Mathématiques

IUT INFO 1 / 2012-2013

Mathématiques discrètes pour l'informatique (I)

Guillaume CONNAN

IUT NANTES
UNIVERSITÉ DE NANTES

TABLE DES MATIÈRES

CHAPITRE 1

Récursion et programmation fonctionnelle

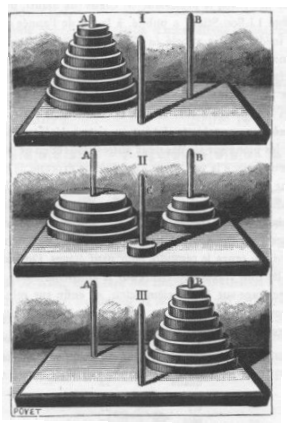
« *D'une goutte d'eau, un logicien pourrait* ***inférer*** *la possibilité d'un océan Atlantique ou d'un Niagara, sans avoir vu ni l'un ni l'autre* » : voici comment Sherlock HOLMES présente sa méthode d'enquête, l'induction (et non pas la déduction). Nous verrons comment le célèbre détective de CONAN DOYLE va nous permettre de simplifier l'écriture d'un programme...Menons l'enquête...

1 Un petit jeu pour commencer

GNU : Gnu's Not Unix

Définition : si vous ne comprenez toujours pas le sens du mot « récursion », relisez cette phrase.

Mathémator : Commençons notre année par un petit jeu :



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, 8 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.

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 : $\mathrm{M}_1 = 1$. Avec deux disques, hop, hop, hop : trois mouvements ; $\mathrm{M}_2 = 3$. 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 :

$$\mathrm{M}_{n+1} \leqslant 2\mathrm{M}_n + 1$$

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 étudierons cela en détail dans le prochain chapitre sur la logique mais nous pouvons nous contenter pour le moment d'une approche intuitive.

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 $P \Longrightarrow Q$.
- Q est une **condition suffisante** pour avoir P s'il suffit que Q soit vraie pour que P soit vraie. On note souvent $P \Longleftarrow Q$.
- 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 $P \Longleftrightarrow Q$.

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

Nous développerons des méthodes de preuve dans la section suivante.
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 une lettre, 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 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 ailleurs.* **(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 reparlerons plus loin. 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 $n \geqslant n_0$, $\mathcal{P}_n \Longrightarrow \mathcal{P}_{n+1}$ (on parle parfois d'hérédité).

Cela peut par exemple se prouver en *raisonnant par l'absurde* comme nous le verrons en exercice mais nous en reparlerons plus tard.
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 $\mathcal{P}_k \Longrightarrow \mathcal{P}_{k+1}$ 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 et nous pouvons espérer passer des vacances de Noël 2012 assez tranquilles.

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 informaticien 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 OCAML qui aime les récursions :
On commence par créer une fonction qui affichera le mouvement effectué :

```
let mvt depart arrivee=
print_string
("Déplace un disque de la tige "^depart^" vers la tige "^arrivee);
print_newline();;
```

mouvement élémentaire

Le programme proprement dit :

```
let rec hanoi a b c= function
   | 0 -> ()  (*0 disque : on ne fait rien*)
   | n -> hanoi a c b (n-1); (*n-1 disques sont déplacés de a vers b*)
          mvt a c; (*on déplace le disque restant en a vers c*)
          hanoi b a c (n-1) (*n-1 disques sont déplacés de b vers c*);;
```

résolution récursive du problème des tours de Hanoï

Les phrases entre `(*` et`*)` sont des commentaires.
Par exemple, dans le cas de 4 disques, on obtient instantanément :

```
# hanoi "A" "B" "C" 4;;
Déplace un disque de la tige A vers la tige B
Déplace un disque de la tige A vers la tige C
Déplace un disque de la tige B vers la tige C
Déplace un disque de la tige A vers la tige B
Déplace un disque de la tige C vers la tige A
Déplace un disque de la tige C vers la tige B
Déplace un disque de la tige A vers la tige B
Déplace un disque de la tige A vers la tige C
Déplace un disque de la tige B vers la tige C
Déplace un disque de la tige B vers la tige A
Déplace un disque de la tige C vers la tige A
Déplace un disque de la tige B vers la tige C
Déplace un disque de la tige A vers la tige B
Déplace un disque de la tige A vers la tige C
Déplace un disque de la tige B vers la tige C
- : unit = ()
```

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 »

2 Fonction

2.1 Souvenirs du lycée

Nous étudierons dans un prochain chapitre les relations en général et plus spécifiquement les fonctions qui sont un cas particulier de relations.
Nous nous contenterons pour l'instant des acquis(?) du lycée dans ce domaine...
Prenons un énoncé d'exercice de Seconde :
« *Soit f la fonction qui à un entier k associe l'entier naturel k^2. Calculer $f(2)$ et $f(2-3)$* ».
La première partie de l'énoncé *définit* la fonction f. Plus formellement, cela donne :

$$f : \begin{array}{ccc} \mathbb{Z} & \to & \mathbb{N} \\ k & \mapsto & k \times k \end{array}$$

f est le nom de la fonction, $\mathbb{Z}$ est son domaine et $\mathbb{N}$ son codomaine.

Ensuite, on nous demande d'évaluer deux applications de la fonction f : avec l'argument 2 puis avec l'argument $2-3$ qu'il faudra d'abord évaluer.
Jusque là, tout va bien. Dans n'importe quel langage informatique, une telle fonction peut être définie.
Par exemple en ADA :

```
function F (X: in INTEGER) return INTEGER is
begin
   return X*X;
end F;
```

ou en CAML :

```
let f = fun x -> x*x;;
```

Cependant, peu de fonctions peuvent être définies de la sorte, c'est-à-dire *explicitement*.
Par exemple, la fameuse fonction puissance que votre professeur de collège a introduite ainsi :

$$\text{puissance}(x,n) = \underbrace{x \times x \times \cdots \times x}_{n \text{ fois}}$$

Comment rentrer ces $\cdots$ dans la machine ? En fait, ils expriment ici une relation de *récurrence* qui aurait due être introduite au collège sous cette forme :

$$\text{puissance}(x,n) = \begin{cases} 1 \text{ si } x = 0 \\ x \times \text{puissance}(x, n-1) \text{ si } x > 0 \end{cases}$$

On peut montrer que la fonction puissance est l'*unique* fonction vérifiant ces équations.
Elle est alors définie *implicitement*.
Mais attention ! Toute équation ne donne pas une bonne définition.
Par exemple, l'équation $f(x) = f(x)$ a une infinité de solutions et va embarrasser notre machine.
Quant à l'équation $f(x) = f(x) + 1$, ce ne sera guère mieux : pourquoi ?
De manière général, en mathématiques, on peut définir des fonctions à l'aide de l'*opérateur de description* :

$$[f \mid \mathrm{P}(f)]$$

qui se lit « l'unique fonction f qui vérifie la propriété P ».
À la fin de ce module, on pourra définir la fonction puissance ainsi :

$$\text{puissance} = \left[f \mid \forall x \, (x \in \mathbb{R} \wedge f(x,0) = 1) \wedge \forall x \forall n \, ((x,n) \in \mathbb{R} \times \mathbb{N} \wedge f(x,n+1) = x \times f(x,n))\right]$$

Voici notre fonction *spécifiée* : notre client nous commande un programme et nous explique ce qu'il attend. Si nous vivions dans un monde uniquement mathématique, on s'arrêterait ici : on peut prouver qu'une unique fonction vérifie cette spécification et donc ça marche.
Malheureusement, il y a une part d'électronique en informatique et il va donc falloir créer un programme pour expliquer la chose à un tas de circuits : c'est là que les ennuis commencent...
Un autre problème est qu'une machine a besoin de *construire* un résultat.

Par exemple, en mathématiques, il est possible de démontrer qu'un objet existe tout en étant incapable de l'exhiber ce qui n'est pas possible pour la machine qui a besoin d'un raisonnement *constructiviste*. Ce problème est l'objet de recherches actuelles tendant à unifier langage mathématique et informatique : c'est assez compliqué c'est pourquoi nous ne ferons qu'évoquer ce problème.

2 2 Langage des expressions

Une expression, au niveau syntaxique, est formée de symboles, de noms, de parenthèses, de virgules, de noms de fonctions, de constantes, de noms d'opérateurs, etc. selon une *grammaire* (notion que nous étudierons dans un prochain module) qui en fixe les règles.
Par exemple, `2*(3+4)` est une *expression arithmétique* qui, après une phase d'*expansion*, peut être transformée en 14.
D'autre part, `a*(b+c)` est une *expression algébrique* qui permet de faire *abstraction* des valeurs particulières. Si le contexte nous permet de donner des valeurs aux arguments qui apparaissent dans l'expression. Il s'agit alors de la phase de *réduction*. Par exemple, si on fixe $a = 1$, $b = 2$, $c = 3$, l'expression réduite devient $1 * (2 + 3)$. On passe ensuite à la phase d'expansion.

Lorsqu'on travaille avec des expressions algébriques, il vaut mieux utiliser des fonctions, au sens mathématique du terme, c'est-à-dire qui à une liste d'arguments fixés associe une *unique* valeur. Ainsi l'évaluateur du langage fournira une valeur unique, ce qui lui fera plaisir.
Une fonction a une *signature* constituée de son *nom* et de son *type*. CAML par exemple effectue une inférence de type : il devine la signature de la fonction d'après l'expression associée. Par exemple on entre :

```
let f = fun k -> k*k;;
```

et l'interpréteur nous renvoie la signature de la fonction telle qu'il l'a comprise :

```
val f : int -> int = <fun>
```

c'est-à-dire : f est une fonction qui à un entier associe un autre entier.

Remarque

Pourquoi CAML a deviné qu'il s'agissait d'entiers ? C'est parce que l'opérateur $*$ ne permet que de travailler avec des entiers. Pour effectuer le produit de deux flottants, il faut utilisé l'étoile pointée `*.` :

```
# let g = fun x -> x *. x;;
val g : float -> float = <fun>
```

On peut *appliquer* la fonction f pour obtenir des valeurs particulières :

```
# f(3);;
- : int = 9
```

mais il faut bien rester dans le domaine :

```
# f(1.2);;
Characters 1-6:
  f(1.2);;
   ^^^^^
Error: This expression has type float but an expression was expected of type int
```

On peut utiliser cette fonction dans une fonction plus complexe :

```
# let h = fun a b -> f (b-a);;
val h : int -> int -> int = <fun>
# h 7 3;;
- : int = 16
```

Comment a été calculé $h\ 7\ 3$?

```
h 7 3 ⟶ f (7-3) (* c'est la phase d'expansion de la définition de h *)
f (7-3) ⟶ f 4 (* c'est la phase de réduction après évaluation du - *)
f 4 ⟶ 4*4 (* expansion de la définition de f *)
4*4 ⟶ 16 (* réduction après évaluation de * *)
```

Haskell CURRY

Aparté

CURRYFICATION

Le mathématicien américain Haskell CURRY (1900-1982) a largement développé le **lambda-calcul** d'Alonzo CHURCH qui constitue la base théorique de la programmation fonctionnelle. Une fonction curryfiée est une fonction de plusieurs variables transformée en une fonction d'une seule variable renvoyant une fonction.
Par exemple, créons une fonction curryfiée qui semble renvoyer la somme de deux entiers :

```
# let biplus x y = x+y;;
val biplus : int -> int -> int = <fun>
```

Cette définition est différente de celle-ci, plus habituelle :

```
# let monoplus(x,y) = x+y;;
val monoplus : int * int -> int = <fun>
```

mais est identique à celle d'un opérateur infixe :

```
# let (++) x y = x+y ;;
val ( ++ ) : int -> int -> int = <fun>
```

Notons $\mathcal{F}(D,A)$ l'ensemble des fonctions de D vers A.
Alors `biplus` $\in \mathcal{F}\big(\mathbb{N},\mathcal{F}(\mathbb{N},\mathbb{N})\big)$ mais `monoplus` $\in \mathcal{F}(\mathbb{N}\times\mathbb{N},\mathbb{N})$: ce n'est pas pareil !
En effet, **`biplus 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 :

```
# let plus3 = biplus 3;;
val plus3 : int -> int = <fun>
# plus3 5;;
- : int = 8
```

2 3 Emploi d'expressions intermédiaires

Plutôt que de définir deux fonctions séparées, on peut vouloir définir la fonction intermédiaire à l'intérieur d'une fonction principale. On utilise alors la structure *soit...dans...* :

```
let hh a b =
  let ff k =
    fun k -> k*k
```

```
in
ff (b-a);;
```

La fonction `ff` est ici définie localement.

Remarque

On peut aussi utiliser le *soit...dans...* hors de la définition d'une fonction pour éviter des redondances.
Par exemple, au lieu d'écrire `(1 + (b-a))* (2 - (b-a))` on pourra écrire :
`let s = b-a in (1+s)* (2-s).`

3 Récursion

3 1 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.

3 2 Induction mathématique

Nous en avons parlé au moment de l'étude du problème des tours de Hanoï :

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 $n \geqslant n_0$, $\mathcal{P}_n \Longrightarrow \mathcal{P}_{n+1}$ (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 quelques exemples...

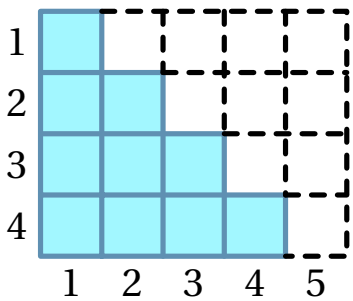
3 2 1 Génétique syldave

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 maldie »
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 $k+1^{e}$ génération seront également infectés, ce qui se traduit par
   $$\mathcal{P}(k) \text{ vraie} \Longrightarrow \mathcal{P}(k+1) \text{ vraie}$$
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.

3 2 2 Jouons aux cubes

Voici un test de fin d'étude maternelle en Syldavie : prenez un cube, placez en-dessous deux autres cubes, et encore en-dessous trois cubes, etc.



Combien y a-t-il de cubes bleus au total sur le dessin ci-dessus ? On peut encore les compter à la main, mais que faire si je vous demande le nombre de cubes lorsqu'on a placé 100 rangées ? n rangées ?
Le dessin nous donne une idée : si nous complétons la figure pour former un rectangle, il y a deux fois plus de cubes, mais maintenant nous pouvons les compter. Il y en a en effet $\frac{4\times(4+1)}{2}$, et donc

$$1+2+3+4=\frac{4\times(4+1)}{2}$$

Reprenons la méthode adoptée pour étudier la génétique syldave :

1. Nous allons essayer de prouver que la propriété suivante est vraie pour tout entier naturel non nul n
   $$\mathcal{P}(n) \text{ : } \text{« } 1+2+3+\cdots+n=\frac{n(n+1)}{2} \text{ »}$$
2. Il est facile de vérifier que $1=\frac{1(1+1)}{2}$, donc la deuxième étape de notre raisonnement est vérifiée
   $$\mathcal{P}(1) \text{ est vraie}$$
3. Supposons qu'une « génération », appelons-la par exemple la k^{e}, soit « infectée ». Plus sobrement on dira : soit k un entier supérieur à 1. Supposons que $\mathcal{P}(k)$ soit vraie et essayons alors de montrer que cela implique que la génération suivante, la $k+1^{e}$, sera elle aussi infectée, c'est à dire
   $$\mathcal{P}(k) \text{ vraie} \Longrightarrow \mathcal{P}(k+1) \text{ vraie}$$

Il s'agit donc de calculer $1+2+3+\cdots+k+(k+1)$ sachant que $1+2+3+\cdots+k=\dfrac{k(k+1)}{2}$, or

$$\begin{aligned}
1+2+3+\cdots+k+(k+1) &= \underbrace{1+2+3+\cdots+k}+(k+1)\\
&= \frac{k(k+1)}{2}+(k+1)\\
&= (k+1)\left(\frac{k}{2}+1\right)\\
&= (k+1)\left(\frac{k+2}{2}\right)\\
&= \frac{(k+1)(k+1+1)}{2}
\end{aligned}$$

Nous en déduisons que $\mathcal{P}(k+1)$ est vraie elle aussi.

4. Nous avons vérifié que la propriété était vraie au rang 1 et qu'elle était héréditaire. Nous allons donc en déduire que la propriété sera toujours vraie, quelque soit l'entier naturel non nul n grâce au théorème admis suivant

3.3 Un exemple de 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 :

$$n! = \begin{cases} 1 \text{ si n=1} \\ n \times (n-1)! \text{ si n>1} \end{cases}$$

ou encore `fac(n): si n=1 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=1 alors 1 sinon (fac (n+1))/(n+1)`
Quel est le problème ?
Voici différentes syntaxes utilisables avec CAML :

```
let rec fac = function
  |1 -> 1
  |n -> n*fac(n-1);;

let rec fac n =
  if n = 1 then 1 else n*fac(n-1);;

let rec fac n =
  match n with
    |1 -> 1
    |_ -> n*fac (n-1);;

let fac n =
  let rec temp = function
    |1,acc -> acc
    |n,acc -> temp(n-1 , n*acc)
  in
  temp(n,1);;
```

Nous verrons lors de l'étude des ensembles et des listes qu'une exploration récursive de ces objets est naturelle : on regarde la tête puis on applique la fonction à la queue...
De même, de nombreux ensembles (ou types) peuvent être définis récursivement. Par exemple, un mot est soit une lettre, soit une lettre suivie d'un mot...
La pratique nous rendra ces schémas de pensée naturels.

EXERCICES

Exercice 1 - 1

Spécifiez puis définissez une fonction `poly` telle que `poly a b c d x` donne la valeur de la fonction qui à x associe ax^3+bx^2+cx+d.
Faites de même avec la fonction `bipoly` qui donne la valeur de la fonction qui à x associe ax^4+bx^2+c.

Exercice 1 - 2

Donnez une définition possible de la fonction racine carrée.

Exercice 1 - 3

Écrire une fonction de x et de y qui donne la valeur de $x(1+xy)^2+y(1-y)+(1+xy)(1-y)$ en évitant les redondances.

Exercice 1 - 4 Induction

Les images des entiers naturels par la fonction $n \mapsto n^2-n+41$ sont des nombres premiers : vrai ou faux ?

Exercice 1 - 5 Somme

Définissez récursivement la somme des entiers de 1 à n puis leur produit puis la somme des carrés des entiers de 1 à n puis la somme de n'importe quelle fonction appliquée aux entiers de 1 à n.

Exercice 1 - 6 Division euclidienne

Déterminer un algorithme récursif qui renvoie le quotient de deux entiers naturels a et b, le deuxième étant non nul.
Déterminer un algorithme récursif qui renvoie le reste entier de deux entiers naturels non nuls a et b, le deuxième étant non nul.

Exercice 1 - 7 Somme de chiffres

Donnez un algorithme récursif qui calcule la somme des chiffres d'un entier naturel n écrit en base 10.

