur un ordinateur équipé d'un processeur 3GHz effectuant une opération arithmétique élémentaire en 100 cycles.
Programmer **`algo3`** sur Haskell. On l'appellera **`prap a n`**.
Calculer 2^{10}, 2^{29}, 2^{30}, $2^{30}-1$, $2^{30}+1$.

ANNEXE

A

Raccourcis emacs

DÉPLACEMENTS	
C-v	Pagedown
M-v	Pageup
C-p	Ligne du dessus (previous line)
C-n	Ligne du dessous (next line)
C-b	Caractère précédent (back)
C-f	Caractère suivant (next)
M-b	Mot précédent
M-f	Mot suivant
C-a	Début de ligne
C-e	Fin de ligne
M-a	Début de phrase
M-e	Fin de phrase
M-<	Fin de fichier
M->	Début de fichier
C-l	Curseur en milieu de page
C-u C-v	Curseur en haut de page
COMMANDES PRINCIPALES (MENU FICHIER)	
C-x C-f	Ouvrir fichier (tab de complétion)
C-x C-s	Sauvegarder fichier
C-x C-c	Quitter Emacs
C-z	Sortie provisoire (récupération possible par %emacs)
C-g	Arrêt d'une commande (ou sortie d'un sous-menu de commande)
ESC ESC ESC	Idem (mais sortie d'un recursive editing level)
ÉDITION	
C-x u	Annulation (ou **C-_**)
C-x z	Répéter
Suppr	Suppression du caractère avant
C-d	Suppression du caractère après
M-suppr	Suppression du mot avant
M-d	Suppression du mot après
C-k	Couper (kill) jusqu'en fin de ligne
M-k	Couper (kill) jusqu'en fin de phrase
C-x k	Couper (kill) le buffer
C-espace	Marquer (début d'un copier/couper)
C-w	Fin d'un couper (kill)
M-w	Fin d'un coller
M-h	Marque le paragraphe
C-x h	Marque la totalité du buffer
C-x C-espace	Coller (pop) global mark
C-y	Coller (yank du killed)
M-y	Passage entre les différents kills (anciens copier, après un **C-y**)
C-s	Recherche après (suivi de **C-w** recherche le mot sous le curseur)

C-r	Recherche avant (suivi de **C-w** recherche le mot sous le curseur)
M-Shift ù	Remplacer
Espace	Pour remplacer l'occurence suivante
Suppr	Pour passer l'occurence sans la remplacer
!	Pour remplacer toutes les occurences
M-u	Mot en majuscule
M-l	Mot en minuscule
M-c	Mot en capitales
TAMPONS (BUFFERS)	
C-x C-b	Liste des buffers (fichiers ouverts)
C-x s	Sauvegarde les buffers (pose la question)
C-x 0	Supprime cette fenêtre
C-x 1	Supprime les autres fenêtre
C-x 2	Divise la fenêtre en 2 verticalement
C-x 3	Divise la fenêtre en 2 horizontalement
C-x o	Passage d'un écran à l'autre (other)
C-x ^	Aggrandir la fenêtre
C-v	Pagedown dans l'autre fenêtre (ESC **C-v** en cas de non méta)
C-x 4 C-f	nomFichier Ouverture de nomFichier dans une fenêtre en bas
C-x 4 b	Fermeture fenêtre
DIVERS	
C-u chiffre	Itération d'une action (ex : **C-u 5 C-n** Descend de 5 lignes)
C-x	Commande suivi d'un seul caractère
M-x	Commande à partir d'une commande texte (tab de complétion)
C-h ?	Aide générale
C-h c nomCommande	Description de la commande
C-h k nomCommande	Aide sur la commande
C-h f nomFonction	Description d'une fonction
C-h a nom	(Apropos) : liste les commandes contenant le nom
C-h i	Lire les infos (Manuels en-ligne)
ESC !	Commande shell

ANNEXE

B

Module logique en Haskell

Voici quelques fonctions qui pourront vous permettre d'utiliser Haskell comme machine à calculer logique et vérifier les résultats de vos exercices.

Haskell

```haskell
import qualified Data.Map as Dic
import qualified Data.Set as Set

type Atome = Char
type Environnement = Dic.Map Atome Bool

data Formule =
  Faux
  |Atomic Atome
  |Non Formule
  |Et Formule Formule
  |Ou Formule Formule
  |Imp Formule Formule
  |Equiv Formule Formule deriving (Show,Eq)

-- Opérateurs infixes plus pratiques à utiliser
μ x = Atomic x
x & y = Et x y
x  § y = Ou x y
x ==> y = Imp x y
x <==> y = Equiv x y

-- exemple de formule
-- de_morg = Equiv (Non (Et (Atomic 'p') (Atomic 'q')))  (Ou (Non (Atomic 'p')) (Non
    (Atomic 'q')))
de_morg = Non ((μ 'p') & (μ 'q')) <==> (Non (μ 'p')  § Non (μ 'q'))

-- exemple d'environnement
env1 = Dic.fromList[('p',True),('q',False)]

-- évaluation récursive d'une formule sous un environnement
eval :: Formule -> Environnement -> Bool
eval Faux env        = False
eval (Atomic a) env  = env Dic.! a
eval (Non f) env     =  if (eval f env) then False else True
eval (Et f g) env    = if (eval f env) then (eval g env) else False
eval (Ou f g) env = if (eval f env) then True else (eval g env)
eval (Imp f g) env   = eval (Non (f & (Non g))) env
eval (Equiv f g) env = eval ((f & g) §  ((Non f) & (Non g))) env

-- renvoie l'ensemble des atomes présents dans une formule
ens_atomes :: Formule -> Set.Set Atome
ens_atomes Faux        = Set.empty
ens_atomes (Atomic a)  = Set.singleton a
ens_atomes (Non f)     = ens_atomes f
ens_atomes (Et f g)    = Set.union (ens_atomes f ) (ens_atomes g )
ens_atomes (Ou f g)    = Set.union (ens_atomes f ) (ens_atomes g )
ens_atomes (Imp f g)   = Set.union (ens_atomes f ) (ens_atomes g )
ens_atomes (Equiv f g) = Set.union (ens_atomes f ) (ens_atomes g )

-- "dit" la formule comme on l'énonce habituellement
dit :: Formule -> String
dit Faux        = "Faux"
dit (Non Faux)  = "Vrai"
dit (Atomic a)  = [a]
dit (Non p)     = "Non " ++ (dit p)
dit (Et p q)    = "( " ++ (dit p) ++ " et " ++ (dit q) ++ " )"
dit (Ou p q)    = "( " ++ (dit p) ++ " ou " ++ (dit q) ++ " )"
dit (Imp p q)   = "( " ++ (dit p) ++ " implique " ++ (dit q) ++ " )"
dit (Equiv p q) = "( " ++ (dit p) ++ " equivaut a " ++ (dit q) ++ " )"
```

```haskell
-- renvoie la liste de tous les environnements de taille n
-- on nomme les atomes dans l'ordre alpha à partir de ini
tous_les_env :: Int -> Atome -> [Environnement]
tous_les_env 0 ini = [Dic.empty]
tous_les_env n ini =
  let insere v dic = Dic.insert  ini v dic in
  (map (\d -> insere True  d) (tous_les_env (n-1) (succ ini))) ++
  (map (\d -> insere False d) (tous_les_env (n-1) (succ ini)))

-- teste si une formule est une tautologie
est_tautologie :: Formule -> Bool
est_tautologie f =
  let ens = ens_atomes f in
  let ini = Set.findMin ens in
  let n = Set.size ens in
  and ( map (\env -> eval f env) (tous_les_env n ini) )

-- renvoie la table de vérité sous forme d'une liste de couples (envir, valeur)
liste_verite :: Formule -> [([(Atome,Bool)],Bool)]
liste_verite f =
  let ens = ens_atomes f in
  let ini = Set.findMin ens in
  let n = Set.size ens in
  map (\env -> (Dic.toList env,eval f env)) (tous_les_env n ini)

-- joli (?) affichage de la table de vérité
table_verite :: Formule -> IO ()
table_verite f =
  let lv = liste_verite f in
  let s = foldl (++) "" [(show (fst c)) ++ " --> " ++ (show (snd c)) ++ "\n" | c <-
      lv] in
  putStrLn s
```