Exercice 1 - 8 Base 2

On dispose des fonctions donnant le quotient et le reste de la division euclidienne de deux entiers. Définissez récursivement une fonction qui a un nombre de 4 chiffres écrit en base 2 renvoie son écriture en base 10.

Exercice 1 - 9 Maximum d'une liste

Nous étudierons bientôt les listes. Disons pour le moment qu'il s'agit d'une structure ordonnée dont les éléments peuvent être répétés.
On définit deux fonctions **`tete`** et **`queue`** qui renvoient respectivement le premier élément d'une liste et la liste privée de sa tête. La syntaxe dépend des langages.
Écrivez à présent une fonction récursive qui renvoie le maximum d'une liste de nombres.

Exercice 1 - 10 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 !).

1. Il y en a un caché entre 1 et 10 : trouvez-le...

2. Écrivez un algorithme récursif qui donne la liste des diviseurs d'un entier autres que lui-même.
3. Écrivez un algorithme qui calcule la somme des éléments d'une liste.
4. Écrivez un algorithme qui détermine si un nombre entier est parfait.
5. Écrivez un algorithme récursif qui donne la liste des nombres parfaits entre 1 et un nombre n donné.

Exercice 1 - 11 Décomposition en base 2

Une méthode pour obtenir l'écriture en base 2 d'un nombre est d'effectuer des divisions successives. Par exemple pour 11 :

$$\begin{array}{r|l} 11 & 2 \\ \hline 1 & 5 \end{array} \qquad \begin{array}{r|l} 5 & 2 \\ \hline 1 & 2 \end{array} \qquad \begin{array}{r|l} 2 & 2 \\ \hline 0 & 1 \end{array} \qquad \begin{array}{r|l} 1 & 2 \\ \hline 1 & 0 \end{array}$$

$$\begin{aligned} 11 &= \Big(2\times 5+1\Big) \\ &= \Big(2\times\big(2\times 2+1\big)+1\Big) \\ &= \Big(2\times\big(2\times(2\times 1)+1\big)+1\Big) \\ &= \Big(2\times(2^2+1)+1\Big) \\ &= 2^3+2+1 \\ &= 1\times 2^3+0\times 2^2+1\times 2^1+1\times 2^0 \end{aligned}$$

L'écriture de 11 en base 2 est donc 1011 : c'est la liste des restes obtenus mais dans l'ordre inverse.
La méthode est facilement généralisable.
Écrivez un algorithme récursif qui donne la décomposition d'un entier en base 2 sous forme d'une liste.

Exercice 1 - 12

On se propose de démontrer que tous les étudiants en informatique ont le même âge et, pour cela, on note $P(n)$ l'affirmation
« *si on choisit n étudiants en informatique* ($n \in \mathbb{N}^*$), *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 $n \geq 1$. **Y a-t-il une erreur dans le raisonnement ?**

Exercice 1 - 13

Prouvez par récurrence que l'algorithme suivant permet de calculer le carré d'un entier :

```
Fonction Carré( n: entier positif ): entier positif
Début
    sq ← 0
    Pour i De 1 Jusque n Faire
        sq ← sq+2*i-1
    FinPour
    Retourner sq
Fin
```

CHAPITRE 2

Ensembles : une approche fonctionnelle

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.

En informatique, c'est aussi une notion essentielle : un *type* est en effet un ensemble au sens mathématique du terme, comme les types de base que vous avez vu en ALGO-1 (entiers, chaînes, booléens) ou les types construits que nous allons créer selon nos besoins.

Nous allons prendre une approche fonctionnelle de la notion d'ensemble et des opérations associées ce qui nous permettra d'implémenter directement nos travaux dans le langage CAML.

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

Rue de Strasbourg,13 Mai 2003

1 Définition

1.1 Définition récursive

Définition 2 - 1

Un ensemble d'*éléments* est :

- soit l'ensemble vide, noté $\{\}$ (ou bien $\emptyset$) ;
- soit construit en « ajoutant » un élément x à un ensemble E. On notera alors cette opération $x \oplus E$.

Cette définition est *récursive* comme nous l'avons vu au chapitre précédent. Elle nous permettra de définir un type construit le plus abstrait possible avec CAML.

1.2 Avec Caml

```
type 'a ens =
  | Vide
  | Ens of ('a * 'a ens);;
```

L'ensemble vide et l'opération $\oplus$ sont les *constructeurs* du type construit ens. Les éléments sont d'un type quelconque 'a.
On peut par exemple construire des ensembles d'entiers :

```
# Ens(1,Vide);;
- : int ens = Ens (1, Vide)
# Ens(3,Ens(2,Ens(1,Vide)));;
- : int ens = Ens (3, Ens (2, Ens (1, Vide)))
# let e = Ens(3,Ens(2,Ens(1,Vide)));;
val e : int ens = Ens (3, Ens (2, Ens (1, Vide)))
# Ens(4,e);;
- : int ens = Ens (4, Ens (3, Ens (2, Ens (1, Vide))))
```

Nous reviendrons en fin de chapitre sur la programmation des différentes opérations sur les ensembles avec CAML. Revenons au cas général.

2 Propriétés et notations

Définition 2 - 2

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

Ça paraît idiot comme axiome mais cela permet de bien cerner le problème : c'est la notion d'appartenance d'un élément à un ensemble qui est primordiale.
Une première conséquence est que l'ordre d'apparition des éléments dans la construction des ensembles n'est pas important :

$$x \oplus \{y\} = y \oplus \{x\}$$

Notations

On notera $x \in A$ pour signifier que x est un élément de l'ensemble A. Cette notation 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 ∈.

Par souci de simplification, nous noterons aussi :

- le *singleton* $x \oplus \{\}$ sous la forme $\{x\}$;
- l'ensemble $x \oplus (y \oplus (z \oplus \{\}))$ sous la forme $\{x, y, z\}$.

Ainsi

$$x \oplus (y \oplus \{\}) = x \oplus \{y\} = \{x, y\}$$

Recherche

On considère les ensembles :
$E_1 = \{a, b, c\}$, $E_2 = \{b, c, a\}$, $E_3 = \{b, a, a, c, b\}$, $E_4 = \{c, c, c, a, b, a, c, b, b, b\}$, $E_5 = c \oplus E_2$.
Quels sont, parmi ces ensembles, ceux qui sont égaux ?

L'opérateur ⊕ n'est pas *commutatif* car l'opérande de gauche est un élément et celui de droite un ensemble : leurs rôles ne peuvent pas être échangés.

Remarque

On pourrait cependant modifier notre définition et ne pas accepter des ensembles du type $\{1, 2, 2, 3\}$, cet ensemble étant pour nous égal à $\{1, 2, 3\}$. Cependant, cela n'est pas gênant et correspond à la construction proposée avec CAML. Il faudra malgré tout faire attention à la définition du *cardinal* d'un ensemble que nous allons voir plus loin.

3 Extension - compréhension

3.1 Définition par extension

On peut définir un ensemble *par extension*, c'est-à-dire en écrivant tous ses éléments :

$$A = \{1, 3, 5, 7, 9\}$$

C'est pourquoi on a parlé d'axiome d'*extensionnalité* : on compare les éléments des ensembles.
Vous aurez remarqué que des ensembles égaux peuvent avoir des définition en extension différentes.
Cependant, il peut s'avérer fastidieux de définir un ensemble par extension. Nous allons introduire un peu de vocabulaire de la *logique des prédicats* pour définir certains ensembles plus facilement.

3 2 Un très rapide survol de la logique

Nous consacrerons plusieurs chapitres à l'étude de la logique qui est un des piliers de l'informatique. Cependant, nous allons voir rapidement quelques notions simples en avant-première dont nous avons besoin tout de suite.
Il existe un type (donc un ensemble..) défini dans tous les langages informatiques car il en constitue la base. Il s'agit du type *booléen* qui ne contient que deux éléments : VRAI et FAUX (ou TRUE et FALSE ou 0 et 1 ou ...). Nous noterons cet ensemble $\mathcal{B}_2 = \{\text{VRAI}, \text{FAUX}\}$.
Si E est un ensemble quelconque, un *prédicat* est une fonction de E dans l'ensemble $\mathcal{B}_2$.

Exemple

Par exemple « *est un entier pair* » est un prédicat défini sur l'ensemble $\mathbb{N}$ des entiers naturels que nous noterons P. Alors P(2) = VRAI et P(3) = FAUX.

Nous serons plus rigoureux un peu plus tard afin de pouvoir comprendre le fonctionnement de la machine. Nous définirons en particulier la notion de *proposition logique*. Pour l'instant cette approche naïve sera suffisante.
Soit un prédicat P défini sur $\mathbb{N}$ par $P(x) =$ « $x^2 = 2$ ».
Quelque soit l'entier x, on a $x^2 \neq 2$. On notera cette propriété par :

$$\forall x, x \in \mathbb{N} \text{ ET } P(x) = \text{FAUX}$$

Remarque

Nous noterons ceci plus tard : $\forall x(x \in \mathbb{N} \wedge \neg P(x))$...

$\forall$ est un *quantificateur universel* (c'est un A à l'envers, car en anglais, ce symbole se lit *for All* et a été introduit par l'anglais RUSSEL...).
Soit maintenant le prédicat R défini sur $\mathbb{N}$ par $R(x) =$ « $x^2 = 4$ ».
Il existe au moins un entier dont l'image par R est VRAI : il s'agit de 2.
On note alors :

$$\exists x, x \in \mathbb{N} \text{ ET } R(x) = \text{VRAI}$$

$\exists$ est l'autre quantificateur universel introduit par RUSSEL (un E à l'envers comme dans *it Exists*).
On peut même préciser que cet entier est *unique* en faisant suivre le quantificateur d'un point d'exclamation :

$$\exists! x, x \in \mathbb{N} \text{ ET } R(x) = \text{VRAI}$$

3 3 Définition par compréhension

Un entier pair est un entier multiple de 2.
Nous pouvons à présent définir *par compréhension* l'ensemble $\mathbb{N}_2$ des entiers naturels pairs :

$$\mathbb{N}_2 = \{x \mid (x \in \mathbb{N}) \text{ ET } (\exists n, n \in \mathbb{N} \text{ ET } x = 2n)\}$$

Recherche

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

3 4 Définition récursive

On sait que 0 est un entier naturel pair.
On peut montrer par récurrence que si l'entier x est pair, alors l'entier $x+2$ l'est aussi.
On définit ainsi l'ensemble $\mathbb{N}_2$ précédent de manière récursive par la donnée :

- d'une base : 0 est un entier pair (c'est l'élément ou l'ensemble qui servira à construire *tous* les autres éléments à l'aide de la règle de récurrence) ;
- d'une règle de récurrence : ici, si x est pair, alors $x+2$ l'est aussi.

La base et la règle de récurrence constituent les CONSTRUCTEURS de l'ensemble.
Informatiquement, les langages fonctionnels permettent de créer des types (donc des ensembles...) construits récursivement comme nous le verrons avec CAML.

4 Inclusion

Définition 2 - 3

Un ensemble désigné par B est **contenu** ou **inclus** dans l'ensemble désigné par A ou encore est une **partie** de l'ensemble A si, et seulement si, tout élément de B est aussi un élément de A. On note alors $B \subseteq A$ ou $A \supseteq B$.
On dit aussi que B est **une partie ou un sous ensemble de** A.

La définition précédente affirme que $\{\}$ est inclus dans n'importe quel ensemble A car, si ce n'était pas le cas, l'ensemble vide possèderait au moins un élément qui n'appartient pas à A ce qui est absurde. On notera aussi que tout ensemble est inclus dans lui même. On retiendra donc que si A désigne un ensemble **on a toujours**

$$\{\} \subseteq A \text{ et } A \subseteq A$$

Remarque

Pratiquement tous les auteurs d'ouvrages mathématiques en langue française utilisent le symbole $\subset$ pour définir l'inclusion (large), nous estimons qu'ils le font à tort. Nous ne nous alignons pas sur une notation anglo-saxonne, nous ne faisons qu'utiliser la notation du système internationnal. Pour nous, le symbole $\subset$ désigne l'inclusion stricte en ce sens que si $B \subset A$ on ne peut avoir $B = A$ et par conséquent l'énoncé $A \subset A$ est faux. Pour éviter les confusions toujours possibles nous utiliserons le symbole $\subsetneq$ ou $\subsetneqq$ pour désigner l'inclusion stricte

$$B \subsetneq A \text{ signifie } (B \subseteq A \text{ et } B \neq A)$$

qu'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 $B \neq \{\}$ (il est nécessaire que A ne soit pas vide et ne soit pas un singleton).

Comme nous l'avons déjà indiqué, un ensemble est toujours inclus dans lui même. Pour tout ensemble A il est correct d'écrire $A \subseteq A$. La définition de l'égalité de deux ensembles s'écrit maintenant

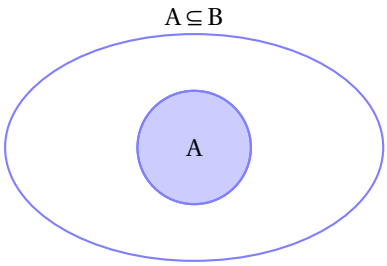
$$[A \subseteq B \text{ et } B \subseteq A] \text{ si, et seulement si, } A = B$$

À retenir

Cette propriété est à retenir. Pour démontrer l'égalité de deux ensembles, on n'a pratiquement pas d'autre méthode que de démontrer la double inclusion. Notons aussi le résultat évident

$$\text{SI } [C \subseteq B \text{ et } B \subseteq A] \text{ ALORS } C \subseteq A$$

Pour guider les raisonnements ou pour « imaginer » les ensembles en présence on est amené à schématiser les ensembles par une ligne fermée sans point double, une « patate » peut très bien faire l'affaire mais un cercle ou un rectangle sont aussi d'excellentes solutions. Ces figures sont appelées diagrammes d'Euler-Venn ou *diagrammes de Venn*. Pour éviter les confusions, il est convenu que les éléments de l'ensemble représenté « sont tous à l'intérieur du dessin et pas sur la frontière ». Pour mettre en exergue un ou plusieurs éléments, on utilise des points ou croix et pour représenter l'ensemble vide, il n'y a aucune bonne solution.
Les patates permettent d'illustrer l'inclusion :



5 Parties d'un ensemble

5 1 Définition

Définition 2 - 4

Nous admettons que si E désigne un ensemble alors **l'ensemble des parties de** E est un ensemble (mathématiquement parlant) noté $\mathcal{P}(E)$ mais aussi $\mathbb{P}(E)$. Cet ensemble peut s'écrire en « pseudo-compréhension » par :

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

Nous précisons bien « pseudo-compréhension » car nous ne pouvons indiquer à quel ensemble appartient X puisque nous cherchons à définir cet ensemble. Il faut immédiatement retenir l'équivalence

$$X \in \mathcal{P}(E) \text{ si, et seulement si, } X \subseteq E$$

qui seule permet de vérifier que X est bien un élément de $\mathcal{P}(E)$ car il ne faut surtout pas

croire que notre esprit soit capable de « maîtriser » un ensemble de parties quasi immédiatement. Pour imaginer un peu plus ce qu'est $\mathcal{P}(E)$ on peut dire que $\mathcal{P}(E)$ est l'**ensemble** des solutions du problème $X \subseteq E$ où X est l'inconnue. On remarque alors que $\{\}$ et E sont des solutions évidentes et donc que $\mathcal{P}(E)$ ne peut être vide. Si $E = \{\}$, $\mathcal{P}(E)$ ne contient qu'un seul élément qui est $\{\}$; $\mathcal{P}(\{\}) = \{\{\}\}$ et on se persuadera que $\{\{\}\}$ n'est pas l'ensemble vide, c'est parfaitement clair, entre les deux accolades « il y a quelque chose » qui existe et qui est l'ensemble vide, $\{\{\}\}$ est un singleton ! Si $E \neq \{\}$ alors $\mathcal{P}(E)$ possède au moins deux éléments distincts $\{\}$ et E :

$$\mathcal{P}(E) = \left\{\{\}, \ldots, E\right\}$$

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

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

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 :

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

On comprend tout de suite la difficulté d'imaginer l'ensemble des parties d'un ensemble d'autant plus que les diagrammes de Venn ne peuvent nous venir en aide.

5 2 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 $E = \{\}$, alors $\mathcal{P}(E) = \left\{\{\}\right\}$.

Sinon, on peut écrire E sous la forme $x \oplus F$ avec x un élément quelconque de E.

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

6 Opérations

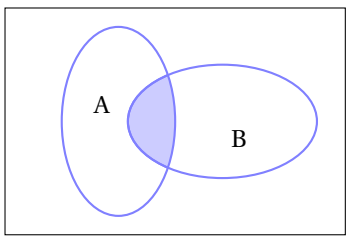
6 1 Intersection

Définition 2 - 5

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

$$A \cap B = \left\{ x \in E \mid x \in A \text{ et } x \in B \right\}$$

C'est une partie de E. L'intersection est une *loi de composition* (ou opération) *interne* dans $\mathcal{P}(E)$ qui est *associative, commutative* et qui admet E comme *élément neutre* ($A \cap E = A$), de plus, tout élément est idempotent ($A \cap A = A$). Lorsque $A \cap B = \{\}$ on dit que les ensembles A et B sont **disjoints**.



$A \cap B$

Remarque

On a l'équivalence $A \cap B = B \Leftrightarrow B \subseteq A$, en effet, supposons en premier que $A \cap B = B$, par définition $A \cap B \subseteq A$ et on obtient tout de suite $B \subseteq A$. Supposons maintenant que $B \subseteq A$, les éléments communs à A et B sont exactement les éléments de B et donc $A \cap B = B$.

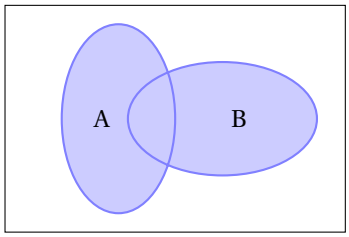
6 2 Union

Définition 2 - 6

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

$$A \cup B = \left\{ x \in E \mid x \in A \text{ ou } x \in B \right\}$$

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 $(A \cap B) \subseteq (A \cup B)$. L'union est une loi de composition (ou opération) interne dans $\mathcal{P}(E)$ qui est associative, commutative et qui admet $\{\}$ comme élément neutre ($A \cup \{\} = A$), de plus tout élément est idempotent ($A \cup A = A$).



$A \cup B$

Remarque

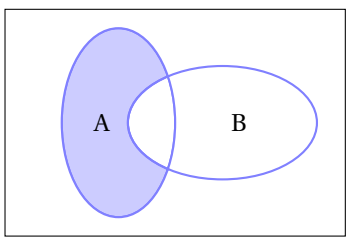
On a l'équivalence $A \cup B = A \Leftrightarrow B \subseteq A$. En effet, si $A \cup B = A$ et comme $B \subseteq (A \cup B)$, on obtient $B \subseteq A$. Si $B \subseteq A$, tout élément de B est aussi un élément de A donc tout élément de $A \cup B$ est aussi un élément de A ; on obtient alors $(A \cup B) \subseteq A$ et comme $A \subseteq (A \cup B)$ on conclut que $A \cup B = A$.

6 3 Différence

Définition 2 - 7

La **différence** des ensembles A et B est l'ensemble

$$A \setminus B = \left\{ x \in E \mid x \in A \text{ et } x \notin B \right\}$$



$A \setminus B$

Remarque

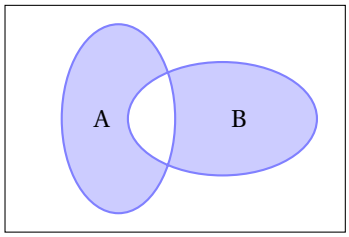
Certains préfèrent utiliser la notation $A - B$ à la place de $A \setminus B$.

6 4 Différence symétrique

Définition 2 - 8

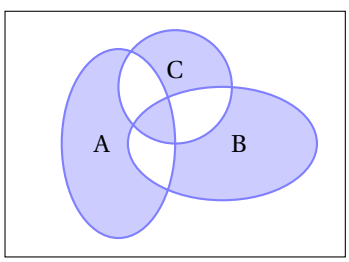
La **différence symétrique** des ensembles A et B est l'ensemble :

$$A \triangle B = \left\{ x \in E \mid x \in A \setminus B \text{ ou } x \in B \setminus A \right\} = (A \setminus B) \cup (B \setminus A)$$
$$A \triangle B = \left\{ x \in E \mid x \in A \cup B \text{ et } x \notin A \cap B \right\} = (A \cup B) \setminus (A \cap B)$$



$A \triangle B$

L'opération $\triangle$ est associative (nous démontrons cette affirmation plus loin), commutative, admet $\{\}$ comme élément neutre et tout élément A est son propre symétrique ($A \triangle A = \{\}$). On résume toutes ces propriétés en disant que $\mathcal{P}(E)$ muni de la loi $\triangle$ est un groupe [a] commutatif. Vous pourrez vérifier que $A \triangle B \triangle C$ peut être représenté par :



$A \triangle B \triangle C$

Remarque

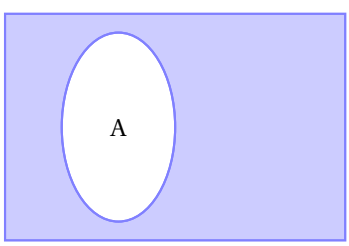
La différence symétrique correspond au « ou exclusif ».

6 5 Complémentaire

Définition 2 - 9

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

$$\complement_E A = E - A = \left\{ x \in E \mid x \notin A \right\}$$



$\complement_E A$

a. Les structures de groupe, anneau et corps doivent devenir pour vous des notions familières mais ne peuvent faire l'objet de questions à un DS.

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

6 6 Lois de De Morgan

Théorème 2 - 1

Lois de De Morgan

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

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.

Remarque

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

6 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	$A \cup \{\} = A$	$A \cap E = A$
Involution	$\overline{\overline{A}} = A$	
Complémentaire	$A \cup \overline{A} = E$ $\overline{E} = \{\}$	$A \cap \overline{A} = \{\}$ $\overline{\{\}} = E$
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 $\{\}$ respectivement par $\cup$, $\cap$ $\{\}$ et E.

Le *principe de dualité* affirme si e est vérifiée pour tout élément de $\mathcal{P}(E)$, alors sa duale $e^\star$ aussi.

6 8 Appartenance d'un élément à un ensemble

Cette fonction sera très utile en programmation. On prend un élément x et un ensemble en argument et on renvoie un élément de $\mathcal{B}_2$.

7 Partition d'un ensemble

E désignant ici un ensemble non vide. Soit P, une partie non vide de $\mathcal{P}(\mathrm{E})$ (on a donc $\mathrm{P} \subseteq \mathcal{P}(\mathrm{E})$ et $\mathrm{P} \neq \emptyset$) 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 $\mathrm{P} = \{\mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_k\}$ avec

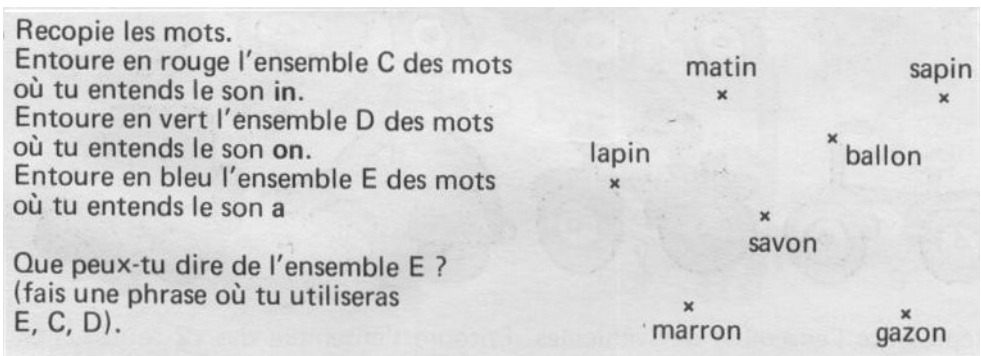
$$\mathrm{A}_i \subseteq \mathrm{E}, \mathrm{A}_i \neq \emptyset, \mathrm{A}_i \cap \mathrm{A}_j = \emptyset \text{ si } i \neq j \text{ et } \bigcup_{i=1}^{k} \mathrm{A}_i = \mathrm{E}$$

On peut imaginer une partition de E comme un découpage de E, aucun des morceaux n'étant 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 !

8 Fonction caractéristique

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 $\mathbb{1}_B$.

Je vous donne un coup de main :

- $\mathbb{1}_{\overline{A}} = 1 - \mathbb{1}_A$, 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 [b] $\overline{X\cup Y}=\overline{X}\cap\overline{Y}$ 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 qu'il faut connaître, certaines sont évidentes et d'autres pas, nous reprenons des résultats déjà énoncés et nous ne donnons que quelques démonstrations, le lecteur étant invité à les reprendre et à en faire d'autres. Il doit être clair que tous les ensembles utilisés sont des parties de l'ensemble E !

– $\cup$ 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 $k\in\mathbb{N}^*$,

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

– $A\subseteq B\iff A\cup B=B\iff A\cap B=A\iff A-B=\{\}$

– $A\subseteq(A\cup B),(A\cap B)\subseteq A$

– $A-A=A\triangle A=A\cap\{\}=\{\}$

– $A\cap A=A\cup A=A\cup\{\}=A\cap E=A\triangle\{\}=A$

– $A\cap(A\cup B)=A=A\cup(A\cap B)$

– $\overline{\left(\overline{A}\right)}=A,\overline{A}\cap A=\{\},\overline{A}\cup A=E,\complement_E\{\}=\overline{\{\}}=E,\complement_E E=\overline{E}=\{\}$

b. C'est une des formules de De Morgan.

– $\cap$ 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 :

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

donne

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

On vérifie que $\mathbb{1}_{(A\triangle B)\triangle C}$ 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..

9 Produit cartésien

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 :

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

– l'ensemble des plats de résistance :

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

– l'ensemble des desserts :

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

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.

Généralisons sans trop entrer dans les détails. La notion de couple se généralise naturellement en notion de triplet, de quadruplet,..., de n-uplet, un n-uplet étant noté $\underbrace{(a_1, a_2, \cdots, a_n)}_{n \text{ éléments}}$.

L'égalité de deux n-uplets étant définie par

$$(a_1, a_2, \cdots, a_n) = (a'_1, a'_2, \cdots, a'_n) \Leftrightarrow a_i = a'_i \text{ pour tout } i$$

E_1 et E_2 désignant deux ensembles, nous admettons que l'ensemble des couples (a_1, a_2) avec $a_1 \in E_1$ et $a_2 \in E_2$ est bien un ensemble que l'on note $E_1 \times E_2$ appelé **produit cartésien** des ensembles E_1 et E_2, une écriture en pseudo compréhension étant

$$E_1 \times E_2 = \{(a_1, a_2) \mid a_1 \in E_1 \text{ et } a_2 \in E_2\}$$

On définit de même le produit cartésien des n ensembles ($n \in \mathbb{N}^*$ et $n \geq 2$) $E_{i\in\{1,2,3,\cdots,n\}}$ comme étant l'ensemble :

$$E_1 \times E_2 \times \cdots \times E_n = \prod_{i=1}^{n} E_i = \{(a_1, a_2, \cdots, a_n) \mid a_i \in E_i\}$$

et lorsque tous les ensembles E_i sont égaux à E on note $E \times E \times \cdots \times E$ par E^n.

Pour se représenter le produit cartésien $E \times F$ 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).

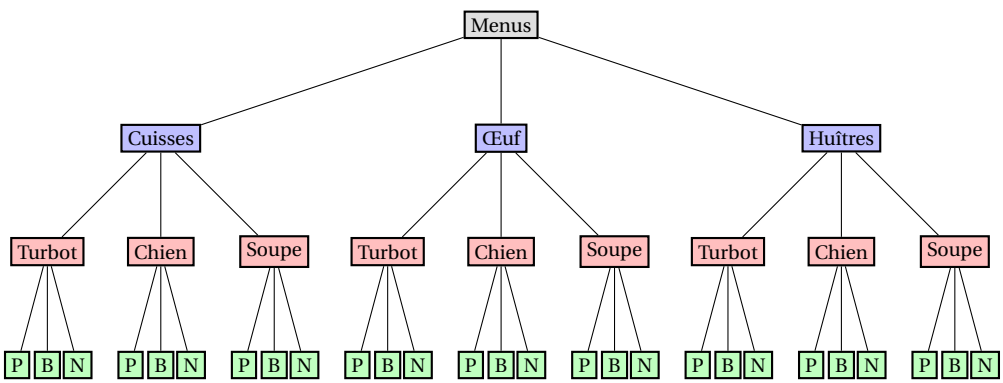
$$E \times F = \{(a,1), (a,2), \cdots, (c,5)\}$$

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

×	**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 notre menu, il faudrait un tableau en 3D !

On utilise plutôt dans ce cas un arbre :



Un menu est donc un triplet élément de $E \times P \times D$. On peut ensuite s'intéresser aux menus végétariens, aux menus sans viande, etc. Nous nous intéresserons par la suite à la manière de les extraire, de les compter, etc.

Recherche

Comment définir récursivement le produit cartésien de deux ensembles E_1 et E_2 ? Cela nous sera utile pour programmer en CAML...

10 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 **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 :

$$\mathrm{Card}\,(\mathrm{E}) \text{ ou } |\mathrm{E}| \text{ ou } \#\mathrm{E}$$

Par exemple $|\varnothing| = 0$, 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** :

Propriétés 2 - 1

- Si $\mathrm{A} \subseteq \mathrm{B}$ alors $|\mathrm{A}| \leq |\mathrm{B}|$. Une conséquence intéressante est celle ci : si $\mathrm{A} \subseteq \mathrm{B}$ avec $|\mathrm{A}| = |\mathrm{B}|$ alors $\mathrm{A} = \mathrm{B}$.
- $|\mathrm{A} - \mathrm{B}| \leq |\mathrm{A}|$.
- $|\mathrm{A} \cup \mathrm{B}| = |\mathrm{A}| + |\mathrm{B}| - |\mathrm{A} \cap \mathrm{B}|$
- **Si on a** $\mathrm{E}_i \cap \mathrm{E}_j = \varnothing$ **lorsque** $i \neq j$, alors $\left|\bigcup_{i=1}^{n} \mathrm{E}_i\right| = \sum_{i=1}^{n} |\mathrm{E}_i|$.
- $|\mathrm{E}_1 \times \mathrm{E}_2| = |\mathrm{E}_1|.|\mathrm{E}_2|$ et

$$\begin{aligned} |\mathrm{E}_1 \times \mathrm{E}_2 \times \cdots \times \mathrm{E}_n| &= |\mathrm{E}_1|.|\mathrm{E}_2|.\cdots.|\mathrm{E}_n| \\ |\mathrm{E}^n| &= (|\mathrm{E}|)^n \end{aligned}$$

- La formule suivante est absolument à connaître :

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

  Nous la démontrerons en exercice.

Recherche

Comment calculer récursivement le cardinal d'un ensemble ?

EXERCICES

Papier - crayon

Exercice 2 - 1

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 - 2

$E = \{0, 1, 2, 3, 4, 5, 6\}$. 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 - 3

$E = \{0, 1, 2, 3, 4\}$. 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 ») :

$$\in, \ni, \subseteq, \supseteq, =, \neq, \subsetneq, \cdots$$

$2 \cdots E$, $\{2, 3\} \cdots E$, $\{2\} \cdots E$

$\{2, 3, 4\} \cdots \{4, 3, 2\}$, $\{2, 3, 4\} \cdots \{4, 3, 0\}$

$\{\} \cdots E$, $E \cdots E$, $E \cdots \{E\}$, $\{\} \cdots \{E\}$

$E \cdots \{0, 1, 2, 3, \ldots, 10\}$, $\{0, 1, 2, 3, 4, 5\} \cdots E$

Exercice 2 - 4

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 en extension de manière formelle puis Déterminez les éléments de Ω qui vérifient les conditions suivantes :

1. X et B sont disjoints ; **2.** $X \subseteq D$ et $X \not\subseteq B$; **3.** $X \subseteq A$ et $X \not\subseteq C$; **4.** $X \subseteq C$ et $X \not\subseteq A$.

Exercice 2 - 5

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 - 6

On considère les parties de $\mathbb{N}$ suivantes : $A = \{n \mid n \leqslant 6\}$, $B = \{n \mid 4 \leqslant n \leqslant 9\}$, $C = \{1,3,5,7,9\}$ et $D = \{2,3,5,7,8\}$. Déterminez :

1. $A \Delta B$ **2.** $B \Delta C$ **3.** $A \cap (B \Delta D)$ **4.** $(A \cap B) \Delta (A \cap D)$

Exercice 2 - 7

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 - 8

Démontrer que la différence symétrique est commutative, associative, simplifiable à gauche (si $A \Delta B = A \Delta C$ alors $B = C$) et que l'intersection est distributive sur la différence symétrique.

Exercice 2 - 9

Expliciter $\mathcal{P}(\mathbb{N}_n)$ pour pour $n \in \{1,2,3\}$. On rappelle la notation $\mathbb{N}_n = \{1,2,3,...,n\} = [\![1,n]\!]$.

Exercice 2 - 10

Expliciter $\mathcal{P}(\{a,\{a\}\})$.

Exercice 2 - 11

Expliciter $\mathcal{P}(\{\})$, $\mathcal{P}(\mathcal{P}(\{\}))$.

Exercice 2 - 12

Expliciter $\mathcal{P}(\mathcal{P}(\mathbb{N}_2))$.

Exercice 2 - 13

Donner cinq éléments de $\mathcal{P}(\mathcal{P}(\mathcal{P}(\{a,b,c\})))$.

Exercice 2 - 14

Démontrer par récurrence que si E possède n éléments alors $\mathcal{P}(\mathrm{E})$ possède 2^n éléments.

Exercice 2 - 15

Combien d'éléments $\mathcal{P}(\mathcal{P}(\mathcal{P}(\{a,b,c\})))$ contient-il ?

Exercice 2 - 16

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

Exercice 2 - 17

$\mathrm{A} = \{0,1,2,3\}$, $\mathrm{B} = \{a,b\}$ et $\mathrm{C} = \{1,2\}$. Donner quelques éléments de :

1. $\mathrm{A} \times \mathrm{B}$, $(\mathrm{A} \times \mathrm{B}) \times \mathrm{C}$, $\mathrm{A} \times \mathrm{B} \times \mathrm{C}$, $\mathrm{A} \times (\mathrm{B} \times \mathrm{C})$, $\mathrm{C} \times (\mathrm{A} \cup \mathrm{B})$, A^5
2. $\mathcal{P}(\mathrm{A}) \times \mathcal{P}(\mathrm{B})$
3. $\mathcal{P}(\mathrm{A} \times \mathrm{B})$
4. $\{\} \times \mathrm{A}$, $\mathrm{A} \times (\{\} \times \mathrm{B})$, $\{\{\}\} \times \mathrm{B}$

Exercice 2 - 18

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

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

Exercice 2 - 19

$E = \{a, b, c, d, e\}$

1. Donner 4 partitions de E.
2. Vrai ou Faux , $\{\{\}, \{a\}, \{b, c, e\}, \{d\}\}$ est une partition de E.
3. Vrai ou Faux , $\{\{a\}, \{b, c\}, \{d\}\}$ est une partition de E.

Exercice 2 - 20

1. On pose $A = \{a, b\}$ et $B = \{b, c\}$.
   a. Comparer $\mathcal{P}(A) \cap \mathcal{P}(B)$ et $\mathcal{P}(A \cap B)$.
   b. 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.

Avec Caml

On rappelle que l'on part d'un type construit :

```
type 'a ens =
  | Vide
  | Ens of ('a * 'a ens);;
```

et donc que l'ensemble $e = \{1, 2, 3\}$ se construit ainsi :

```
# let e = Ens(3,Ens(2,Ens(1,Vide)));;
val e : int ens = Ens (3, Ens (2, Ens (1, Vide)))
```

et l'on peut construire $\{1, 2, 3, 4\}$ à partir de e :

```
# Ens(4,e);;
- : int ens = Ens (4, Ens (3, Ens (2, Ens (1, Vide))))
```

Nous (vous...) allons (allez...) construire différentes fonctions en rapport avec le cours.

Dans ma grande bonté, je vous donne un premier exemple d'une fonction fondamentale : celle qui va tester l'appartenance d'un élément à un ensemble.

Nous commençons par spécifier la fonction :

`appartient : un élément, un ensemble d'éléments` $\longrightarrow$ `un booléen (* appartient(x,E)` $\Leftrightarrow$ $x \in E$ `*)`

Nous allons ensuite raisonner récursivement suivant la taille décroissante de l'ensemble :

- `appartient(x,{}) = FAUX`
- `appartient(x,t` $\oplus$ `Q) = (t=x) ou bien appartient(x,Q)`

Cela se traduit en CAML de la manière suivante :

```
let rec appartient el ensemble =
  match ensemble with
    |Vide -> false
    |Ens(t,q) ->
      if t = el then true
      else appartient el q;;
```

Ainsi, en reprenant l'exemple précédent :

```
# appartient 3 e;;
- : bool = true
# appartient 5 e;;
- : bool = false
```

À vous de jouer avec les fonctions suivantes :

1. `ajoute el ens` qui ajoute l'élément `el` à l'ensemble `ens`. On testera si l'élément appartient déjà à l'ensemble.
2. `union ens1 ens2` qui renvoie la réunion des deux ensembles `ens1` et `ens2`.
3. `cardinal` qui calcule le cardinal d'un ensemble : attention aux répétitions !
4. `max2 a b` qui renvoie le maximum entre `a` et `b` puis `max ens` qui renvoie le plus grand élément de l'ensemble `ens`.
5. `applique f ens` qui renvoie l'ensemble des images des éléments de `ens` par `f`.
6. `parties ens` qui renvoie l'ensemble des parties de l'ensemble `ens`.
7. `retire el ens` qui renvoie l'ensemble `ens` privé de l'élément `el`.
8. On rappelle qu'un prédicat est une fonction à valeurs dans $\mathcal{B}_2$. Par exemple, « est positif » se traduira par :

   ```
   fun x -> x >= 0
   ```

   Déterminez une fonction `selec pred ens` qui renvoie l'ensemble des éléments de `ens` satisfaisant le prédicat `pred`. Vous obtiendrez par exemple :

   ```
   # selec (fun x -> x>1) e;;
   - : int ens = Ens (3, Ens (2, Vide))
   ```

9. `parties_taille_n ens n` qui renvoie les parties de `ens` de cardinal `n`.
10. `existe pred ens` qui renvoie VRAI si au moins un élément de `ens` satisfait le prédicat `pred` et FAUX sinon.