Par exemple, pour la table de la section 1.4.3 page 9 :

Haskell

```haskell
*Main Dic> let p9 =  μ  'p'  § ( Non ( μ  'q') ==>  μ  'p')
*Main Dic> table_verite p9
[('p',True),('q',True)] --> True
[('p',True),('q',False)] --> True
[('p',False),('q',True)] --> True
[('p',False),('q',False)] --> False
```

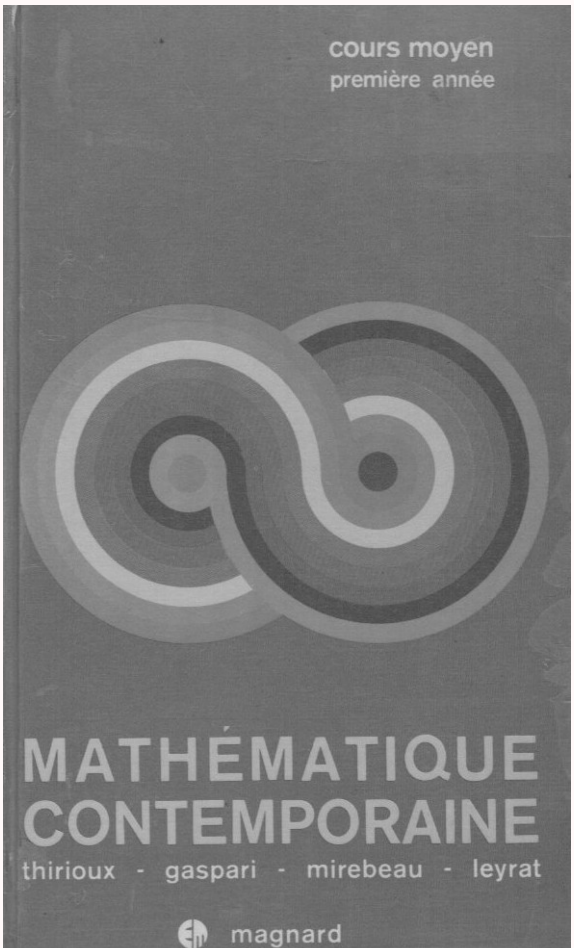
On retrouve bien le résultat annoncé :

p	q	$\neg q$	$\neg q \to p$	$p \vee (\neg q \to p)$
1	1	0	1	1
1	0	1	1	1
0	1	0	1	1
0	0	1	0	0

ANNEXE

C

Mon livre de CM1



Voici quelques pages d'un livre de CM1 des années 1970...Cela constitue une bonne base d'exercices pour réviser le DS ;-)

Exercices et problèmes

I

1. On a l'ensemble A = { 1 ; 3 ; 5 ; 7 ; 9 }. Comment peut-on appeler cet ensemble ?
2. Ecris l'ensemble B des voyelles de notre alphabet.
3. A-t-on le droit de dire que les noms brochet, carpe, grenouille sont des éléments d'un ensemble de noms de poissons ?
4. Forme un ensemble C de mots qui commencent par s, un ensemble D de mots qui se terminent par un u.
5. Ecris l'ensemble E des rois qui ont régné en France entre les années 1600 et 1780.
6. Regarde sur ton dictionnaire ce qu'on appelait une « merveille du monde » Quels étaient les éléments de cet ensemble ?
7. Regarde sur un atlas quels sont les éléments de l'ensemble des Etats qui ont une frontière commune avec la France.
8. Quand on cite les éléments d'un ensemble **on n'a pas le droit de citer deux fois le même élément.** Jean a écrit F = { Louis XIV ; Louis XV ; le roi Soleil } Quelle faute a-t-il commise ?

I

9. Les ensembles A = { m ; p ; r ; l ; k } et B = { p ; l ; k ; r ; m } sont-ils égaux ?
10. Le ensembles C = { 1 ; 7 ; 3 ; 6 } et D = { 6 ; 7 ; 3 ; 1 ; 5 } sont-ils égaux ?
11. On sait que E = { 13 ; 20 ; 12 ; 16 } que F = { 20 ; 16 ; x ; 12 } et que E = F .Trouve x.

V

12. On donne A = { 25 ; 36 ; 59 ; 41 ; 17 ; 10 } .Quel est l'ensemble B des nombres de l'ensemble A dans lesquels on trouve le chiffre 1 ? Quel est l'ensemble C des nombres de l'ensemble A dans lesquels on trouve le chiffre 8 ?

II

13. Les éléments d'un ensemble B sont choisis parmi ceux d'un univers A :

éléments de A	x	y	z	t	u	v
pour B, numéros des éléments de A	1	0	0	1	1	1

Quels sont les éléments de B ? + x + y + t + u + z + w + v
Ecris sur une feuille, comme sur le livre :
Mets en évidence les ensembles A et B.

14. Les éléments d'un ensemble D sont choisis parmi ceux d'un univers C on sait que C = { ◇ ; △ ; ▭ ; ⊠ } et que D = { ⊠ ; △ }
complète le tableau : éléments de C.

pour D numéros des éléments de C				

Sur un même dessin, obtiens un schéma représentant les ensembles C et D.

Exercices et problèmes

1. Voici les ensembles A = { 2 ; 9 ; 6 ; 4 ; 7 } et B = { 2 ; 4 ; 6 ; 8 ; 10 }
Quelle est leur intersection C ? Fais un diagramme.

2. Tu récites la suite des nombres entiers naturels : 0 ; 1 ; 2... A est l'ensemble des nombres que tu dis après 197, mais avant 205. B est l'ensemble des nombres que tu dis après 200, mais avant 210. Ecris les éléments de A, de B, de leur intersection C. Fais un diagramme.

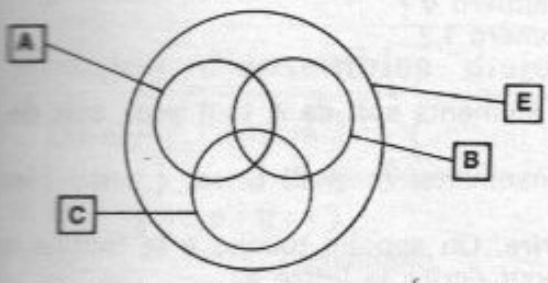
3. On donne l'univers des noms d'animaux E = {chat ; brochet ; tigre ; serpent ; poule ; vache ; mouton ; éléphant ; aigle}

éléments de E	c	b	t	s	p	v	m	e	a
pour A, N^{os} des éléments de E	1	0	1	0	0	1	1	1	0
pour B, N^{os} des éléments de E	1	0	0	0	1	1	1	0	0

Ecris les éléments de A. Qualifie les par un caractère commun (pense aux pattes).
Ecris les éléments de B et qualifie les (as-tu déjà visité une ferme ?)
Quelle est l'intersection C des ensembles A et B ? Qualifie les éléments de C.
Fais un diagramme pour les ensembles E, A, B, C

4. On te donne un univers d'enfants **E** = {Bernard ; François ; Christophe ; Didier ; Alain ; Régis ; Michel ; Pierre ; Georges ; Xavier}. Ceux de ces enfants qui lisent « Tintin » forment un ensemble **A.** Ceux de ces enfants qui lisent « Pilote » forment un ensemble **B.** Ceux de ces enfants qui lisent « Spirou » forment un ensemble **C.**

éléments de E	B	F	C	D	A	R	M	P	G	X
pour **A,** N^{os} des éléments de E	1	1	1	0	0	1	1	1	0	0
pour **B,** N^{os} des éléments de E	0	1	1	1	1	1	1	0	0	0
pour **C,** N^{os} des éléments de E	0	1	1	1	0	1	0	1	1	0



Ecris les éléments de : **A,B,C.**
Ecris les éléments de :
H, intersection de **A** et **B.**
I, intersection de **A** et **C**
J, intersection de **B** et **C.**
Y a-t-il des éléments communs à **A, B, C** ?
Leur ensemble est K.
Place les prénoms sur le diagramme ci-contre.