11. `pour_tout pred ens` qui renvoie VRAI si tous les éléments de ens satisfont le prédicat `pred` et FAUX sinon.

12. `partition pred ens` qui renvoie une partition de ens sous forme d'un couple de deux ensembles : la première composante est l'ensemble des éléments satisfaisant le prédicat et la deuxième est l'ensemble des autres éléments.

13. `inter ens1 ens2` qui renvoie l'intersection de ens1 et ens2.

14. `complement ens univers` qui renvoie le complémentaire de ens dans `univers`.

15. `difference ens1 ens1` qui renvoie ens1 \ ens2.

16. `est_inclus ens1 ens2` qui teste si ens1 est inclus dans ens2.

17. `est_egal ens1 ens2` qui teste cette fois l'égalité des deux ensembles. Comparez avec le = de CAML.

18. `couples el ens` qui renvoie l'ensemble des couples de première composante `el` et de deuxième composante les éléments de ens. Par exemple :

```
# couples 5 e;;
- : (int * int) ens = Ens ((5, 3), Ens ((5, 2), Ens ((5, 1), Vide)))
```

19. En déduire `produit_cart ens1 ens2` qui renvoie l'ensemble `ens1` × `ens2`.

20. ens étant un ensemble de couples, `assoc cle ens` renvoie la deuxième composante du premier couple rencontré de première composante `cle`.

Par exemple :

```
# let base = Ens(("albert","12"),Ens(("barnabe","11"),Ens(("claude","12"),Ens(("desire","13"),Ens(("
    barnabe","rugby"),Vide)))));;
val base : (string * string) ens =
  Ens
   (("albert", "12"),
    Ens
     (("barnabe", "11"),
      Ens
       (("claude", "12"),
        Ens (("desire", "13"), Ens (("barnabe", "rugby"), Vide)))))
# assoc "barnabe" base;;
- : string = "11"
```

21. `ens_assoc cle ens` qui renvoie l'ensemble des composantes associées à `cle` dans ens :

```
# ens_assoc "barnabe" base;;
- : string ens = Ens ("11", Ens ("rugby", Vide))
```

CHAPITRE 3

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 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 peut 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*.

2 Syntaxe

2 1 Les symboles

On [a] doit d'abord vérifier si les formules sont bien écrites. On a donc d'abord besoin d'un alphabet [b], 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 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 $\mathcal{F}$;

a. Le programmeur ou la machine...
b. Notion qui sera développée lors de l'étude des langages et automates

– 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 : ça vous dit quelque chose ?

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 $\leftrightarrow$.

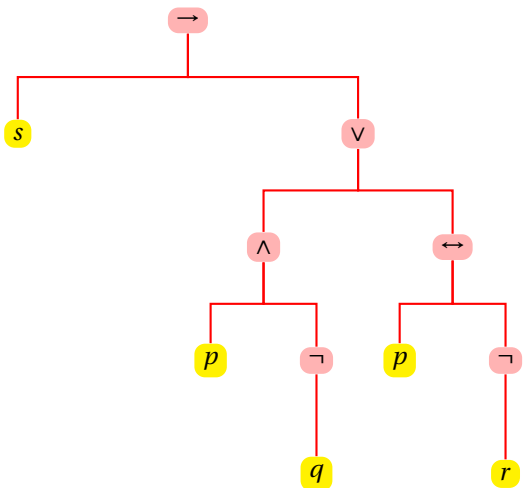
Mais attention ! $p \vee q \wedge r$ est ambigu.

Remarque

Il y a bien plus de connecteurs logiques que ceux évoqués ici. En fait, un connecteur logique est une fonction totale de $\mathcal{B}_2^n$ dans $\mathcal{B}_2$. On dit que n est son *arité*. Par exemple, $\neg$ est un connecteur d'arité 1 et $\wedge$ est un connecteur d'arité 2.

2 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 →).

2.3 Démonstration par induction

Théorème 3 - 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 3 - 2

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

La preuve est laissée en exercice.

2.4 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

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

2.5 Fonction booléenne

On considère un ensemble $\mathcal{B}_2 = \{0,1\}$ ou $\{\text{F},\text{V}\}$ ou $\{\text{Oui},\text{Non}\}$, etc.
Une fonction booléenne à n arguments est une fonction de type $\mathcal{B}_2^n \rightarrow \mathcal{B}_2$.
Par exemple :

$$f : \begin{array}{ccc} \mathcal{B}_2^3 & \rightarrow & \mathcal{B}_2 \\ (p,q,r) & \mapsto & (p \rightarrow q) \vee \neg(q \leftrightarrow r) \end{array}$$

3 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 $\mathcal{B}_2$.

3.1 Valeurs sous un environnement

Définition 3 - 1

Une *distribution de vérité* (ou *environnement* ou *interprétation propositionnelle*) est une fonction totale 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 :

- si p est une variable atomique alors $\text{Val}(p, v) = v(p)$;
- $\text{Val}(\bot, v) = 0$;
- $\text{Val}(\neg p, v) = 1$ si, et seulement si, $\text{Val}(p, v) = 0$ (NÉGATION) ;
- $\text{Val}(p \wedge q, v) = 1$ si, et seulement si, $\text{Val}(p, v) = \text{Val}(q, v) = 1$ (CONJONCTION) ;
- $\text{Val}(p \vee q, v) = 0$ si, et seulement si, $\text{Val}(p, v) = \text{Val}(q, v) = 0$ (DISJONCTION INCLUSIVE) ;
- $\text{Val}(p \rightarrow q, v) = 0$ si, et seulement si, $\text{Val}(p, v) = 1$ et $\text{Val}(q, v) = 0$ (IMPLICATION) ;
- $\text{Val}(p \leftrightarrow q, v) = 1$ si, et seulement si, $\text{Val}(p, v) = \text{Val}(q, v)$ (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 $\text{Val}(f, v)$.

À retenir

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, $\text{Val}(\bot, \langle 0, 1, 1\rangle) = 0$.

3 2 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 (pourquoi ?) 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

3 3 Tautologies, formules satisfiables, insatisfiables

Un peu de vocabulaire....

Définition 3 - 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 3 - 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 3 - 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 3 - 5

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

Par exemple $\neg p \wedge p$.

3 4 Conséquences et équivalences logiques

Définition 3 - 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. Votre maître de stage vous dit :
« Je vous paierez seulement si votre programme marche. Or votre programme ne marche pas donc je ne vous paierai pas. »
Ce raisonnement est-il correct ?
Notons p la variable atomique : « Le client paye » et m la variable : « le programme marche ».
Si le client paye, cela implique que le programme marche donc on a $p \rightarrow m$. De plus on sait que $\neg m$. La conséquence logique en est $\neg p$.
Le raisonnement du client peut donc être modélisé par :

$$p \rightarrow m,\ \neg m \models \neg p$$

Est-il correct ?
Et celui-ci :
« Je vous paierez seulement si votre programme marche. Or je ne vous paierai pas donc votre programme ne marche pas »

À retenir

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, « $F_1, F_2, ...F_p \models G$ » signifie « $(F_1 \wedge F_2 \wedge \cdots \wedge F_p) \rightarrow G$ est une tautologie ».

Recherche

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					

Des remarques ?

3 5 Formules d'équivalence de la logique des propositions

Définition 3 - 7

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

Prouvez les différentes équivalences logiques suivantes qui nous seront fort utiles :

Équivalence entre connecteurs	$p \rightarrow q \equiv \neg p \vee q$
	$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p) \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
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	$p \vee (p \wedge q) \equiv p$
	$p \wedge (p \vee q) \equiv p$

3 6 Principe de déduction par réfutation

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

Théorème 3 - 3

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.

On dit alors qu'on a montré la validité par réfutation de la conclusion.

3 7 Système complet de connecteurs

Définition 3 - 8

On appelle *système complet de connecteurs* tout ensemble $\mathbb{C}$ de connecteurs tel que toute formule est équivalente logiquement à une formule écrite avec les seuls connecteurs de $\mathbb{C}$.
Ce système est *minimal* si aucun sous-ensemble strict de $\mathbb{C}$ n'est un système complet de connecteurs.

On montre le plus souvent qu'un ensemble est un S.C.C. par induction.
Par exemple, nous montrerons en exercice que $\{\neg, \wedge\}$ et même $\uparrow$ sont des S.C.C.

3.8 Formes normales

Définition 3 - 9

On appelle *littéral* toute formule atomique ou sa négation.

Par exemple p, $\neg r$.

Définition 3 - 10

Une formule est dite sous *forme normale conjonctive (fnc)* si, et seulement si, elle est composée d'une conjonction de disjonctions de littéraux.
Une formule est dite sous *forme normale disjonctive (fnd)* si, et seulement si, elle est composée d'une disjonction de conjonctions de littéraux.

Théorème 3 - 4

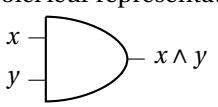
Toute formule de la LP admet une fnc minimale et une fnd minimale uniques, à l'ordre près des littéraux, qui lui sont logiquement équivalentes.

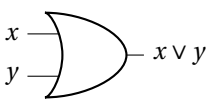
Afin d'obtenir ces formes normales, qui seront importantes pour un traitement automatique par une machine, on suit l'algorithme suivant :

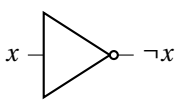
1. on passe au SCC $\{\neg, \wedge, \vee\}$ en utilisant les équivalences entre connecteurs ;
2. on réduit les négations pour n'avoir plus que des littéraux à l'aide des lois de DE MORGAN et de la double négation ;
3. distributivité, absorption et commutativité permettent enfin de conclure selon la forme voulue :
   - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ pour la fnc ;
   - $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ pour la fnd.

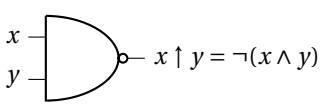
4 Portes logiques

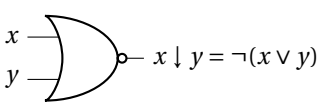
L'informatique, concrètement, c'est au départ des circuits électroniques : ça passe ou ça ne passe pas. Les calculs dans une algèbre de Boole sont donc tout à fait adaptés. On trouve dans le commerce des composants électroniques modélisant les fonctions NOT, OR, AND et également NOR, NAND, XOR que nous avons étudiées en exercice.
Voici leur représentation à la mode américaine :

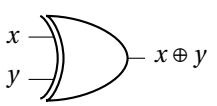




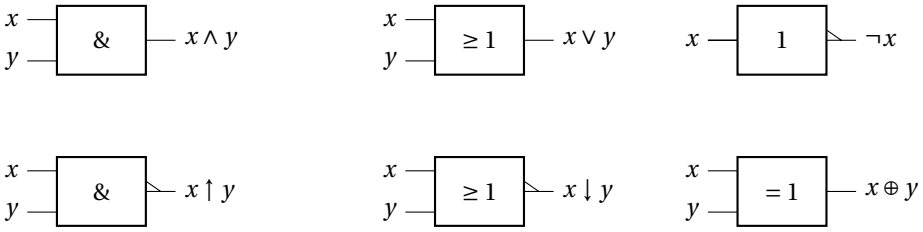




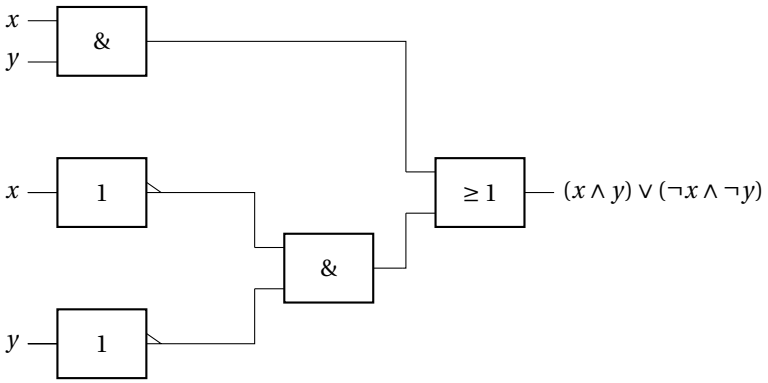




et à la mode européenne :



L'Europe, c'est mieux, alors on adoptera la deuxième série de représentations.
Construisons par exemple un circuit correspondant à $f(x, y) = (x \wedge y) \vee (\neg x \wedge \neg y)$:



Construisez les circuits correspondant à $(x \wedge y \wedge z) \vee (x \wedge \neg y \wedge z)$ et $x \wedge z$. Des remarques ?
Pouvoir placer un maximum de circuit sur un même chip est une tâche importante. Pour cela, il faut que les circuits soient les plus petits possibles tout en effectuant les tâches requises.
C'est un problème qui peut devenir compliqué et même actuellement, on a du mal à mettre au point des algorithmes traitant des expressions booléennes dépassant 25 littéraux.

Recherche

Si vous héritez d'un semi-remorque de circuits NAND, vous allez pouvoir construire des ordinateurs : pourquoi ?

5 Approche formelle de la logique propositionnelle

5.1 Principe général

Vous avez créé sur CAML une fonction de type `int -> bool`. Vous l'appliquez ensuite à 3, qui est de type `int` : le résultat sera forcément de type `bool` : il s'agit d'un exemple d'application de la règle du *Modus Ponens*, très importante en informatique. C'est la machine qui a trouvé ça toute seule, on regardant juste la *syntaxe* et non pas en étudiant la *sémantique* comme aurait tendance à le faire un humain.
Pour pouvoir effectuer une déduction logique uniquement à l'aide d'une analyse syntaxique, 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 :

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

Remarque

Notez bien la différence avec $C \rightarrow N,\ N \models B$! Pour l'obtenir, on utilise des tables de vérité, on étudie donc la sémantique des propositions. Dans le cas de $C \rightarrow N,\ N \vdash B$, 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 LP mais ce n'est pas le cas de toutes les logiques, mais ceci est une autre histoire...
Ainsi, dans le cadre de notre cours, 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	$A, \neg A \vdash B$

Aparté

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*.

Recherche

Considérons par exemple le petit jeu suivant : vous êtes Frodon et vous vous rendez à Mordor. Vous arrivez à un embranchement. Deux envoyés de Saroumane vous donnent chacun un renseignement. Le premier dit « Une au moins des deux routes mène à Mordor ». Le deuxième dit « La route de droite ne mène pas à Mordor mais au repaire de Balrog ». Saroumane croit se jouer de vous en vous précisant que soit les deux créatures disent la vérité, soit elles mentent toutes les deux. Il ne sait pas que vous avez suivi des cours de logique...Quel route allez-vous prendre ?

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 3 - 5

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

L'algorithme est le suivant :

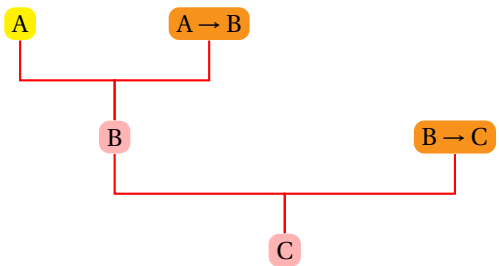
1. On fait l'*hypothèse* de A : on le rajoute temporairement aux prémisses ;
2. on démontre B en utilisant A et le reste des prémisses ;

3. on fait abstraction de A : A n'est plus forcément valide mais on obtient la conclusion $A \rightarrow B$.

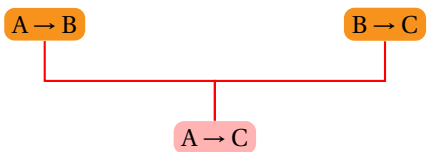
Démontrons par exemple le syllogisme hypothétique en utilisant uniquement le *modus ponens* (MP).
Les prémisses au départ sont $A \rightarrow B$ et $B \rightarrow C$:

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 claire. 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 $A \rightarrow B$:

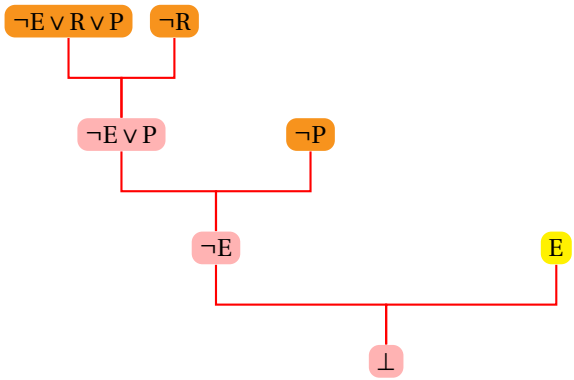


donne le nouveau théorème :



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 rigolera de mes blagues », P :« il a le petit doigt écarté ».
Les prémisses sont donc $E \rightarrow (R \vee P)$, $\neg R$, 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 normale conjonctive.
Or $A \rightarrow B \equiv \neg A \vee B$ donc $E \rightarrow (R \vee P) \equiv \neg E \vee (R \vee P) \equiv \neg E \vee R \vee P$. 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.

Aparté

Un raisonnement effectué à l'aide d'un arbre : c'est magique non ? En tout cas, ça rend la chose programmable, ce qui ne saurait manquer de vous interpeller....

5 4 Clauses de Horn - PROLOG

Une clause de littéraux n'ayant qu'un seul littéral positif est une clause de HORN [c] : c'est un cas particulier de fnc qui a un intérêt certain du fait de l'équivalence :

$$\neg A \vee B \equiv A \rightarrow B$$

En effet, une clause de HORN s'écrit :

$$(\neg P_1 \vee \neg P_2 \vee \cdots \neg P_n) \vee C$$

formule équivalente à :

$$(P_1 \wedge P_2 \wedge .... \wedge P_n) \rightarrow C$$

Le langage PROLOG est basé sur l'utilisation de clauses de HORN. Par exemple, la règle $(P_1 \wedge P_2 \wedge .... \wedge P_n) \rightarrow C$ s'écrit :

```
C :- P1,P2, ... ,Pn.
```

Un fait est juste le fait suivi d'un point : `P1,P2.`. Une question sur un but : `?- C.`.
Par exemple :

```
parallelogramme :- cote_para,cote_meme_long.
rectangle :- parallelogramme,un_angle_droit.
losange :- parallelogramme,deux_cote_meme_long.
carre :- rectangle,losange.
cote_para.
un_angle_droit.
cote_meme_long.
deux_cote_meme_long.

?- carre.
true.
```

c. Alfred HORN, (1918-2001), mathématicien américain.