5. On te donne l'univers E = { Canada ; Etats-Unis ; Mexique ; Colombie ; Vénézuela ; Brésil ; Pérou ; Bolivie }. Consulte une carte de l'Amérique.
Quels numéros faut-il associer aux éléments de E pour obtenir l'ensemble A des Etats baignés par l'Océan Pacifique, pour obtenir l'ensemble B des Etats baignés par l'Océan Atlantique ?
Quelle est l'intersection C de A et B ? Fais un diagramme pour E, A, B, C.

6. On te donne l'univers de lettres E = { a ; b ; c ; d ; e ; f ; g ; h ; i }
Essaie de trouver les éléments d'un ensemble A, d'un ensemble B pour que le cardinal de A soit 3, le cardinal de B, 5, le cardinal de l'intersection C de A et B, 1. Toujours avec cardinal A = 3, cardinal B = 5, est-il possible d'avoir cardinal C = 2 ou bien cardinal C = 3, ou bien cardinal C = 4 ? Fais un diagramme dans chaque cas possible.

7. Tu as un jeu de 32 cartes. Qualifie l'intersection de l'ensemble des cartes rouges et de l'ensemble des cartes noires.

8. Prends une boîte de blocs logiques. Regarde les blocs un à un et mets dans un sac les grands triangles bleus au fur et à mesure que tu les trouves. Reprends tous les blocs, entoure d'une ficelle l'ensemble A des grands triangles et l'ensemble B des triangles bleus. Vois-tu l'ensemble des triangles bleus et grands ?

Exercices et problèmes

I

1. On donne l'ensemble E des chiffres $E = \{0 ; 1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9\}$ Ecris le sous-ensemble A des chiffres impairs. Forme le sous-ensemble B complémentaire de A par rapport à E. Comment appeler les éléments de B ?

2. On donne l'ensemble E des lettres de l'alphabet. Ecris les éléments du sous-ensemble A des consonnes. Forme le sous-ensemble B complémentaire de A par rapport à E. Comment appeler les éléments de B ?

3. On donne $E = \{1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9\}$, $A = \{1 ; 3 ; 7 ; 9\}$, $B = \{2 ; 4 ; 6 ; 8\}$. $C = \{1 ; 3 ; 5 ; 7 ; 9\}$, $D = \{2 ; 4 ; 5 ; 6 ; 8\}$. Dis si les sous-ensembles suivants sont complémentaires par rapport à E

A et B ; C et D ; A et D ; B et C

II

4. Tu as l'ensemble des blocs logiques d'une boîte et trois sacs. Dans le premier sac tu mets les blocs triangulaires ; dans le deuxième sac tu mets les blocs non triangulaires. Dans le troisième sac, y a-t-il des blocs à mettre ?

5. Tu as l'ensemble des blocs logiques d'une boîte. Montre un bloc carré et **non** bleu. Prends les blocs un à un et mets tous les blocs carrés et non bleus dans un sac.

Reprends tous les blocs ; forme sur ta table l'ensemble des blocs carrés, l'ensemble des blocs non-bleus ; entoure-les chacun d'une ficelle. Où sont placés les blocs carrés et non-bleus ?

6. Recommence comme au numéro 5 avec les blocs **non**-rouges **et non**-petits.

7. Recommence comme au numéro 5 avec les blocs carrés **ou non**-bleus.

8. Recommence comme au numéro 5 avec les blocs **non**-rouges **ou non**-petits.

9. Recommence comme au numéro 5 avec les blocs non-jaunes et non-ronds, puis avec les blocs non-jaunes ou non-ronds.

III

10. Réponds par oui ou bien par non aux questions suivantes. Peux-tu calculer (9-6), (6-3), (3-6), (6-6), (8-1), (2-7), (4-4), (123 465-0), (100 001-99 999), (4350-4035) ?

11. Calcule les nombres représentés par des lettres :

$2 + x = 9$	$15 = a + 4$	$8 + 8 = m$
$y = (7 - 3)$	$(17 - q) = 9$	$(18 - 9) = n$
$1 = (6 - z)$	$0 = (c - 5)$	$p + 5 = 10$
$(t - 8) = 3$	$d + d = 8$	$14 - q = 7$

12. A un couple de nombres, on fait correspondre un nombre par une addition ou bien une soustraction. Mets dans les cases les signes convenables :
$(8 ; 6) \xrightarrow{\square} 14$ $(13 ; 10) \xrightarrow{\square} 3$ $(15 ; 7) \xrightarrow{\square} 8$ $(3 ; 3) \xrightarrow{\square} 6$ $17 ; 9 \xrightarrow{\square} 8$
Si on te propose $(5 ; 0) \xrightarrow{\square} 5$, quels signes peux-tu mettre ?
Si on te propose $(0 ; 5) \xrightarrow{\square} 5$, as-tu le choix ?

13. Mets dans chacun des cas suivants V (vrai) ou bien F (faux) :
$(12 - 4) = 8$; $(6 - 5) < 9$; $4 > (12 - 6)$; $(10 - 9) < 2$; $(15 - 8) > 5$
$(14 - 6) > 10$; $9 = (12 - 5)$; $0 < (11 - 3)$; $5 > (13 - 9)$; $9 < (16 - 8)$

14. Quelqu'un dit : « (8 – 6) est plus grand que 3 » ; a-t-il raison ?
Quelqu'un dit : « Il est faux d'affirmer que (8 – 6) est plus grand que 3 » ; a-t-il raison ?

ercices et problèmes

1. Voici un ensemble de fleuves F = { Seine ; Loire ; Rhône ; Garonne } et un ensemble de cours d'eau : { Marne ; Allier ; Saône ; Oise ; Somme }. Représente ces ensembles et trace les traits fléchés signifiant : «....a pour affluent...» Trace le tableau représentant tous les couples (fleuve ; cours d'eau). Indique comme au paragraphe I ceux de ces couples qui associent un fleuve et un de ses affluents.

2. On a représenté un tableau indiquant les années de naissance des enfants Lesage. Construis un schéma qui te donnera les mêmes renseignements que ce tableau. Complète :

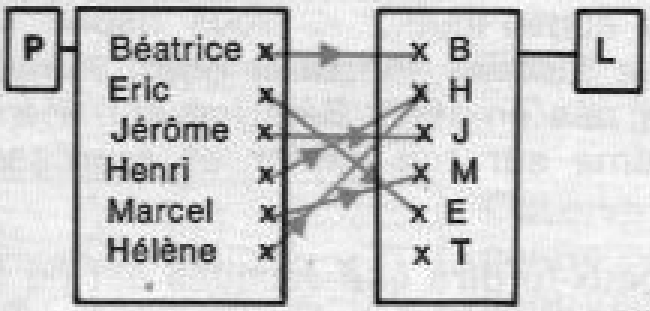
   ..... est né en 1960 ;

   ..... est né en 1958 ;

   Olivier est né en .....