5 5 Chaînage arrière - PROLOG

Le principe utilisé jusqu'ici, le chaînage avant, correspondant au théorème de la déduction, est le suivant : tous les prémisses d'une implication étant connues, on ajoute la conclusion à la base des connaissances, selon le principe du *modus ponens*. On continue jusqu'à ce que :

- on ne puisse plus rien déduire ;
- on obtient le résultat recherché.

On dit que le raisonnement est *guidé par les connaissances déjà acquises*.
Il existe une autre manière de procéder, le chaînage arrière, qui constitue le principe du moteur d'inférence du langage PROLOG.
On part du but à atteindre et on recherche dans notre catalogue de connaissances les implications dont notre but est la tête et on continue jusqu'à arriver à une proposition valide.

EXERCICES

Exercice 3 - 1

On considère les propositions atomiques B, T, V, C représentant les assertions suivantes : « Hélène est belle », « Hélène aime le thé vert », « Hélène porte une robe violette », « Chri-Chri aime Hélène ». Énoncez des phrases simples traduisant les propositions suivantes :

1. **a.** $\neg B$
   b. $\neg B \wedge \neg T$
   c. $\neg(B \vee T)$
   d. $(B \wedge V) \rightarrow C$
   e. $C \rightarrow (B \leftrightarrow V)$
   f. $((B \vee T) \wedge \neg V) \rightarrow \neg C$
2. Traduisez par une proposition simple les phrases
   a. « Chri-Chri aime Hélène seulement si elle porte une robe violette ».
   b. « Chri-Chri aime Hélène si elle porte une robe violette ».
   c. « Chri-Chri aime Hélène si, et seulement si, elle porte une robe violette ».
   d. « Si Chri-Chri aime Hélène alors elle porte une robe violette ».
   e. « Si Hélène porte une robe violette alors Chri-Chri l'aime ».
   f. « Il est suffisant qu'Hélène porte une robe viloette pour que Chri-Chri l'aime ».
   g. « Il est nécessaire qu'Hélène porte une robe viloette pour que Chri-Chri l'aime ».

Exercice 3 - 2

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

1. Si la Terre est ronde alors $2+2=5$;
2. Si $2+2=5$ alors la Terre est ronde ;
3. Si $2+2=5$ alors la Terre est plate.

Exercice 3 - 3

On considère les deux propositions :

- A :« Si je porte une robe violette alors Chri-Chri m'aime » ;
- B : « Si Chri-Chri m'aime alors je ne porte pas de robe violette ».

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

Exercice 3 - 4

Prouvez, à l'aide de tables de vérité, la distributivité de la disjonction sur la conjonction, la première loi de DE MORGAN, la règle d'absorption.

Exercice 3 - 5

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

Exercice 3 - 6

L'implication est-elle associative ?

Exercice 3 - 7

Que signifie : « Il ne m'arrive jamais de n'avoir à me plaindre de rien. »

Exercice 3 - 8

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 3 - 9

- 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 3 - 10

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 :
   - **a.** $x \oplus 0$
   - **b.** $x \oplus 1$
   - **c.** $x \oplus x$
   - **d.** $x \oplus \neg x$
2. Montrer que :
   - **a.** $x \oplus y \equiv \big(x \vee y\big) \wedge \neg \big(x \wedge y\big)$
   - **b.** $x \oplus y \equiv \big(x \wedge \neg y\big) \vee \big(\neg x \wedge y\big)$
3. L'opération $\oplus$ est-elle commutative ?
4. Vrai ou faux ?
   - **a.** $x \oplus \big(y \oplus z\big) \equiv \big(x \oplus y\big) \oplus z$
   - **b.** $x \vee \big(y \oplus z\big) \equiv (x \vee y) \oplus (x \vee z)$
   - **c.** $x \oplus (y \vee z) \equiv (x \oplus y) \vee (x \oplus z)$

Exercice 3 - 11

On considère la fonction booléenne de 4 variables définie par :

$$f = (a \wedge \neg b \wedge c) \vee (\neg a \wedge c \wedge d) \vee (a \wedge b \wedge c \wedge \neg d)$$

1. L'écriture précédente est-elle une écriture disjonctive ? conjonctive ?
2. Mettre f sous forme normale disjonctive puis sous forme normale conjonctive.

Exercice 3 - 12

Donner l'écriture normale disjonctive puis conjonctive de toutes les fonctions booléennes de 2 variables. On remarquera néanmoins que la fonction nulle n'admet pas d'écriture normale disjonctive.

Exercice 3 - 13

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

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

Exercice 3 - 14 Machine à voter

Le gouvernement syldave est constitué de trois membres qui prennent toutes les décisions en votant à bulletins secrets. Une proposition est adoptée lorsqu'elle reçoit au moins deux votes sur trois. Leur conseiller technique, ancien étudiant de l'IUT de Nantes, leur propose un petit circuit qui détermine si une décision est adoptée.
On note x, y et z les variables booléennes qui correspondent à chacun des votes du triumvirat. Déterminer une fonction booléenne $f(x, y, z)$ qui réponde au problème et dessiner le circuit correspondant avec les portes NOT, AND et OR.

Exercice 3 - 15 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 3 - 16 Divisibilité booléenne

Pour représenter les nombres en base 2, on va utiliser quatre variables booléennes 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 $b_3 = 1$, $b_2 = 0$, $b_1 = 1$ et $b_0 = 1$. Déterminez l'expression d'une fonction booléenne 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 3 - 17 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 LUKASIEWICZ (philosophe polonais (1878 - 1956))...). $\{\wedge, \vee\}$ est-il un scc ?

Exercice 3 - 18 arbre

Donnez les arbres de construction des formules suivantes :

1. $\neg(\neg(x \rightarrow y)) \vee \neg x \rightarrow \neg y$

2. $(((\neg x \rightarrow y \wedge z) \leftrightarrow t) \rightarrow \neg x \wedge z) \leftrightarrow \neg - y \wedge t$

Exercice 3 - 19 arbres

Dressez les différents arbres correspondant aux formules obtenues à partir de $a \rightarrow b \rightarrow c \rightarrow d$ en appliquant des parenthésages distincts.

Exercice 3 - 20 Étude syntaxique et sémantique

Remplacez les ? par $\vdash$ et $\models$ après vérification (préciser le type de la vérification) :

1. $p \rightarrow q \,?\, q \rightarrow p$

2. $(p \vee r) \rightarrow q \,?\, p \vee q$

3. $p \rightarrow \neg q, \neg r \rightarrow p \,?\, q \rightarrow r$

Exercice 3 - 21 théorème de déduction

En employant le théorème de déduction, dites si les raisonnements suivants sont corrects (pensez à employer des arbres) :

1.
$$\frac{a \rightarrow \neg b \vee c \qquad b \rightarrow a \wedge \neg c}{c \rightarrow b}$$

2.
$$\frac{\neg a \rightarrow b \qquad c \rightarrow \neg a}{\neg a \rightarrow (\neg c \rightarrow b)}$$

3.
$$\frac{p \rightarrow r \wedge t \qquad t \vee s \rightarrow \neg q}{\neg(p \wedge q)}$$

4.
$$\frac{(s \rightarrow r) \wedge p \qquad q}{\neg r \wedge \neg s \wedge p}$$

5.
$$\frac{p \rightarrow q \wedge r \qquad r \oplus s}{p \rightarrow s \vee t}$$

6.
$$\frac{x \vee y \rightarrow z \qquad z \leftrightarrow t}{x \vee t \rightarrow \neg y}$$

Exercice 3 - 22 Chaînages avant-arrière

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 ? Effectuer un raisonnement sur des clauses de HORN par chaînage avant puis par chaînage arrière.

Exercice 3 - 23 Police Squad

C'est bientôt la fin de l'épisode. 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.* »
Habitué aux frasques de l'inspecteur DREBIN, vous tentez de réfuter ce raisonnement avec un arbre...

Exercice 3 - 24 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.

Exercice 3 - 25 Sujet CCP 2008

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.
4. Les interrupteurs A et B laissent passer un courant (valeur logique vraie) quand ils sont fermés. Proposer un circuit électronique composé de portes logiques comportant deux sorties O et D. La sortie O laisse passer le courant pour ouvrir la porte. La sortie D laisse passer le courant pour détruire la salle.

Exercice 3 - 26 École de l'Air 2004

On souhaite concevoir un circuit logique qui renvoie le nombre modulo 2 de 0 présents dans un mot passé en entrée au circuit constitué uniquement de 0 et de 1.
On pourra utiliser les portes logiques élémentaires *et, ou, non* et *xor* (ou exclusif).

1. Commençons par un mot de longueur 1 : $u = a$, avec $a \in \{0,1\}$.
   Construire un circuit $\mathcal{P}_0$ à une seule entrée a et une sortie s qui vaut 0 si u contient un nombre pair de 0 et 1 sinon.
2. Supposons conçu un circuit $\mathcal{P}_n$ qui prend en entrée un mot u de longueur 2^n et possède une sortie s qui vaut 0 si u contient un nombre pair de 0 et 1 sinon.
   Construire, à l'aide de $\mathcal{P}_n$, un circuit $\mathcal{P}_{n+1}$ qui résout le même problème pour un mot de longueur 2^{n+1} en entrée.

Exercice 3 - 27 Politique Bordure

Grand meeting du PUB, Parti Unique Bordure, afin de choisir l'unique candidat des futurs élections dictatoriales.
Deux candidats sont en lice : Josef Chrtzw et Nikalaï Schrpuntz. Un premier orateur intervient :
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 ».
La réponse du deuxième candidat :
Nikalaï Schrpuntz : « Je n'ai qu'une seule ambition : l'unité de notre parti unique. C'est pourquoi, mes chers camarades, me soutenir à la direction du PUB, c'est soutenir l'action de notre modèle à tous, Josef Chrtzw ».
Finalement, les militants du PUB ont élu Nikalaï Schrpuntz avec 99% des voix.
Traduisez en logique des propositions les déclarations des candidats et la conclusion des militants. Vous utiliserez les propositions atomiques suivantes :

- S : soutenir Nikalaï Schrpuntz ;

– C : soutenir Josef Chrtzw ;
– V : Villenstein est un menteur ;
– SM : Nikalaï Schrpuntz est un menteur.

Le raisonnement des militants est-il valide ?

Un peu de CAML...

Les types

On crée un module pour les formules avec un type pour les atomes à choisir au départ. On prendra ici des caractères à partir de 'p' comme nous en avons l'habitude.

```
module Logic =
  struct
    type atom = char
    type formule =
    |Faux
    |Atomic of atom
    |Non of formule
    |Et of (formule * formule)
    |Ou of (formule * formule)
    |Implique of (formule * formule)
    |Equiv of (formule * formule);;
    let rang atom = Char.code(atom) - Char.code('p')
    let to_string atom = Char.escaped atom
  end;;

(* On travaille avec des atomes nommés DANS L'ORDRE p q r s t u v w x y z  : peut être personnaliser
   La seule contrainte est de conserver l'ordre

   Voici un e autre exemple avec des entiers :

   module Logic =
  struct
    type atom = int
    type formule =
    |Faux
    |Atomic of atom
    |Non of formule
    |Et of (formule * formule)
    |Ou of (formule * formule)
    |Implique of (formule * formule)
    |Equiv of (formule * formule);;
    let rang atom = atom
    let to_string atom = string_of_int atom
  end;;

*)
```

Ensuite on charge ce module :

```
open Logic;;
```

Par exemple, quelle formule représente :

```
Equiv(Non(Et(Atomic 'p',Atomic 'q')) , Ou(Non(Atomic 'p'),Non(Atomic 'q')))
```

Donnez l'arbre syntaxique associé.
C'est peu lisible malgré tout... Je vous donne dans ma grande bonté un début de petite fonction récursive qui va faciliter la lecture :

```
let rec dit_formule formule = match formule with
  |Faux -> "Faux"
  |Non Faux -> "Vrai"
  |Atomic a -> to_string a
  |Non p -> "non " ^ dit_formule p
  |Et(p,q) -> "( " ^ dit_formule p ^ " et " ^ dit_formule q ^ " )"
...
```

Par exemple :

```
# dit_formule ( Implique(Et(Atomic 's',Non Faux),Non (Ou(Faux,Atomic 't'))) );;
- : string = "( ( s et Vrai ) implique non ( Faux ou t ) )"
```

Approche sémantique

Un environnement sera modélisé sur CAML par une liste de booléens.
Pour calculer la valeur booléenne d'une formule sous un environnement donné `f`, on va procéder ainsi :

- la constante `Faux` sera associée à `false` ;
- un atome a sera associé à sa valeur dans l'environnement, c'est-à-dire renverra l'élément en position a de la liste `f` (d'où l'intérêt de travailler avec des entiers mais on aurait pu contourner le problème...) ;
- si p est une formule, la valeur de sa négation est la négation booléenne de sa valeur ;
- etc.

Il faudra donc disposer tout d'abord d'une fonction `nieme k liste` qui renvoie le k^e élément de la liste `liste`.

```
exception Mauvais_numero;;

let rec nieme k liste =
  match liste with
    |[] -> raise Mauvais_numero
    |t::q -> if... then ...
      else ...;;
```

```
# nieme 3 [1;2;3;4;5];;
- : int = 3
```

On abusera du `if...then...else....` et on utilisera certaines équivalences :

```
let rec eval formule env =
  match formule with
    |Faux -> false
    |Atomic a -> nieme (rang a) env
    |Non p -> if (eval p env) then false else true
    |Et (p,q) ->
      if (eval p env) then
        if (eval q env) then
          ...
        else
          ...
      else
        ...
```

```
    |Ou (p,q) -> ...
    |Implique (p,q) -> ...
    |Equiv (p,q) -> ... ;;
```

Ensuite, il faudra tester si une formule est une tautologie.
Pour cela, il faudrait créer la liste de tous les environnements possibles. On a besoin de quelques outils sur les listes :

- `concat l1 l2` qui concatène les deux listes `l1` et `l2` : à vous de la fabriquer ;
- `ajoute_el el liste_de_listes` qui fait ça :

```
let rec ajoute_el el liste_de_listes =
  match liste_de_listes with
    |[] -> []
    |t::q -> (el::t)::(ajoute_el el q);;
```

Ça fait quoi au juste ?

Ensuite on cherche tous les environnements de longueur n. Par exemple :

```
# tous_environnements 3;;
- : bool list list =
[[true; true; true]; [true; true; false]; [true; false; true];
 [true; false; false]; [false; true; true]; [false; true; false];
 [false; false; true]; [false; false; false]]
```

Comment faire ? Penser à la construction récursive de l'ensembles des parties d'un ensemble.

Il ne reste plus qu'à évaluer une formule sous tous ces environnements pour savoir si c'est une tautologie.

```
# let est_tautologie formule n =
...
val est_tautologie : Logic.formule -> int -> bool = <fun>
# est_tautologie ( Equiv(Non(Et(Atomic 'p',Atomic 'q')) , Ou(Non(Atomic 'p'),Non(Atomic 'q')))) 2;;
- : bool = true
```

Testez vos résultats sur l' Exercice 3 - 13 page 60.

CHAPITRE

4 Relations

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 43 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.

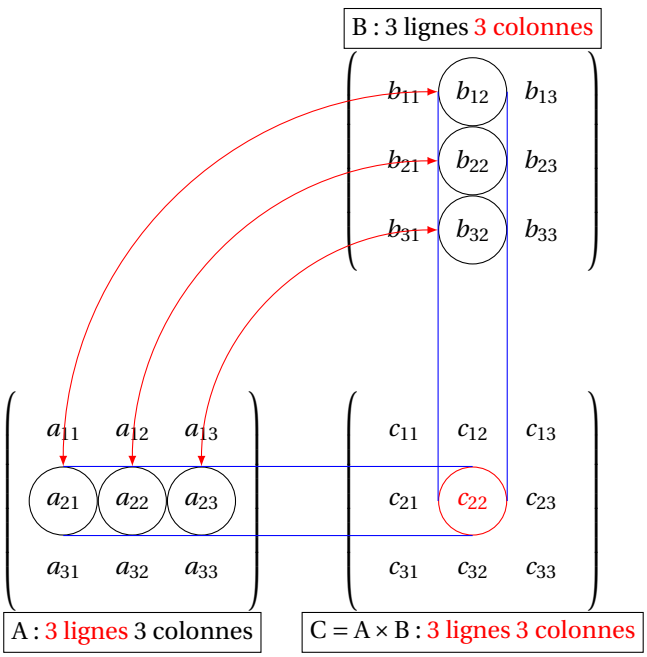
1 1 Un peu de calcul matriciel

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.

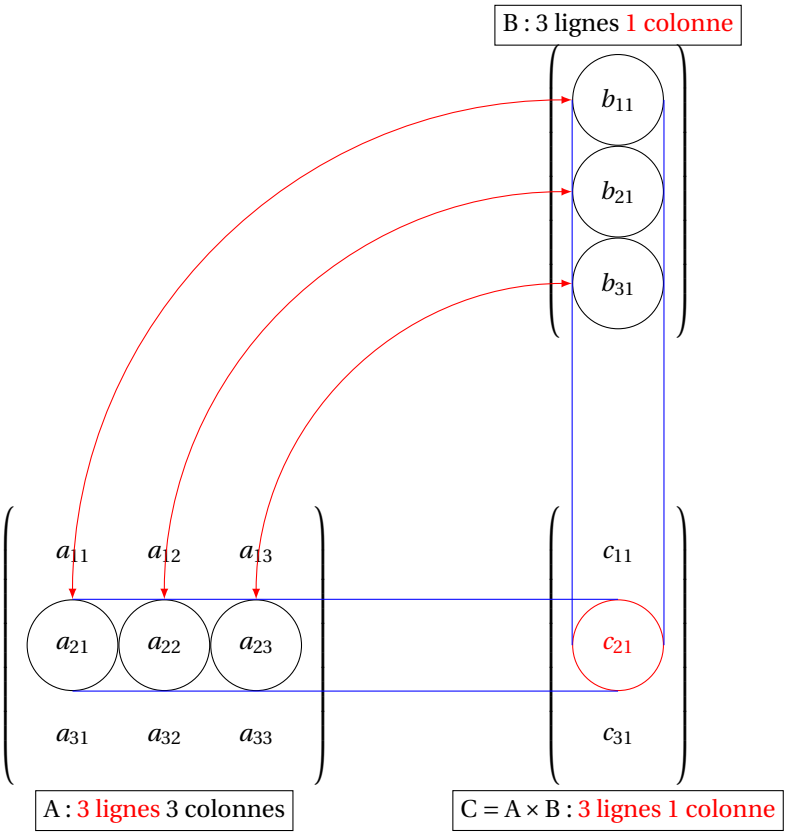
1 1 1 Produit de deux matrices « carré »



Danger

produit de matrice non commutatif
Le produit de deux matrices n'est pas commutatif. Cela signifie qu'en général A × B ≠ B × 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

$\underline{\vee}$	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.