	Gérard	Henri	Nathalie	Olivier	Nicole
1958	1	0	0	0	0
1960	0	1	0	1	0
1963	0	0	0	0	1
1968	0	0	1	0	0

3. On a représenté un schéma, essaie de l'interpréter. Fais un tableau donnant les mêmes renseignements :



4. Représente le schéma de l'ensemble des villes : V = { Nice ; Paris ; Lille ; Lyon ; Toulouse } puis le schéma de l'ensemble des départements D = { Seine ; Rhône ; Nord ; Alpes Maritimes ; Haute-Garonne }. Trace les traits fléchés que tu veux de chaque ville à un département convenable. Peux-tu donner une interprétation aux traits que tu as tracés ?

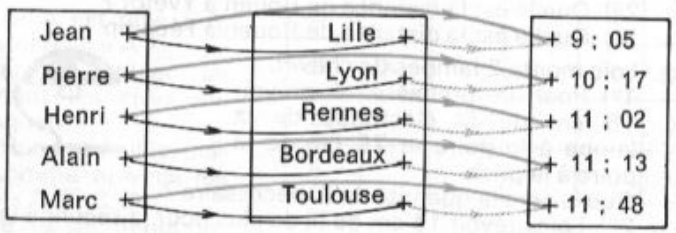
26. COMPOSITION DES RELATIONS. RELATIONS RECIPROQUES

I Composition des relations

1. Relations non numériques

Pour les vacances de Noël, un groupe d'amis parisiens se sépare. Jean va à Lille, Pierre à Lyon, Henri à Rennes, Alain à Bordeaux et Marc à Toulouse. Jean s'est renseigné au bureau de la gare sur les horaires de départ des trains.

Il a représenté un schéma indiquant l'heure de départ de chacun.



Que signifient les traits fléchés noirs ? les traits fléchés bleus ? les traits fléchés gris ?

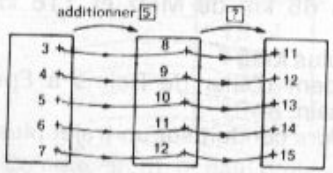
Ecris les couples (garçon ; ville) dont les termes se correspondent.

Ecris les couples (ville ; heure) dont les termes se correspondent.

Ecris les couples (garçon ; heure) dont les termes se correspondent. Que remarques-tu ?

Sur le dessin, comment Jean est-il relié à « 9 ; 05 ». (Il y a deux possibilités, soit un trait fléché bleu, soit un trait fléché noir puis fléché gris.)

2. Relations numériques



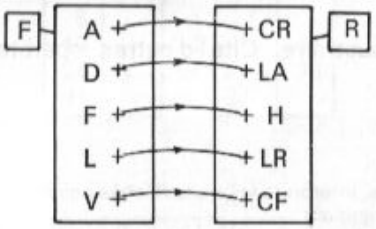
Olivier a dessiné avec des ensembles de nombres un schéma dont les traits fléchés sont semblables à ceux que nous avons tracés dans le n° 1.

Interprète les traits fléchés noirs, les traits fléchés bleus, les traits fléchés gris.

6 7	11 12	

II Relations réciproques

1) Relations non numériques



Chacune des filles : Anne (A), Denise (D), Françoise (F), Lydie (L), Véronique (V) a récité une fable de La Fontaine.

L'ensemble des filles est représenté par un schéma (chaque fille est désignée par l'initiale de son prénom)

Chaque fable est désignée par des initiales dans le schéma de l'ensemble R. Il y a : le Corbeau et le Renard (CR), le Loup et l'Agneau (LA), le Héron (H), le Lion et le Rat (LR), et la Cigale et la Fourmi (CF).

Les traits fléchés noirs signifient... a récité...

Ecris les couples dont les termes sont reliés par les traits fléchés noirs.

Qui a récité « le Héron » ? Qui a récité « la Cigale et la Fourmi » ?

Tu peux répondre : « Françoise a récité « Le Héron », à la première question.

Mais tu peux donner le même renseignement par la phrase :

« Le Héron » a été récité par Françoise.

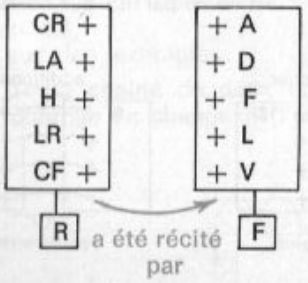
Réponds à la seconde question de deux manières.

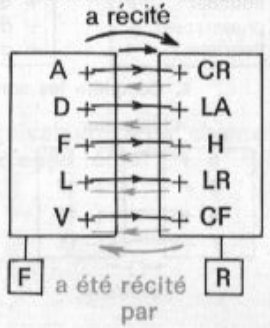
Donne d'autres exemples.

Trace sur le tableau de gauche les traits fléchés signifiant :... a été récité par...

Tu aurais pu donner les mêmes renseignements sur le premier tableau comme cela a été fait sur le schéma de droite.

Ecris les couples : (récitation-fille) dont les termes sont reliés par les traits fléchés bleus.





2. Ecris en base trois, en base quatre, en base cinq, en base six :

a) le nombre de doigts de tes mains ;
b) le nombre de crayons de ta boîte de crayons de couleurs ;
c) le nombre de tubes de ta boîte de peinture.

3. Les nombres suivants sont-ils écrits ?

a) 2 1 3 en base trois ? pourquoi ?
b) 2 4 2 en base trois ? en base quatre ? pourquoi ?
c) 3 3 5 en base trois ? en base quatre ? en base cinq ? pourquoi ?

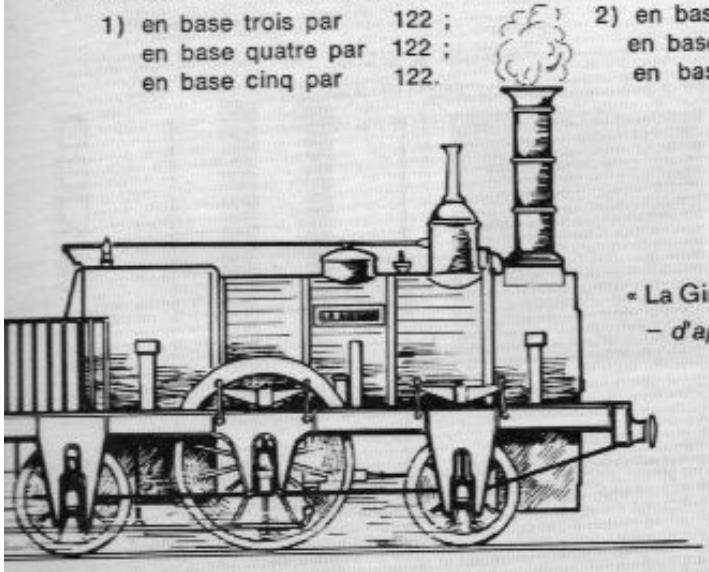
4. J'indique le nombre qui précède et celui qui suit :

	nombre qui précède	nombre donné	nombre qui suit
1) base trois		1 1 1	
		1 2 1	
		2 0 1	
		2 1 1	
		2 2 1	
2) base quatre		1 2 3	
		1 3 2	
		2 0 1	
		2 1 0	
		2 1 3	
		2 2 2	
		2 3 1	
		3 0 0	
3) base cinq		2 3 4	
		2 4 2	
		3 0 0	
		3 0 3	
		3 1 1	
		4 4 3	

5. Je dessine les billes dénombrées :

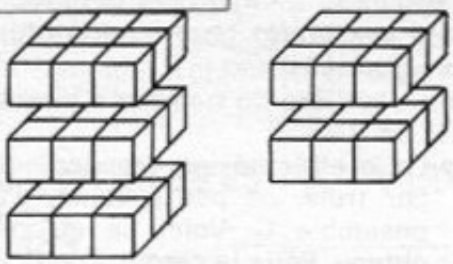
1) en base trois par 122 ;
en base quatre par 122 ;
en base cinq par 122.