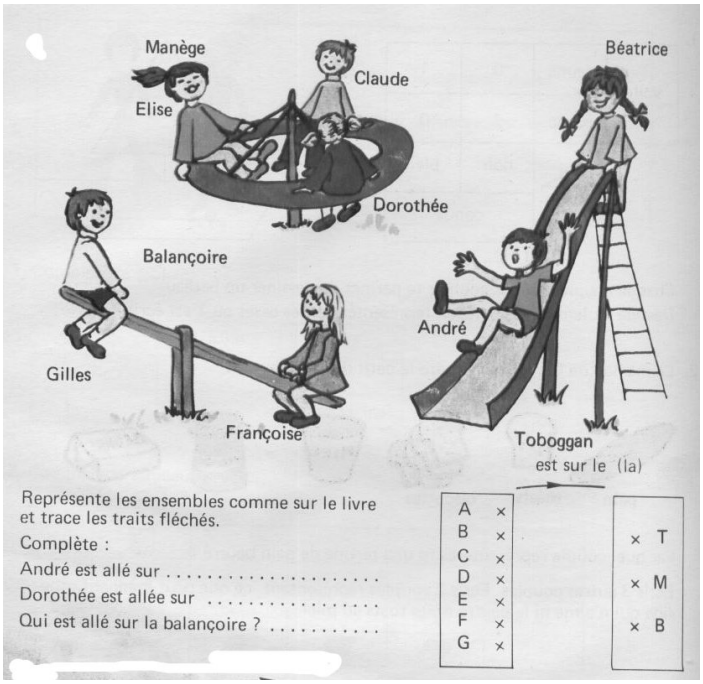
Soit par exemple

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

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).

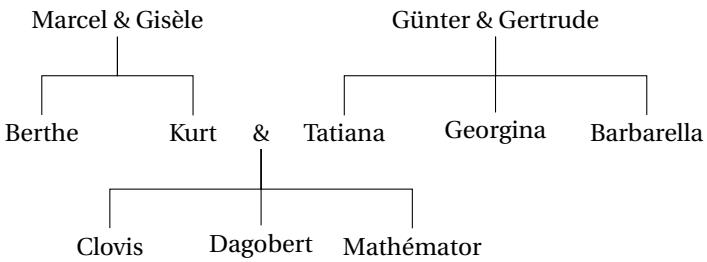
2 Relations binaires

2 1 Au CE1



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 4 - 1

Une *relation binaire* entre deux ensemble E et F est la donnée de E, F et d'un sous-ensemble du produit cartésien $E \times F$.
Une *relation n-aire* entre n ensembles E_1, E_2,...,E_n est la donnée de ces ensembles et d'un sous-ensemble du produit cartésien $E_1 \times E_2 \times \cdots E_n$.

Remarques

- Dans un premier temps, nous nous contenterons de l'étude des relations binaires ;
- les deux ensembles E et F peuvent être égaux ;
- en fait, plus rigoureusement, une relation binaire de l'ensemble E vers (ou dans) l'ensemble F est un triplet $(E, F, G_{\mathcal{R}})$ où $G_{\mathcal{R}}$ est une partie de $E \times F$ appelé le *graphe* de la relation. Par abus et parce que ça n'entraîne pas de confusions, on confond une relation avec son graphe.

Il faut bien comprendre que pour définir parfaitement la relation $\mathcal{R}$ il nous faut connaître trois éléments E, F et $G_{\mathcal{R}} \subseteq E \times F$; si au moins l'un de ces trois éléments manque, la relation $\mathcal{R}$ n'est pas définie. Si $G_{\mathcal{R}} = \{\}$, $\mathcal{R}$ est la relation vide de E vers F, ce sera forcément le cas si E ou F est vide. L'ensemble des relations binaires de E vers (ou dans) F est noté $E \longleftrightarrow F$ ou $E \leftrightarrow F$ et est confondu, si besoin, avec $\mathcal{P}(E \times F)$, l'ensemble des parties de $E \times F$.
Les écritures $\mathcal{R} \in E \longleftrightarrow F$ ou $\mathcal{R} : E \longleftrightarrow F$ ou $\mathcal{R} \in \mathcal{P}(E \times F)$ doivent se traduire par « $\mathcal{R}$ est une relation binaire de E vers F ». Si E = F on dit que $\mathcal{R}$ est une relation binaire de E vers E ou de E dans E ou encore une relation binaire **sur** E.
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 $(x, y) \in G_{\mathcal{R}}$ ou $(x, y) \in \mathcal{R}$ puisque l'on se permet de confondre $\mathcal{R}$ et $G_{\mathcal{R}}$:

$$x \mathcal{R} y \Leftrightarrow \mathcal{R}(x, y) \Leftrightarrow (x, y) \in G_{\mathcal{R}} \Leftrightarrow (x, y) \in \mathcal{R}$$

et 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

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 CE1 page précédente.
La relation $\mathcal{R}$ est définie par « est sur le (la) » avec un ensemble de départ qui est en fait l'ensemble des élèves : $E = \{a, b, c, d, e, f, g\}$ et un ensemble d'arrivée qui est en fait l'ensemble des jeux : $J = \{T, M, B\}$
Son graphe est :

$$G_{\mathcal{R}} = \{(a, T), (b, T), (c, M), (d, M), (e, M), (f, B), (g, B)\}$$

Les écritures suivantes sont équivalentes :

$$(a, T) \in G_{\mathcal{R}} \Longleftrightarrow a \mathcal{R} T \Longleftrightarrow (a, T) \in \mathcal{R}$$

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 $2^{848} \approx 2.10^{255}$ et le nombre d'atomes estimé dans tout l'univers est 10^{80} : il y a donc pas mal de relations possibles entre les 106 garçons et les 8 filles d'INFO 1...

2 5 Domaine, codomaine

Définition 4 - 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 $\text{dom}(\mathcal{R})$. Remarquons que l'on a forcément $\text{dom}(\mathcal{R}) \subseteq \text{E}$.

$$\text{dom}(\mathcal{R}) = \{x \in \text{E} \mid \text{il existe au moins } y \in \text{F} \text{ avec } x\mathcal{R}y\}$$

Si on est sûr que $\text{dom}(\mathcal{R}) = \text{E}$, 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**

Définition 4 - 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 $\mathcal{R}$

$$\text{Im}\,\mathcal{R} \text{ ou } \text{Im}(\mathcal{R}) \text{ ou } \text{Ran}(\mathcal{R}) \text{ ou } \text{codom}(\mathcal{R})$$

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

$$\text{Im}(\mathcal{R}) = \{y \in \text{F} \mid \text{il exite au moins } x \in \text{E} \text{ avec } x\mathcal{R}y\}$$

et que l'on a forcément $\text{Im}(\mathcal{R}) \subseteq \text{F}$.

2 6 Représentation d'une relation

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

2 6 1 Représentation linéaire ensembliste

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

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

$\text{G}_{\mathcal{R}}$ est aussi noté $\text{G}_{\mathcal{R}} = \{a \mapsto \gamma, b \mapsto \beta, b \mapsto \delta, d \mapsto \gamma, e \mapsto \delta\}$.
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, $x\mathcal{R}y$.

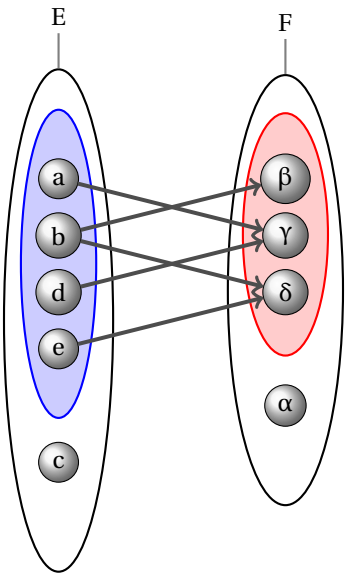


FIGURE 4.1 – Relation non totale

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 [a] $R = (r_{i,j}) \in \mathfrak{M}_{n,p}$ 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 :

↷	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 6 6 Un exemple de représentation sur Caml

On crée une relation par listes d'images : une relation binaire est une liste de paires

`sommets,[liste des images]`

implémentée à l'aide de notre type ens précédemment étudié. Les ensembles d'arrivée et de départ peuvent être distincts et les variables de types `'a` et `'b` distincts ou non :

```
type ('a,'b) relation =
  Rel of ('a * 'b ens)  ens ;;
```

a. Ceux qui ne savent pas encore ce qu'est une matrice (ce doit être le cas de la majorité d'entre vous) le découvriront avec le module Algèbre et Analyse 1, il suffit de savoir ici qu'une matrice est tout simplement un tableau de « nombres ».

Par exemple :

```
let r = Rel(
  Ens((1, Ens ('a', Ens ('b', Vide))),
     Ens((2, Ens ('c', Vide)),
        Ens((3, Ens ('b', Vide)),
           Ens((4, Vide),
              Ens((5, Ens ('b', Vide)),
                 Vide)))))
);;
```

Voyons son type :

```
# r;;

- : (int, char) relation
```

2 7 Relation transposée

Définition 4 - 4

La **relation transposée** de la relation $\mathcal{R} = (E, F, G_{\mathcal{R}})$ est la relation notée $\mathcal{R}^{\sim}$ définie par $\mathcal{R}^{\sim} = (F, E, G_{\mathcal{R}^{\sim}})$ avec $G_{\mathcal{R}^{\sim}} = \{(y, x) \in F \times E \mid (x, y) \in G_{\mathcal{R}}\}$, c'est-à-dire $y\mathcal{R}^{\sim}x \Leftrightarrow x\mathcal{R}y$.

Pour obtenir la représentation sagittale de $\mathcal{R}^{\sim}$, 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 ${}^{t}R$, la transposée de R, est la matrice de $\mathcal{R}^{\sim}$, 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 :

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

Les résultats suivants découlent immédiatement de la définition :

- $\text{dom}(\mathcal{R}) = \text{Im}(\mathcal{R}^{\sim})$
- $\text{dom}(\mathcal{R}^{\sim}) = \text{Im}(\mathcal{R})$
- $(\mathcal{R}^{\sim})^{\sim} = \mathcal{R}$

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 4 - 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 4 - 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 $\mathcal{R}^{\sim}$. Elle est donc notée $\mathcal{R}^{\sim}(V)$.

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

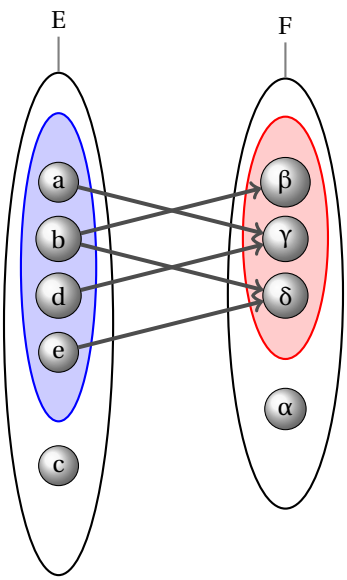


FIGURE 4.2 – Image et contre-image

Nous avons $\mathcal{R}(\{a,b\}) = \{\beta,\gamma\}, \mathcal{R}(\{c\}) = \{\gamma\}, \mathcal{R}^{\sim}(\{\alpha\}) = \{\}, \mathcal{R}^{\sim}(\{\gamma\}) = \{a,c,d\}$ 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 $TV^{\sim}(\{\text{Gulli}\})$.

2 9 Égalité de deux relations

Définition 4 - 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 × 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 4 - 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 $\overline{\text{TV}}$ 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 4 - 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 :

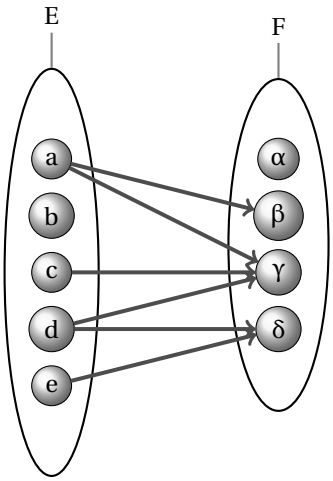


FIGURE 4.3 – $\mathcal{R}_1$

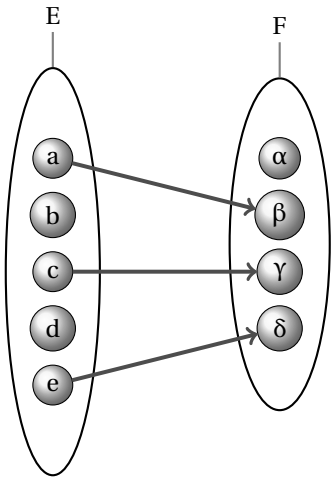


FIGURE 4.4 – $\mathcal{R}_2$ est une sous-relation de $\mathcal{R}_1$

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 ↦ Gulli, Berthe ↦ Gulli, Berthe ↦ Zen TV, Jean-Pierre ↦ Zen TV, Gudrun ↦ Gulli, Gudrun ↦ TF1, Gudrun ↦ Zen TV}.
Alors TV ⊆ 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 $(1 \veebar 1 = 1)$:

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

Notations

Méthode B
Voici les notations utilisées en méthode B, $\mathcal{R}$ désignant une relation de E vers F.
Obligatoirement $A \subseteq E$ **et** $B \subseteq F$.

- *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 4 - 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 $\mathcal{R}_2$:

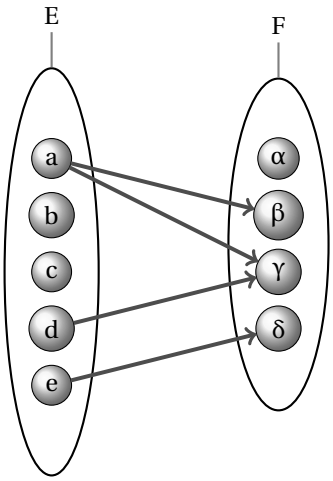


FIGURE 4.5 – $\mathcal{R}_1$

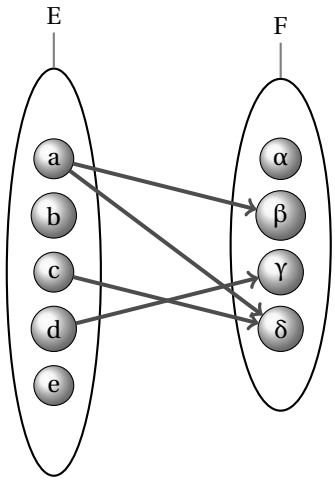


FIGURE 4.6 – $\mathcal{R}_2$

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

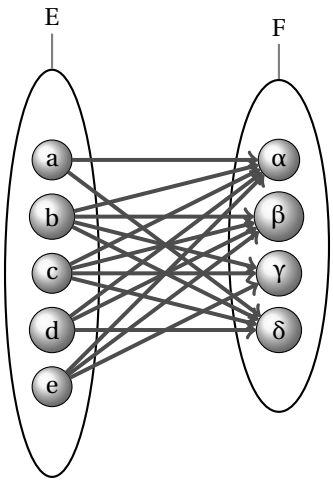


FIGURE 4.7 – $\overline{\mathcal{R}_1}$

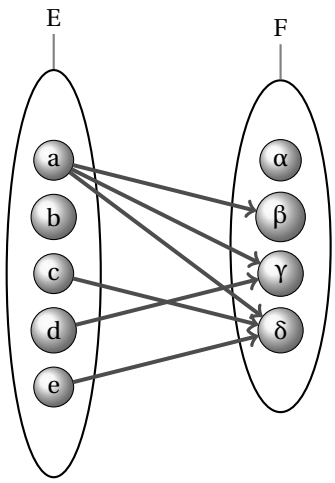


FIGURE 4.8 – $\mathcal{R}_1 \cup \mathcal{R}_2$

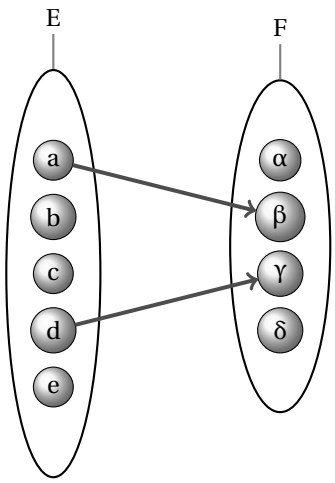


FIGURE 4.9 – $\mathcal{R}_1 \cap \mathcal{R}_2$

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

2 11 Composition de relations

Définition 4 - 11

$\mathcal{R} = (E, F, G_{\mathcal{R}})$ et $\mathcal{S} = (U, V, G_{\mathcal{S}})$ sont deux relations.
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 $\mathcal{S}$) et pour graphe

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

Pour simplifier les choses, $\mathcal{S} \circ \mathcal{R}$ est aussi notée $\mathcal{R}.\mathcal{S}$ ou bien encore $\mathcal{R};\mathcal{S}$ (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 :

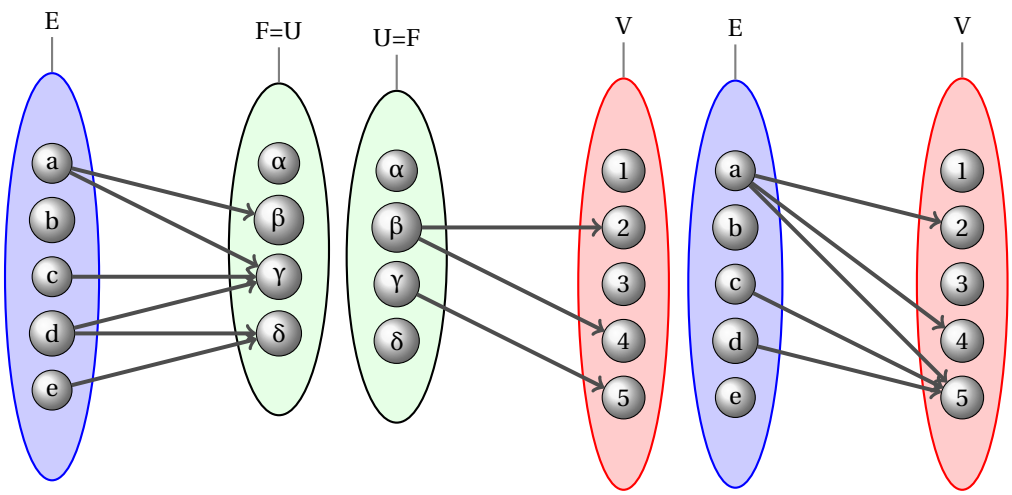


FIGURE 4.10 – $\mathcal{R}$ FIGURE 4.11 – $\mathcal{S}$ FIGURE 4.12 – $\mathcal{S} \circ \mathcal{R}$

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 $\mathcal{R}^{\sim} \circ \mathcal{R}$ ou de $\mathcal{R} \circ \mathcal{R}^{\sim}$? 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↦Josette, alouette↦Pépette, albatros↦Bernard, bécasse↦Germaine$\}$;
- Cri=$\{$Josette↦turlute, Josette↦grisole, Pépette↦turlute, Bernard↦piaule, Germaine↦croul

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

3 Fonctions

3 1 Définitions

Définition 4 - 12

On dit que **la relation** $f = (E, F, G_f)$ est une **fonction** (ou une dépendance fonctionnelle) 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

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

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

Définition 4 - 13

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

Recherche

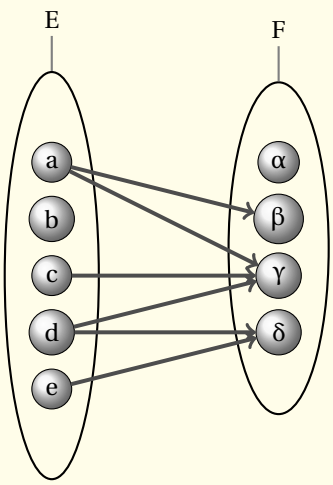


FIGURE 4.13 – f

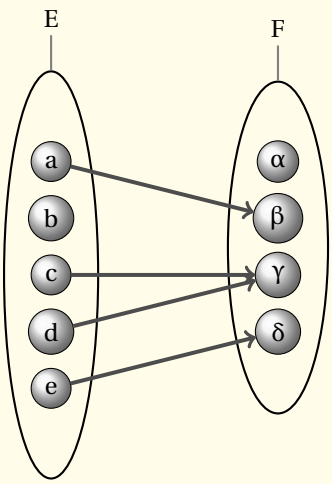


FIGURE 4.14 – g

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 4 - 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 4 - 15

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

$$Id_E : \begin{array}{ccc} E & \to & E \\ x & \mapsto & x \end{array}$$

Définition 4 - 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 4 - 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 4 - 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 Caml, on peut définir de nouveaux opérateurs infixes :

```
# let biplus(a,b) = a + b + b ;;
val biplus : int * int -> int = <fun>
```

```
# let (++) a b = biplus(a,b);;
val ( ++ ) : int -> int -> int = <fun>

# 1 ++ 2 ;;
- : int = 5

# biplus(1,2);;
- : int = 5
```

3 3 Fonction injective

Définition 4 - 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.