2) en base trois par 200 ;
en base quatre par 102 ;
en base cinq par 33.



« La Gironde » - Creusot - 1839
– d'après cl. « La vie du rail »

III Sous-unités



Voici un ensemble A de petites plaques.
En utilisant le tableau, écris le cardinal de A en prenant la petite barre comme unité-repère.

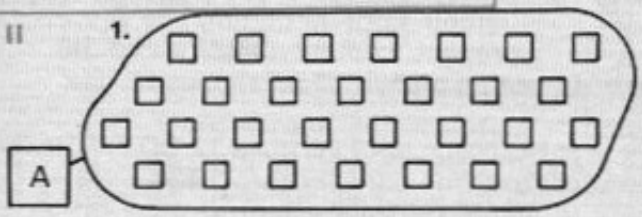
cubes	petites plaques	petites barres	petits cubes
1	2	0	

↑ position de la virgule

Pour indiquer l'absence de petite barre, on met un zéro dans la colonne des petites barres.
On écrit : en petites barres, le cardinal de l'ensemble de petites plaques est exprimé par l'écriture 120.
Ecris le cardinal de A en prenant le petit cube comme unité-repère.

Exercices et problèmes

II **1.**



On a représenté un ensemble A de petits cubes. Exprime en base quatre le cardinal de A en prenant comme unité : la petite barre, la petite plaque, le grand cube.

2. Exprime en base cinq le cardinal de l'ensemble A (exercice n° 1) en prenant comme unité : la petite barre, la petite plaque, le grand cube.

3. Exprime en base six le cardinal de l'ensemble (exercice n° 1) en prenant comme unité : la petite barre, la petite plaque, le grand cube.

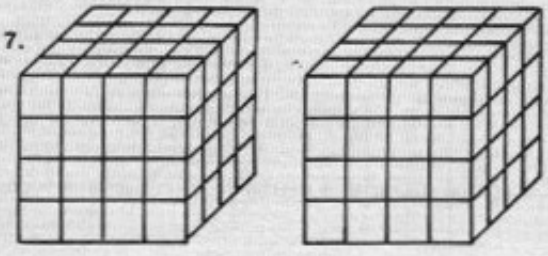
4. A, B, C, sont des ensembles de petits cubes. On a effectué les groupements en base quatre, les résultats obtenus ont été portés sur le tableau. Exprime le cardinal de chacun de ces ensembles en prenant comme unité repère le petit cube, la petite barre, la petite plaque, le grand cube.

ensembles	cubes	petites plaques	petites barres	petits cubes
A	2	1	0	0
B		2	0	3
C			1	3

5. Dessine un ensemble de croix dont le cardinal est l'unité repère étant le groupement de cinq croix, exprimé par l'écriture 12.

6. A est un ensemble de croix. Un groupement remplace cinq croix, un grand groupement remplace cinq groupements.
Exprime le cardinal de A en base cinq en prenant pour unité-repère le grand groupement, si le cardinal de A (l'unité-repère étant la croix) est 10 ; 213 ; 2 ; 100.

III **7.**

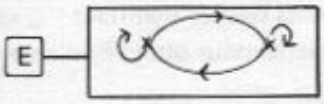


Le matériel utilisé est le matériel base quatre.
Voici un ensemble de grands cubes. Exprime en base quatre le cardinal de cet ensemble en prenant pour unité le grand cube, la petite plaque, la petite barre, le petit cube.

III Propriété de la relation : «... est diviseur de... » dans l'ensemble des nombres entiers

- Choisis deux nombres. Représente l'ensemble de ces deux nombres. Peux-tu relier les points qui les représentent par un trait signifiant "...est diviseur de ..."? (Cela dépend des nombres choisis).

On a représenté un ensemble de trois nombres et des traits fléchés signifiant "... est diviseur de ..." . Ce schéma est-il complet ?
Trouve un ensemble de nombres qui conviennent.



On a représenté un ensemble de deux nombres et des traits fléchés. Les traits ne peuvent pas avoir pour signification "... est diviseur de ...". Pourquoi ?
Supprime un trait pour que le schéma puisse avoir cette signification. Trouve des couples convenables de nombres.

On a représenté un ensemble de nombres et des traits fléchés signifiant : "... est diviseur de ..." Ce schéma n'est pas complet. Pourquoi ? Complète-le. Trouve un ensemble convenable de nombres.

IV Recherche des diviseurs d'un nombre

Recherchons tous les diviseurs du nombre 12. Nous savons que ce sont des nombres inférieurs ou égaux à 12. Tu dois donc essayer tous les nombres.
L'ensemble $\{1, 2, 3, 4, 6, 12\}$ est appelé l'ensemble des diviseurs du nombre 12.

Tableau

est diviseur de	1	2	3	4	5	6	7	8	9	10	11	12
12												
11												
10												
9												
8												
7												
6												
5												
4	1	1	0									
3	1	0	1									
2	1	1	0									
1	1	0	0									

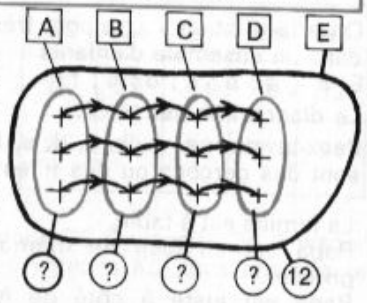
Complète le tableau en écrivant des "1" dans les cases correspondant aux couples dont le premier terme est un diviseur du second, 0 dans les autres cases.
Ce tableau permet de lire directement les diviseurs d'un nombre. Par exemple, en suivant la bande de 12 on voit que 1,2,3,4,6,12 sont des diviseurs de 12. Quels sont les diviseurs de 8, 9, 10,11 ? Certains nombres n'ont pour diviseur que 1 et eux-mêmes. Fais-en la liste.

cahiers ?

46. DEFINITION DE LA DIVISION DES NOMBRES NATURELS

I Partition d'un ensemble en sous-ensembles disjoints de même cardinal

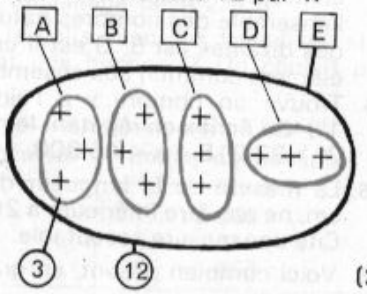
- On veut partager 12 bonbons entre quatre enfants. Chacun doit avoir le même nombre de bonbons que les autres.
Cela revient à partager un ensemble E de cardinal 12 en quatre sous-ensembles de même cardinal.
On attribue successivement un élément à chaque sous-ensemble comme l'indiquent les traits fléchés noirs.



(1)

Les bonbons distribués sont successivement $(4 \times 1) = 4$, $(4 \times 2) = 8$, $(4 \times 3) = 12$. 4, 8, 12 sont des **multiples de 4.**
Le partage est terminé.
On convient d'écrire $(12 : 4) = 3$.
Au couple (12 ; 4) correspond le nombre 3 par une opération qui s'appelle **division.**
Le premier terme du couple (12 ; 4) est le dividende, le second terme est le diviseur. 3 est le quotient exact de 12 par 4.

- On a une pile de 12 cahiers. Il faut distribuer 3 cahiers par élève. Combien d'élèves peut-on servir ?
Tout à l'heure, on connaissait le nombre de sous-ensembles. On cherchait le cardinal de chacun.
Maintenant, on connaît le cardinal de chaque sous-ensemble. On cherche le nombre de sous-ensembles. On trace le schéma.



(2)

- Les cahiers distribués sont successivement
$(3 \times 1) = 3$; $(3 \times 2) = 6$; $(3 \times 3) = 9$; $(3 \times 4) = 12$
La distribution est terminée.
3, 6, 9, 12 sont des **multiples de 3**
On écrit $(12 : 3) = 4$.

- Regarde le premier schéma de la leçon : « Définition de la multiplication des nombres naturels ». Ce schéma ressemble au schéma (2) que nous avons dessiné aujourd'hui.

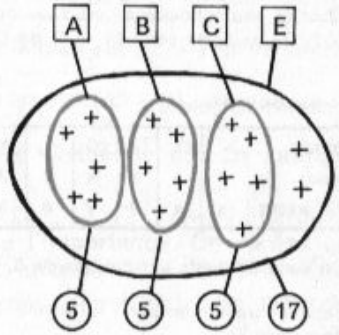
La division exacte $(12 : 3) = 4$ est liée à la multiplication : $12 = (3 \times 4)$.

- A quelles multiplications sont liées les divisions :
$(12 : 4)$; $(15 : 3)$; $(18 : 2)$?

II Division avec reste

On veut distribuer 17 cerises entre 3 enfants. Les règles du jeu sont les suivantes :

- Chacun doit avoir le même nombre de cerises que les autres.
- Chacun doit avoir le plus possible de cerises.



Peut-on trouver x tel que $17 = (3 \times x)$?
Non, car 17 n'est pas multiple de 3.
L'écriture (17 : 3) ne représente pas un nombre naturel.
Complète le tableau :

nombre de cerises reçues par chaque enfant	1	2	
nombre de cerises distribuées	3	6	

Que constates-tu ?
$3 \times 5 < 17$; tu peux donner 5 cerises par enfant.
$3 \times 6 > 17$; tu ne peux pas donner 6 cerises par enfant.
$3 \times 5 < 17 < 3 \times 6$.
Tu peux aussi écrire : $17 = (3 \times 5) + 2$ et $2 < 3$.
5 est le quotient entier de 17 par 3. On note $(17 \div 3) = 5$.
Tu donnes 5 cerises à chacun et il te reste 2 cerises.
Au couple (17 ; 3), tu as fait correspondre le couple (5 ; 2).

dividende diviseur quotient reste

Quand tu écris $17 = (3 \times 5) + 2$, écris bien le diviseur **avant** le quotient pour pouvoir le reconnaître et n'oublie pas l'inégalité $2 < 3$ car le reste doit être inférieur au diviseur.

un séjour de 11 jours.
La longueur d'un rouleau de fil de fer mesure 25 m. Quelle est la mesure de la longueur de 9 rouleaux ?
Quel est le prix de 9 disques à 21 F l'un ?

48. DIVISION DES NOMBRES NATURELS : LE DIVISEUR A UN CHIFFRE. LE QUOTIENT EST QUELCONQUE

I Division et addition

Cas de deux quotients exacts.

Complète le tableau.

x	y	z	(x : z)	(y : z)	(x : z) + (y : z)	x+y	(x+y) : z
12	16	4					
28	14	7					
45	36	9					

Que remarques-tu ?

Cas d'un seul quotient exact.

Complète le tableau.

x	y	z	(x ÷ z)	(y : z)	(x ÷ z) + (y : z)	x+y	(x+y) ÷ z
35	40	8					
29	18	6					
31	10	5					

Que remarques-tu ?

Cas où il n'y a pas de quotient exact.

As-tu le droit d'écrire $(40 \div 6) + (9 \div 6) = (49 \div 6)$?
non $6 + 1 \neq 8$

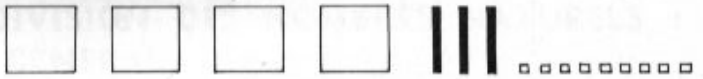
Applications

$86 : 2 = (80 : 2) + (6 : 2) = 40 + 3 = 43$.
$57 \div 5 = (50 : 5) + (7 \div 5) = 11$.
$124 : 4 = (120 : 4) + (4 : 4) = 31$.
$42 \div 3$. As-tu le droit d'écrire $(40 \div 3) + (2 \div 3)$? Non.
Trouve une autre idée.
$96 : 2$. On a le droit d'écrire $(90 : 2) + (6 : 2)$. Trouve une autre idée, pour faire une opération plus simple que 90 : 2 ; pense à 80.

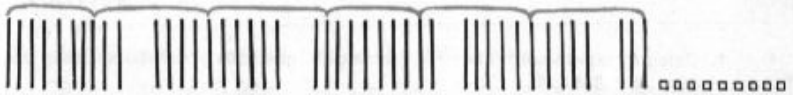
Manipulation

On cherche le quotient entier de 439 par 6.

Représente 439 avec du matériel : petites plaques, petites barres, petits cubes (on pourrait choisir aussi des billets, des pièces).



1. Peux-tu faire 6 sous-ensembles de plaques de même cardinal ?
non.
Remplace les plaques par des barres.



2. Peux-tu faire 6 sous-ensembles de barres de même cardinal ?
oui.
Chaque sous-ensemble sera de 7 barres.
Il reste une barre qu'on ne peut partager. Remplace-la par des cubes.

3. Peux-tu faire 6 sous-ensembles de cubes de même cardinal ?
Oui, chaque sous-ensemble est de 3 cubes. Il reste 1 cube.

Complète : $439 = (6 \times \square) + \square$, $\square < \square$

Calcul :

Tu cherches le quotient entier de 439 par 6. Quel est le nombre de chiffres de ce quotient ?

$6 \times 1 = 6$ $6 \times 10 = 60$ $6 \times 100 = 600$
$6 \times 10 < 439 < 6 \times 100$ Le quotient a deux chiffres.

Tu cherches d'abord le chiffre des dizaines, ensuite le chiffre des unités.

Le nombre des dizaines de 439 est 43. Tu trouves le chiffre des dizaines du quotient en cherchant le quotient entier de 43 par 6.

Ce quotient est 7.

Il reste une dizaine qu'on remplace par 10 unités.

Le nombre des unités à partager est 19.

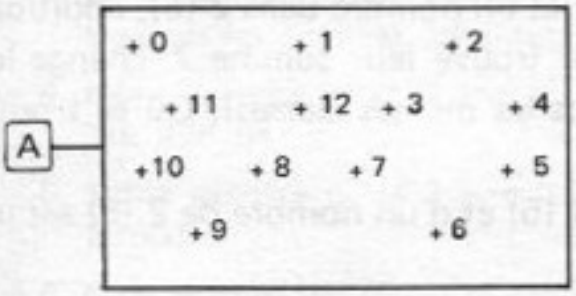
$19 \div 6 = 3$. Il reste une unité.