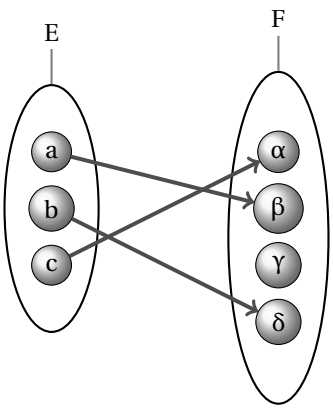


FIGURE 4.15 – Injection

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 :

$$f \text{ injective} \quad \Leftrightarrow \quad \forall (x, x') \in E^2, \left(x \neq x' \Rightarrow f(x) \neq f(x')\right)$$
$$f \text{ injective} \quad \Leftrightarrow \quad \forall (x, x') \in E^2, \left(f(x) = f(x') \Rightarrow x = x'\right)$$

3 4 Fonction surjective

Définition 4 - 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.

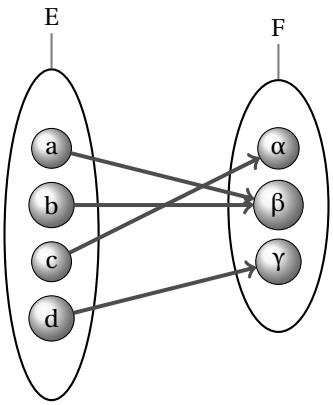


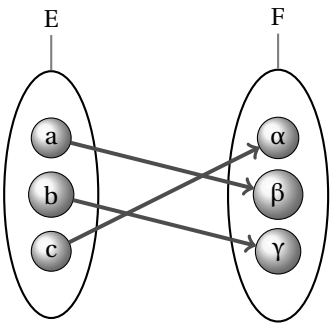
FIGURE 4.16 – Surjection

Cela équivaut à écrire que **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 4 - 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 $\nrightarrow$ 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 $\longrightarrow\!\!\!\!\!\rightarrowtail$ C : injections totales de D vers C ;
- D $\twoheadrightarrow$ C : surjections partielles de D vers C ;
- D $\rightarrow\!\!\!\!\!\twoheadrightarrow$ C : surjections totales de D vers C ;
- D $\rightarrowtail\!\!\!\!\!\twoheadrightarrow$ C : bijections partielles de D sur C ;
- D $\longrightarrow\!\!\!\!\!\rightarrowtail\!\!\!\!\!\twoheadrightarrow$ C : bijections totales de D sur C.

4 Relations binaires sur un ensemble

4 1 Au CE1

4 2 Définition

Définition 4 - 22

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

$$\mathcal{R} = (\mathrm{E}, \mathrm{E}, \mathrm{G}_{\mathcal{R}})$$

où $\mathrm{G}_{\mathcal{R}}$ est une partie de $\mathrm{E} \times \mathrm{E}$.

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* :

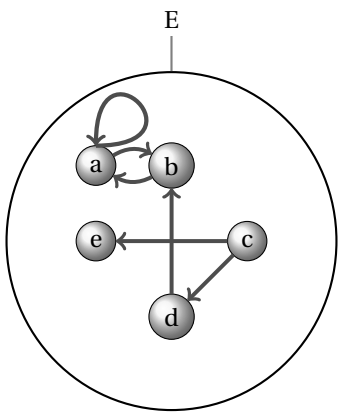


FIGURE 4.17 – Diagramme sagittal d'une relation sur un ensemble (version 2)

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 4 - 1

- La composition ou le produit de relations binaires sur l'ensemble E est une loi de composition interne dans l'ensemble des relations binaires sur E.
- Ce produit est associatif et il a pour élément neutre Id_E :

$$\mathcal{R} \ ; Id_E = Id_E \ ; \mathcal{R} = \mathcal{R}$$

- 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 :

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

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 chemine 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 4 - 23

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

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

Théorème 4 - 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 Relations particulières

Définition 4 - 24

Relation réflexive
$\mathcal{R}$ est **réflexive** si, et seulement si, pour tout x de E on a $x\mathcal{R}x$. En d'autres termes $\mathcal{R}$ est réflexive si, et seulement si, $G_{\mathcal{R}}$ contient Δ la diagonale de $E \times E$, $\Delta = \{(x,x) \mid x \in E\}$.

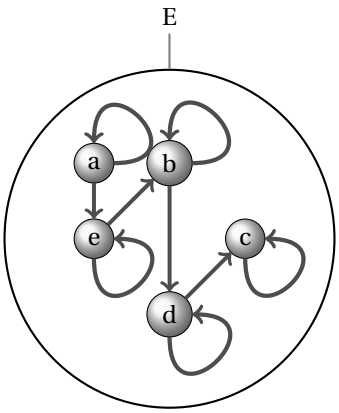


FIGURE 4.18 – Relation réflexive

Théorème 4 - 3

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

Définition 4 - 25

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

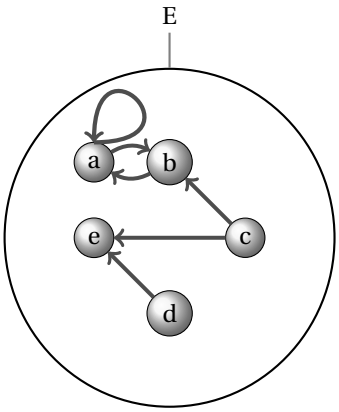


FIGURE 4.19 – La relation $\mathcal{R}$

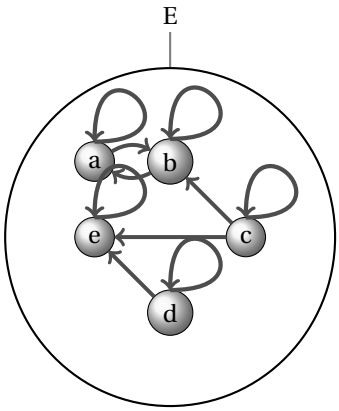


FIGURE 4.20 – La fermeture réflexive de $\mathcal{R}$

Définition 4 - 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...

Définition 4 - 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 $y\mathcal{R}x$.

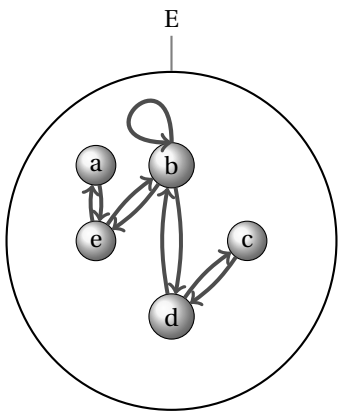


FIGURE 4.21 – Relation symétrique

Recherche

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

Théorème 4 - 4

Matriciellement, $\mathcal{R}$ est symétrique si, et seulement si, sa matrice R est symétrique, c'est-à-dire ${}^t\mathrm{R} = \mathrm{R}$.

Définition 4 - 28

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

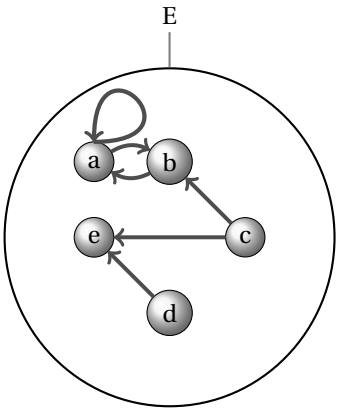


FIGURE 4.22 – La relation $\mathcal{R}$

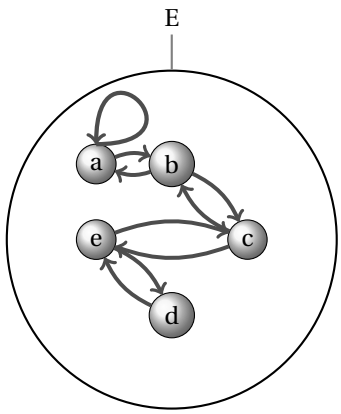


FIGURE 4.23 – La fermeture symétrique de $\mathcal{R}$

Théorème 4 - 5

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

Définition 4 - 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 4 - 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 ».

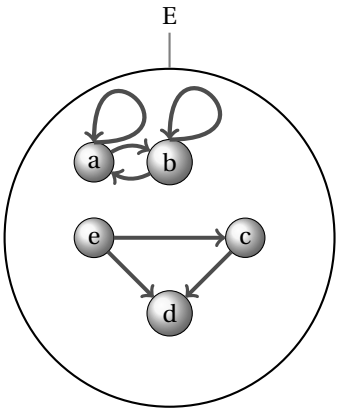


FIGURE 4.24 – Relation transitive

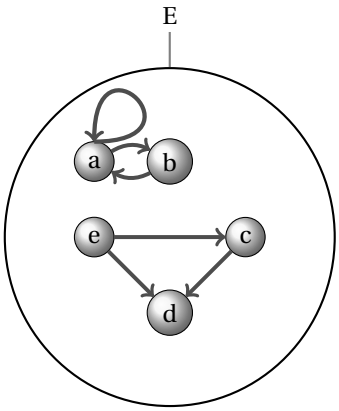


FIGURE 4.25 – Relation non transitive

Recherche

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

Théorème 4 - 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 4 - 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 $G_{\mathcal{R}}$.

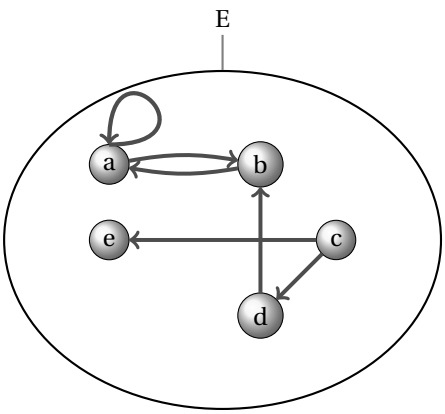


FIGURE 4.26 – Relation

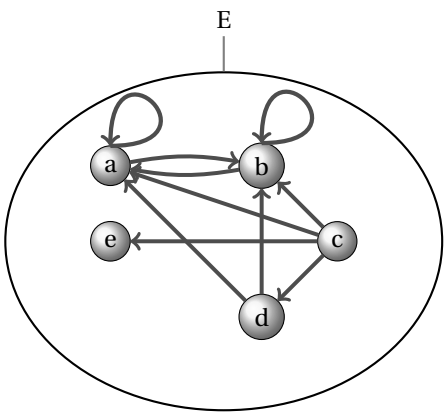


FIGURE 4.27 – Fermeture transitive de la relation

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 4 - 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 4 - 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 4 - 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...

Fermeture d'une relation : du concret

Disons que vous gérer 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 Relations d'équivalence

Tous les étudiants d'Info s'inscrivent pour un camp d'été de maths auprès de leur professeur adoré.
Comme il y a beaucoup d'étudiants, on décide d'inscrire les étudiants dont le nom de famille commence par une lettre comprise entre A et G le lundi, ceux dont le nom commence par une lettre entre H et N le mardi et les autres le mercredi.
Soit $\mathcal{R}$ la relation définie sur l'ensemble $\mathcal{E}$ des étudiants de l'IUT qui contient (x, y) si x et y se sont inscrits le même jour.
On vérifie aisément que cette relation est réflexive, symétrique et transitive.
On remarque surtout que $\mathcal{R}$ divise l'ensemble $\mathcal{E}$ des étudiants en trois classes distinctes. Il suffit de savoir dans quel classe se trouve l'étudiant pour savoir quel jour il s'inscrira, sans avoir besoin de connaître son identité : gain de temps...
Ce genre de relation réflexive, symétrique et transitive s'appelle une **relation d'équivalence**.

Définition 4 - 33

$\mathcal{R} = \left(\mathrm{E}, \mathrm{G}_{\mathcal{R}}\right)$, où E est un ensemble non vide, est une **relation d'équivalence** si, et seulement si, $\mathcal{R}$ est

Réflexive et Symétrique et Transitive (R.S.T.)

x désignant un élément de E, la **classe d'équivalence** de x est l'ensemble

$$cl(x) = \overline{x} = \left\{y \in \mathrm{E} \mid x\mathcal{R}y\right\} = \left\{y \in \mathrm{E} \mid y\mathcal{R}x\right\}$$

Beaucoup d'autres notations sont utilisées : $\overline{x} = cl(x) = \widehat{x} = \widetilde{x} = [x]_{\mathcal{R}}$ ou encore $\mathring{x} = \check{x} = \cdots$.

Propriété 4 - 1

La classe de x, $\overline{x}$, ne peut être un ensemble vide.

En effet, $\mathcal{R}$ étant réflexive, on a pour tout x de E, $x\mathcal{R}x$ et, par conséquent, $\overline{x}$ contient au moins un élément qui est x.

Propriété 4 - 2

$$x\mathcal{R}y \Leftrightarrow \overline{x} = \overline{y}.$$

1. Supposons tout d'abord que $x\mathcal{R}y$ et choisissons un élément $z \in \overline{y}$; nous avons alors $y\mathcal{R}z$ et comme $x\mathcal{R}y$ nous obtenons $x\mathcal{R}z$ qui prouve que $z \in \overline{x}$. Nous venons de démontrer que $\overline{y} \subseteq \overline{x}$. En permutant les rôles de x et y (ne pas oublier que $\mathcal{R}$ est symétrique) on obtient $\overline{x} \subseteq \overline{y}$ et, par conséquent, $\overline{x} = \overline{y}$.
2. Supposons maintenant que $\overline{x} = \overline{y}$, nous savons que $y \in \overline{y}$ et donc que $y \in \overline{x}$; on en déduit que $x\mathcal{R}y$.

Propriété 4 - 3

$\overline{x} \cap \overline{y} = \{\} \Leftrightarrow x$ n'est pas en relation avec y.

Nous venons d'établir que $x\mathcal{R}y \Leftrightarrow \overline{x} = \overline{y}$. En conséquence, si $x\mathcal{R}y$, il est impossible que $\overline{x} \cap \overline{y} = \{\}$ puisque $\overline{x} = \overline{y}$ et qu'aucune classe n'est vide. Nous venons de prouver que $\overline{x} \cap \overline{y} = \{\} \Rightarrow x\overline{\mathcal{R}}y$. transposéement, supposons que $\overline{x} \cap \overline{y} \neq \{\}$. La classe de x et la classe de y ont donc au moins un élément en commun z qui vérifie $x\mathcal{R}z$ et $z\mathcal{R}y$ et, du coup, on a forcément $x\mathcal{R}y$. Nous venons de démontrer que $\overline{x} \cap \overline{y} \neq \{\} \Rightarrow x\mathcal{R}y$ donc aussi $x\overline{\mathcal{R}}y \Rightarrow \overline{x} \cap \overline{y} = \{\}$.

Propriété 4 - 4

L'ensemble des classes d'équivalences, noté $\mathrm{E}/\mathcal{R}$, est une partition de E.

Nous n'allons détailler ce résultat que lorsque le nombre de classes est fini : notons $C_1, C_2, \ldots, C_k$ les différentes classes d'équivalence. Nous venons de prouver que $C_i \cap C_j = \{\}$ si $i \neq j$. Maintenant choisissons un élément quelconque $e \in E$, nous savons que $e \in \overline{e}$ et la classe de e est forcément l'une des classes C_i. Ceci prouve que $\bigcup_{i=1}^{k} C_i = E$

Propriété 4 - 5

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...

6 Structures d'ordre

6 1 Relation d'ordre

La relation $\mathcal{R} = (E, G)$ est une **relation d'ordre** (ordre large) si, et seulement si, $\mathcal{R}$ est

Réflexive et Antisymétrique et Transitive (R.A.T.)

et on vérifie immédiatement que

$$\mathcal{R} \text{ relation d'ordre} \Leftrightarrow \mathcal{R}^{-1} \text{ relation d'ordre}$$

Définition 4 - 34

Une relation d'ordre est dite **totale** si, et seulement si, pour tout couple (x, y) de E on a soit $x\mathcal{R}y$, soit $y\mathcal{R}x$. Cela signifie que deux éléments quelconques sont toujours « comparables ». Si la relation d'ordre n'est pas totale on dit qu'elle est **partielle** et, dans ce cas, il existe au moins deux éléments distincts de E qui ne sont pas comparables.
La relation $\mathcal{R} = (E, G)$ est une relation d'**ordre strict** si, et seulement si, $\mathcal{R}$ est

Irréflexive et Antisymétrique et Transitive

En général une relation d'ordre (large) est notée par $\leq$ ou par $\preceq$ ou par $\leqslant$ et une relation d'ordre strict est notée par $<$ ou par $\prec$.
La relation transposée de la relation d'ordre $\leqslant$ est notée $\geqslant$: $x \leqslant y \Leftrightarrow y \geqslant x$ et il est bien entendu que l'on peut utiliser à loisir, l'une ou l'autre.