	439	6
disposition	42	73
pratique	19	
détaillée	18	
d'ordinaire on n'écrit pas	1	
42 et 18		

87 CLASSEMENT PAR 5

I Restes dans la division par 5

1. Effectue les divisions de 21 par 5 et 46 par 5.
Tu trouves le même reste : 1
Nous dirons que 46 donne dans la division par 5, le même reste que 21. A est l'ensemble des nombres de zéro à 12. Trace les traits fléchés signifiant "... donne le même reste, dans la division par 5 que ..." Recommence le schéma de façon que les flèches ne se coupent pas.



Est-ce possible ? Quel classement peux-tu effectuer dans l'ensemble A ?

2. Classons les nombres entiers.
Comme nous venons de classer les nombres entiers de zéro à 12, nous allons essayer de classer tous les nombres entiers.
Quels sont les restes possibles dans la division d'un nombre par 5 ?
Traçons 5 colonnes. Nous mettrons dans la première les nombres qui donnent pour reste 0, dans le second ceux qui donnent pour reste 1 et ainsi de suite ...

restes	0	1	2	3	4
nombres	0	1	2	3	4
	5	6	7	8	9
	10	11	12	13	14
	..	..	..	..	..

Continue à placer les nombres suivants de la même manière (par exemple jusqu'à 30).
Quels sont les nombres de la 1 ère colonne ? de la 2ème ? la 3 ème ?

Fais la différence entre deux nombres quelconques de la même colonne. Qu'observes-tu ?

3. Notation et vocabulaire.

André, Jean et Pierre font partie de la même équipe. Pour désigner cette équipe on peut aussi bien dire : équipe d'André, équipe de Jean ou bien équipe de Pierre.

L'ensemble des nombres de la 1ère colonne sera appelé la classe de zéro par 5 et noté $\dot{0}$ (5), ou bien classe de 5 par 5 et noté $\dot{5}$(5), ou bien classe de tout autre nombre de la colonne 10 (5), $\dot{1}\dot{5}$(5) ...) ;

Egalité : $\dot{0}$ (5), $\dot{5}$ (5) désignent le même ensemble, tu peux écrire : $\dot{0}$(5) = $\dot{5}$ (5) ou bien, plus simplement $\dot{0} = \dot{5}$ (5).

De la même manière les nombres de la deuxième colonne constituent : $\dot{1}$ (5) ou bien $\dot{6}$ (5), ... et $\dot{1} = \dot{6} = 11$ (5).

4. Comment trouver la classe d'un nombre sans effectuer la division ?

Observe le chiffre des unités des nombres du tableau pour découvrir une règle.

Complète le tableau avec les notations les plus simples ($\dot{0}, \dot{1}, \dot{2}, \dot{3}, \dot{4}$).

Chiffres des unités	0	1	2	3	4	5	6	7	8	9
Classe par 5			$\dot{2}$				$\dot{1}$			

II Addition des classes

Prends un nombre dans $\dot{1}$ (5) et un nombre dans $\dot{2}$ (5), additionne ces deux nombres. Où se trouve leur somme ? change les nombres (en les prenant dans les mêmes classes). Où se trouve la nouvelle somme ?

La somme d'un nombre de $\dot{1}$ (5) et d'un nombre de $\dot{2}$ (5) est un nombre de $\dot{3}$ (5)

Nous écrivons ceci $1 \dot{+} 2 = \dot{3}$ (5) (nous mettons un point sur +, car ce n'est pas l'addition ordinaire et nous lisons un pointé plus deux pointé égale trois pointé).

Complète les égalités : $\dot{0}\dot{+}\dot{2} = \square$; $\dot{1}\dot{+}\dot{3} = \square$; $\dot{3}\dot{+}\dot{2} = \square$

$\dot{0}$	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$
0	1	2	3	4
5	6	7	8	9
.	.	.	.	.
.	.	.	.	.

$\dot{3} \dot{+} \dot{2} = \dot{0}$

Pour donner tous les résultats on dresse une table d'addition.

Complète la avec les notations les plus simples.

$\dot{+}$	$\dot{0}$	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$
$\dot{0}$			$\dot{2}$		
$\dot{1}$			$\dot{3}$	$\dot{4}$	
$\dot{2}$	$\dot{2}$	$\dot{3}$		$\dot{0}$	
$\dot{3}$		$\dot{4}$	$\dot{0}$		
$\dot{4}$					

Remarque. Effectue $4+3$; peux-tu écrire $\dot{4}\dot{+}\dot{3} = \dot{7}$?

$\dot{4}\dot{+}\dot{4}$; peux-tu écrire $4+4 = 3$?

Il ne faut pas confondre l'addition des classes avec celle des entiers.

Multiplication des classes : ce que tu viens de faire pour l'addition, tu peux le faire de la même manière pour la multiplication et dresser une table de multiplication.

III Divisibilité par 5

Quels sont dans le tableau du I les nombres qui sont divisibles par 5 ? (ceux de la 1 ère colonne)

Peux-tu trouver un moyen de savoir quand un nombre est divisible par 5 sans effectuer la division ?

- **Les nombres divisibles par 5 sont ceux qui ont pour chiffre des unités zéro ou bien cinq.**

Exercices

1.

11	+ 21	+ 9	+ 7	
+ 53	+ 19	+ 50	+ 22	+ 45
14	+ 28	+ 1		+ 55

Trace les traits fléchés signifiant : "... donne le même reste dans la division par 5, que..." Classe les nombres de l'ensemble A, d'après les traits fléchés (tu auras soin de refaire un schéma plus propre).

2. Ecris chacun des nombres suivants sous la forme; $(5Xa)+b$ et $b < a$ puis classe les nombres suivant leur reste dans la division par 5 :
17, 32, 26, 30, 5, 9, 22, 38.

3. En adoptant les notations les plus simples, indique dans quelle classe se trouvent les nombres suivants : 6, 3, 7, 11, 17, 38, 45, 52, 73, 94, 203.

4. Même exercice qu'en 3 avec les nombres : 62, 107, 109, 112, 120, 204, 216, 428, 542.

5. Même exercice qu'en 3 avec les nombres : 6 738, 4 724, 21 894, 26 215, 30 000.

6. Note chacune des classes suivantes sous la forme la plus simple (exemple : $\dot{7} = \dot{2}\,(5)$), $\dot{8}\,(5)$, $\dot{9}\,(5)$, $\dot{13}\,(5)$, $\dot{25}\,(5)$, $\dot{32}\,(5)$, $\dot{47}\,(5)$, $\dot{54}\,(5)$.

7. Complète le tableau suivant en utilisant la table.

a	$\dot{2}$	$\dot{1}$	$\dot{3}$	$\dot{4}$	$\dot{2}$	$\dot{0}$	$\dot{1}$	$\dot{4}$	$\dot{0}$
b	$\dot{0}$	$\dot{4}$	$\dot{3}$	$\dot{2}$	$\dot{3}$	$\dot{2}$	$\dot{4}$	$\dot{2}$	$\dot{2}$
c	$\dot{2}$	$\dot{2}$	$\dot{4}$	$\dot{3}$	$\dot{3}$	$\dot{4}$	$\dot{3}$	$\dot{0}$	$\dot{0}$
a + b									
(a+b) + c									
b + c									
a + (b + c)									

Compare les bandes encadrées en bleu.

Trouve une conclusion.

As-tu déjà écrit des égalités du même genre ?

8. Effectue les opérations suivantes : $\dot{2}+\dot{4}+\dot{1}\,(5)$; $\dot{3}+\dot{0}+\dot{2}+\dot{3}+\dot{1}\,(5)$;
$\dot{1}+\dot{2}+\dot{3}+\dot{2}+\dot{3}+\dot{4}\,(5)$; $\dot{1}+\dot{4}+\dot{4}+\dot{4}+\dot{4}+\dot{2}\,(5)$; $\dot{0}+\dot{3}+\dot{2}+\dot{2}+\dot{3}+\dot{4}+\dot{2}+\dot{1}+\dot{3}\,(5)$

9. Complète le tableau suivant en écrivant 1 si le nombre est divisible par 5, 0 sinon.

nombres	27	45	59	76	97	200	121	228	553	610	1225	3004
divisibilité par 5												

10. On remplace autant qu'on peut 48 pièces de 1 centime par des pièces de 5 centimes. Combien reste-t-il de pièces de 1 centime à la fin de l'opération?
Même question quand on remplace 70 pièces de 1 franc par des pièces de 5 francs.
Même question quand on remplace 40 billets de 10 francs par des billets de 50 francs.

Nombres	39	78	40	125	900	1652	9874	10001	25373
Divisibilité par 2									

2.

Divisibles par 2	Non divisib. par 2

Place dans le tableau les nombres :

325, 612, 318, 213, 17, 300, 2 652, 3 839, 4 615, 2, 9, 306, 205, 4 912, 17 822.

3. Dans chacun des schémas suivants, entoure l'ensemble des nombres divisibles par 2.

+2	+7	+191	+13
+6	+8	+80	
			+174
+619			

+29			+713	+2 913
	+202	+116		+3 011
			+94	250+
+2 046	+801		+1 018	+322

4. Effectue les additions suivantes de classes par 2, notées $\dot{0}$ ou bien $\dot{1}$.

$\dot{1}+\dot{0}+\dot{1}=....(2)$; $\dot{1}+\dot{1}+\dot{1}+\dot{0}+\dot{1}=....(2)$; $\dot{1}+\dot{0}+\dot{1}+\dot{0}+\dot{1}=....(2)$

$\dot{1}+\dot{1}+\dot{0}+\dot{0}+\dot{1}=....(2)$; $\dot{1}+\dot{1}+\dot{1}+\dot{1}+\dot{1}+\dot{1}=....(2)$.

II **5.** Dans cet exercice nous allons chercher un moyen rapide de trouver la classe par 4 d'un nombre.

Rappel : Dans le tableau de classement par 4, prends un nombre x. Additionne 4 à ce nombre; dans quelle colonne se trouve x+4? Dans quelle colonne se trouve x+(2X4)? x+(3X4)? x+(17X4)?

Prends un nombre; par exemple 1 531, il peut s'écrire 1 531 = 1 500 + 31.

Fais le même raisonnement avec 617, 828, 9 602, 18 800.

Conclus. Tout nombre (plus grand que 100) est dans la même classe par 4 que le nombre formé par ses deux derniers chiffres.

Enonce une règle de divisibilité par 4.

6. Tu as trouvé dans le 6 la règle de divisibilité par 4 : un nombre est divisible par 4 si le nombre formé par ses deux derniers chiffres est divisible par 4.

On te rappelle qu'en général une année est bissextile si le nombre qui la désigne est divisible par 4. En utilisant la règle de divisibilité par 4 dis si les années suivantes étaient bissextiles? 1907, 1916, 1784, 1839, 1922 1940. 1984 sera-t-elle bissextile ?

7. Le règlement interdit le stationnement devant chez moi du 1er au 15, l'autorisant les autres jours.

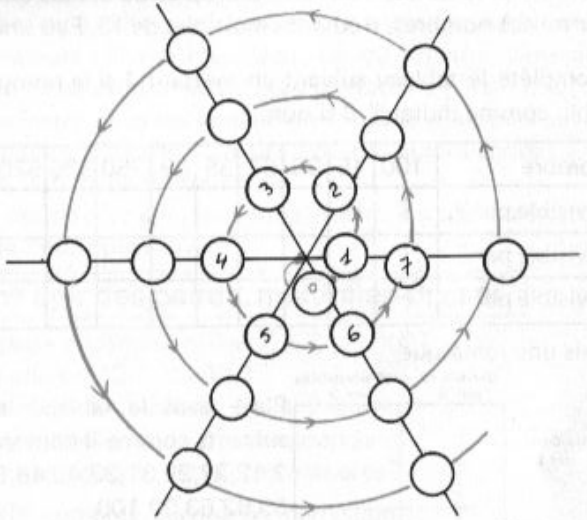
Combien de jours pourrai-je stationner devant chez moi en 1975? 1976?

(9 x 9) + 8 ; (2 x 9) + 1 ; (3 x 8) + 2 ; (5 x 5) + 4 ; (7 x 8) + 4 ; (9 x 7) + 8.

90 PROBLEMES SUR CLASSEMENTS ET DIVISIBILITE

I Classement

1.



Ecris la suite des nombres en suivant le sens des flèches.

Place ensuite les nombres de chaque demi-ligne droite sur ce tableau. Les nombres naturels ne sont-ils pas ainsi classés ?

0	1	2	3	4	5
6	7				

2. Les nombres : 13, 21, 32, 36, 40, 45, 77 sont écrits en base dix. Si tu voulais les écrire en base trois, quels sont ceux qui auraient pour chiffre des unités 0, 1, 2 ? Quels sont ceux qui auraient pour chiffre des unités 0, 1, 2, 3, 4 si tu voulais les écrire en base cinq ?

3. Les nombres 48, 74, 108, 220, 236, 1805, sont écrits en base dix. Trouve pour chacun d'eux le chiffre des unités, si tu voulais les écrire en base trois, en base cinq, en base neuf.

4. On a effectué les additions de classes suivantes; indique pour chacune d'elles à quel classement se rapportent ces classes en complétant les parenthèses.

$\dot{1}\dot{+}\dot{1}=\dot{0}$ (☐) ; $\dot{3}\dot{+}\dot{2}=\dot{1}$ (☐) ; $\dot{4}\dot{+}\dot{3}=\dot{2}$ (☐) ;

$\dot{1}\dot{+}\dot{2}=\dot{0}$ (☐) ; $\dot{8}\dot{+}\dot{6}=\dot{5}$ (☐) ; $\dot{5}\dot{+}\dot{3}=\dot{2}$ (☐)

LECTURES RECOMMANDÉES

J. Backus,
« Can programming be liberated from the von neumann style ? a functionnal style and its algebra of programs »,
Dans Communications of the ACM, vol. 21, p. 613–640, ACM, août 1978.

N. Bourbaki,
Éléments de mathématique - Théorie des ensembles,
Hermann, 1960.

T. Brugère et A. Mollard,
Mathématiques à l'usage des informaticiens,
Ellispes, 2003.

A. Brygoo, T. Durand, M. Pelletier, C. Queinnec et M. Soria,
Programmation récursive (en Scheme),
Dunod, 2004.

K. Doets et J. van Eijck,
The haskell road to logic, math and programming, 2004.

M. Lipovača et V. Robert,
« Apprendre haskell vous fera le plus grand bien ! », adresse : http://lyah.haskell.fr/, 2013.

S. Lipschutz et M. Lipson,
Schaum's outlines : Discrete mathematics,
Mac Graw & Hill, 2007.

J. Munro,
Discrete mathematics for computing,
Chapman & Hall, 1992.

P.-C. Scholl, M.-C. Fauvet, F.Lagnier et F.Maraninchi,
Cours d'informatique : langages et programmation,
Masson, 1993.

D. Spivak,
Category theory for scientists, 2013.

Thirioux, Gaspari, Mirebeau et Leyrat,
Mathématique contemporaine - CM1,
Magnard, 1970.

S. Veigneau,
Approches impérative et fonctionnelle de l'algorithmique,
Springer, 1999.