Théorème 4 - 9

$$\mathcal{R} \text{ est une relation d'ordre} \Leftrightarrow \begin{cases} \mathcal{R} \cap \mathcal{R}^{-1} = \mathrm{I}d_E \\ \mathcal{R} = \mathcal{R}^2 = \mathcal{R}^* \end{cases}$$

Pourquoi ?...

6 2 Ensembles ordonnés

Définition 4 - 35

Un ensemble E est dit ordonné par la relation $\mathcal{R}$ si $\mathcal{R}$ est une relation binaire sur E et si $\mathcal{R}$ est une relation d'ordre. Un **ensemble ordonné** se présente comme le couple $(E, \mathcal{R})$ où $\mathcal{R}$ est une relation d'ordre sur E. On dit que l'ensemble ordonné $(E, \mathcal{R})$ est totalement ordonné si $\mathcal{R}$ est une relation d'ordre total.

Attention ! Un même ensemble peut être muni de plusieurs relations d'ordre différentes, si $\leqslant_1$ et $\leqslant_2$ sont deux relations d'ordre différentes alors les ensembles ordonnés $(E, \leqslant_1)$ et $(E, \leqslant_2)$ sont des ensembles ordonnés différents.

Exemple

Montrer par exemple que la relation d'inclusion $\subseteq$ est une relation d'ordre (de quelle nature ?) sur l'ensemble des parties d'un ensemble donné E.
Que pensez-vous de l'ensemble $\mathbb{Z}^+$ muni de la relation « divise » ?

À retenir

En fait, on peut montrer que le principe de récurrence s'applique sur tout ensemble bien ordonné.

6 3 Ordre lexicographique

Comment sont rangés les mots dans un dictionnaire ? C'est une des manière de ranger des éléments d'un produit cartésien (comment le définir ?). Y en a-t-il d'autres possibles ?

6 4 Diagramme de Hasse

Dresser un diagramme sagittal 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 4 - 36

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

$$\begin{cases} a < b \\ a \leqslant c \leqslant b \Rightarrow a = c \text{ ou } b = c \end{cases}$$

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 :

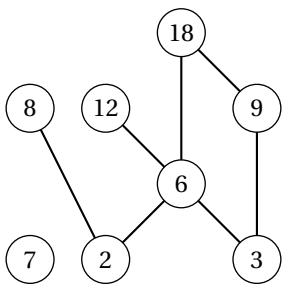


FIGURE 4.28 – Diagramme de HASSE de $\{\{2,3,6,7,8,9,12,18\},|\}$

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

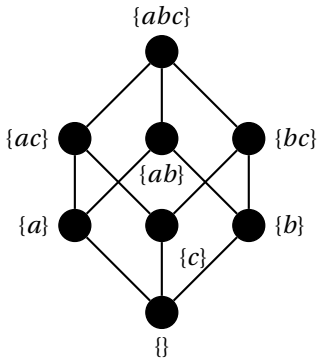


FIGURE 4.29 – Diagramme de HASSE de $\{\mathcal{P}(\{a,b,c\}),\subseteq\}$

6 5 Éléments remarquables

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

Définition 4 - 37

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

$$\text{pour tout } x \text{ de A}, x \leq u$$

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

$$\text{pour tout } x \text{ de A}, v \leq x$$

Définition 4 - 38

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

$$g \in \text{A et } \forall x \in \text{A}, x \leq g$$

Si g existe, il est unique et on ne doit pas parler d'un plus grand él ément de A mais du plus grand élément de A.

Démontrons ce résultat : pour cela supposons que g et g' sont deux plus grands éléments de A, nous avons alors

$$\begin{array}{lll} \forall x & \in & \text{A}, x \leq g \text{ donc } g' \leq g \\ \forall y & \in & \text{A}, y \leq g' \text{ donc } g \leq g' \end{array}$$

L'antisymétrie de ≤ permet alors de conclure $g = g'$.
Le plus grand élément de A (s'il existe) est noté Max(A) et c'est aussi évidemment le plus petit majorant de A.

Définition 4 - 39

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

$$p \in \text{A et } \forall x \in \text{A}, p \leq x$$

Si p existe, il est unique et on ne doit pas parler d'un plus petit élément de A mais du plus petit élément de A.

Le plus petit élément de A (s'il existe) est noté Min(A) et c'est alors le plus grand minorant de A.

Définition 4 - 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 $\text{Sup}_{\text{E}}(\text{A})$ ou Sup(A).

Définition 4 - 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 $\text{Inf}_{\text{E}}(\text{A})$ ou Inf(A).

Définition 4 - 42

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

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 $(\text{E}, \leq)$ à l'aide d'un diagramme de HASSE, on voit apparaître les éléments minimaux ou maximaux éventuels de E ainsi que les bornes ou é léments extrémaux éventuels de toute partie de E.

Observons par exemple le diagramme de la figure 4.28 page 93. 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 4.29 page ci-contre.
Le plus petit élément est l'ensemble vide car $\{\} \subseteq \text{A}$ 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 $\mathcal{E}$.

6 6 Ensemble bien ordonné

Définition 4 - 43

$(\text{E}, \leqslant)$ est un **ensemble BIEN ordonné** si $\leqslant$ est un ordre total et si tout sous-ensemble non vide de E contient un plus petit élément.

Est-ce que $(\mathbb{Z}, \leqslant)$ est bien ordonné ?

EXERCICES

Relations binaires

Exercice 4 - 1

Donnez des relations telles que $\mathcal{R} \circ \mathcal{S} \neq \mathcal{S} \circ \mathcal{R}$.

Exercice 4 - 2

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

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

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 $\mathcal{R}^\sim$ et $\mathcal{S}^\sim$.
6. Vrai ou faux ?
   a. $a \mathcal{R} 1$
   b. $\mathcal{R}(b,0)$
   c. $(1,c) \in \mathcal{G}_{\mathcal{R}^\sim}$
   d. $(a,\beta) \in \mathcal{G}_{\mathcal{S}\circ\mathcal{R}}$
   e. $(a,\beta) \in \mathcal{G}_{\mathcal{R};\mathcal{S}}$
7. $\mathcal{R}^\sim$ est-elle une fonction ?
8. Déterminer la relation $\mathcal{S}^\sim;\mathcal{R}^\sim$.
9. Déterminer $\mathcal{S};\mathcal{R}$.
10. Déterminer les relations suivantes : $\{a\} \lhd \mathcal{R}$, $\{b\} \lhd \mathcal{R}$, $\{a,b\} \lhd \mathcal{R}$, $\{1\} \lhd \mathcal{R}^\sim$, $\mathcal{R} \rhd \{1\}$.

Exercice 4 - 3

$\mathcal{R} = (E, F, G_\mathcal{R})$ et $\mathcal{R}' = (E', F', G_{\mathcal{R}'})$ sont deux relations.

1. Dans quel(s) cas $\mathcal{R};\mathcal{R}'$ n'existe pas ?
2. Dans quel(s) cas $\mathcal{R};\mathcal{R}'$ et $\mathcal{R}';\mathcal{R}$ existent simultanément ?

Exercice 4 - 4

$\mathcal{R}_1 = (E, F, G_{\mathcal{R}_1})$ et $\mathcal{R}_2 = (E, F, G_{\mathcal{R}_2})$ sont deux relations de l'ensemble E vers (ou dans) l'ensemble F, $G_{\mathcal{R}_i}$ est le graphe de la relation $\mathcal{R}_i$. On considère la relation $\mathcal{R} = (E, F, G_\mathcal{R})$ dont les éléments (x,y) du graphe vérifient :

$$\text{si } (x,y) \in G_{\mathcal{R}_2} \text{ alors } (x,y) \in G_\mathcal{R}$$
$$\text{si } (x,y) \in G_{\mathcal{R}_1} \text{ et pour tout } z \in F, (x,z) \notin G_{\mathcal{R}_2} \text{ alors } (x,y) \in G_\mathcal{R}$$

Il vous est demandé d'exprimer le graphe de la relation $\mathcal{R}$ en utilisant les opérations ensemblistes usuelles et des ensembles choisis parmi les ensembles suivants : le domaine de $\mathcal{R}_1$ noté $\mathrm{dom}\,\mathcal{R}_1$, le domaine de $\mathcal{R}_2$ noté $\mathrm{dom}\,\mathcal{R}_2$, $G_{\mathcal{R}_1}$, $G_{\mathcal{R}_2}$, E et F.

Exercice 4 - 5

On considère les ensembles $E = \{a,b,c,d\}$ et $F = \{1,2,3\}$.

1. Combien existe-t-il de relations de E vers F ?
2. 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.
   a. Donner un diagramme sagittal de la relation $r;r = r^2$
   b. Donner un diagramme sagittal de la relation $r;r^\sim$
   c. Donner un diagramme sagittal de la relation $r;s$
   d. Donner un diagramme sagittal de la relation $(\{1\} \lhd s^\sim);(r^\sim \rhd \{a,c,d\})$
3. Donner le résultat ou dire que l'écriture n'a pas de sens en expliquant pourquoi :
   a. $r(d) =$
   b. $(r \rhd \{a\})(c) =$
   c. $(r \rhd \{a\})(b) =$
   d. $\{a\} \lhd r(\{d\}) =$
   e. $(\{d\} \lhd r)(d) =$
   f. $(\{a\} \lhd s)(\{a\}) =$

g. $s^{\sim}(\{2\}) =$

h. $(\{d\} \lhd r)(\{a\}) =$

i. $sr(\{1\}) =$

j. $s^{\sim} r^{\sim}(\{1\}) =$

k. $r^{\sim}(\{2\}) =$

l. $(\{a,b\} \lhd s)^{\sim}(1) =$

Exercice 4 - 6

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 $\mathcal{R}^{\sim}$, $\mathcal{R}^{\sim}$ est-elle une fonction ? L'écriture $\mathcal{R}^{\sim}(6)$ est-elle licite ?
5. Calculer, si cela a un sens,
   a. $\mathcal{R}(6)$, $\mathcal{R}(12)$, $\mathcal{R}(\{12\})$, $\mathcal{R}(E)$
   b. $\mathcal{R}^{\sim}(\{6\})$, $\mathcal{R}^{\sim}(F)$, $\mathcal{R}^{\sim}(2)$, $\mathcal{R}^{\sim}(\{2\})$
6. Déterminer le graphe de la restriction de $\mathcal{R}$ à {33;51}
7. E \ {128} $\lhd \mathcal{R}$ est-elle une fonction totale ? Est-elle injective ? surjective ? bijective ?

Exercice 4 - 7

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 4 - 8

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 4 - 9

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 4 - 10

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 4 - 11

E, F et G sont 3 ensembles (tous les trois $\neq \{\}$). $f \in E \to F$, $g \in F \to G$.

1. $g \circ f$ injective $\to$ f injective. ?
2. $g \circ f$ surjective $\to$ g surjective. ?
3. Si f et $f \circ g$ son injectives (surjectives), est-ce que g est injective (surjective) ?

Exercice 4 - 12

A est une partie non vide de l'ensemble E. Qu'appelle-t-on l'injection canonique de A dans E ?

Exercice 4 - 13

Démontrer que la relation transposée de la fonction totale $f = (E, F, G_f)$ est une fonction si, et seulement si, f est injective.

Exercice 4 - 14

E = {a,b,c,d} et F = {1,2,3,4,5,6}. Démontrer qu'il est impossible de construire une surjection et, a fortiori une bijection, de E sur F.

Exercice 4 - 15

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

Exercice 4 - 16

Lorsqu'on est en présence d'ensembles finis, il est facile de savoir s'il existe des injections ou surjections entre ces ensembles puisque si elles existent on est

capable de les construire. Pour des ensembles infinis c'est plus difficile mais on est sûr qu'il n'existe pas de surjection entre un ensemble E et l'ensemble de ses parties $\mathcal{P}(E)$, c'est le théorème de Cantor[b], on va le démontrer :

1. $E = \{a, b, c, d\}$ et f est la fonction totale de E dans $\mathcal{P}(E)$ définie par $f(x) = \{x\}$, déterminer
$$A = \{x \in E \mid x \notin f(x)\}$$
2. $E = \{a, b, c, d\}$ et f est la fonction totale de E dans $\mathcal{P}(E)$ définie par $f(x) = \complement_E\{x\}$, déterminer
$$A = \{x \in E \mid x \notin f(x)\}$$
3. $E = \{a, b, c, d\}$ et f est la fonction totale de E dans $\mathcal{P}(E)$ définie par $f(a) = \{b, c\}$ et $f(x) = E$ pour $x \neq a$, déterminer
$$A = \{x \in E \mid x \notin f(x)\}$$
4. E est un ensemble non vide quelconque, fini ou infini, et f est une fonction totale de E dans $\mathcal{P}(E)$; A est la partie de E définie par $A = \{x \in E \mid x \notin f(x)\}$ et pour prouver que f n'est pas surjective nous allons prouver que A ne peut avoir le moindre antécédent ! Pour cela on suppose que A possède au moins un antécédent u, $f(u) = A$. De deux choses l'une, ou bien $u \in A$ ou bien $u \notin A$; démontrer l'impossibilité de chaque cas et conclure.

Relations binaires sur un ensemble

Exercice 4 - 17

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

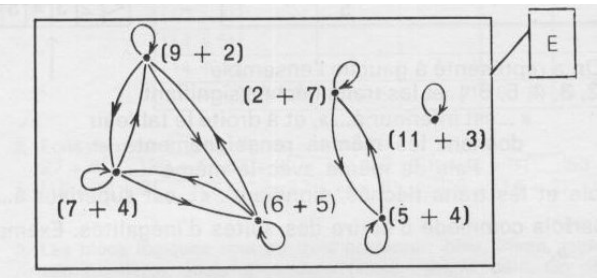
Exercice 4 - 18

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 4 - 19

Commentez cet énoncé de CM1 (on progresse...)



Exercice 4 - 20

1. Donner la représentation matricielle des relations suivantes définies sur $\{1,2,3,4\}$:
   a. $\{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$;
   b. $\{(1,1),(1,4),(2,2),(3,3),(4,1)\}$;
   c. $\{(1,2),(1,3),(1,4),(2,1),(2,3),(2,4),(3,1),(3,2),(3,4),(4,1),(4,2),(4,3)\}$;
   d. $\{(2,4),(3,1),(3,2),(3,4)\}$
2. Quelles sont, parmi ces relations, celles qui sont réflexives ? irréflexives ? 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 4 - 21

1. Énumérer les couples des relations sur $\{1,2,3,4\}$ définies par les matrices suivantes :
   a. $\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$
   c. $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$
   b. $\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$

b. Ne pas confondre avec le fameux théorème de Cantor-Bernstein (qui n'est pas à votre programme) qui dit que s'il existe une injection de E vers F et une injection de F vers E alors il existe une bijection de E sur F.

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 4 - 22

Combien d'éléments non nuls contient la matrice représentant la relation $\mathcal{R}$ sur l'ensemble $\mathcal{E} = \{1,2,3,...,100\}$ 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 4 - 23

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

Exercice 4 - 24

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

Exercice 4 - 25

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. $\mathcal{R}^+$

Utilisez le calcul matriciel ou des dictionnaires ? Quel outil vous semble le plus efficace ?

Exercice 4 - 26

On considère les relations binaires $\mathcal{R}$ et $\mathcal{S}$ sur E = $\{a;b;c;d;e\}$ qui ont pour graphe :

$$G_{\mathcal{R}} = \{(a,a),(a,b),(b,c),(c,b),(e,b)\}$$
$$G_{\mathcal{S}} = \{(a,a),(b,a),(b,b),(c,b),(e,b),(e,d)\}$$

1. Donner une représentation sagittale de $\mathcal{R}$ et de $\mathcal{S}$.
2. Déterminer le graphe de $\mathcal{R} \cup \mathcal{S}$, de $\mathcal{R} \cap \mathcal{S}$, de $\overline{\mathcal{R}}$, de $\mathcal{S}^{\sim}$, de $\mathcal{RS}$ de $\mathcal{SR}$.
3. Déterminer le graphe de $\mathcal{R}^+$ et de $\mathcal{S}^*$.

Exercice 4 - 27

$\mathcal{R}$ est une relation binaire sur l'ensemble E et on suppose que $\mathcal{R}$ n'est pas une relation vide.

1. Rappeler ce qu'est $\mathcal{R}^{k \in \mathbf{N}^*}$.
2. Rappeler ce qu'est $\mathcal{R}^+$.
3. Rappeler ce qu'est $\mathcal{R}^*$.
4. Démontrer que si $x\mathcal{R}^9 z$ et $z\mathcal{R}^{2001} y$ alors $x\mathcal{R}^* y$
5. Traduire correctement $x\mathcal{R}^* y$.
6. Démontrer que $\mathcal{R}^*$ est une relation transitive.

Exercice 4 - 28

$\mathcal{R}$ est une relation binaire sur E

1. Démontrer que $\mathcal{R} \cup \mathcal{R}^{\sim}$ et $\mathcal{R} \cap \mathcal{R}^{\sim}$ sont symétriques.
2. Démontrer que $\mathcal{R}^+$ est transitive.
3. Démontrer que si $\mathcal{R}$ est transitive alors $\mathcal{R} = \mathcal{R}^+$.
4. Démontrer que si $\mathcal{R}$ est réflexive et transitive alors $\mathcal{R} = \mathcal{R}^*$

Exercice 4 - 29

$\mathcal{R}$ est une relation binaire sur E = $\{a,b,c,d,e\}$ dont le graphe est

$$G = \{(a,a),(a,b),(a,c),(b,a),(d,a),(b,e)\}$$

1. Complétez l'écriture : G $\in$
2. Déterminer le graphe de $\mathcal{R}^2$.
3. Déterminer le graphe de $\mathcal{R}^3$, déterminer le graphe de $\mathcal{R}^{k \geq 2}$.
4. Quel est le graphe de $\mathcal{R}^0$?
5. Quel est le graphe de $\mathcal{R}^+$?
6. Quel est le graphe de $\mathcal{R}^*$?
7. $\mathcal{R}^*$ est-elle transitive ? justifier votre réponse.
8. Déterminer le graphe de $\mathcal{R}^{\sim}$, de $\mathcal{R}^{-k}$, $k \in$ N*.

Relations d'équivalence

Exercice 4 - 30

Dans $\mathbb{Z}$ on considère la relation $x\mathcal{R}y \Leftrightarrow x - y \in 3\mathbb{Z} = \{3k \mid k \in \mathbb{Z}\}$. Démontrer que c'est une relation d'équivalence et déterminer l'ensemble quotient ; vérifier que l'ensemble quotient détermine une partition de $\mathbb{Z}$.

Exercice 4 - 31

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 4 - 32

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 $f : x \mapsto x^2$.

Ordre

Exercice 4 - 33

On considère la relation $\preceq$ sur $\mathbb{N}^2$ définie par :

$$(a;b) \preceq (a';b') \Leftrightarrow a \leq a' \text{ et } b \geq b'$$

Est-elle une relation d'ordre ?

Exercice 4 - 34

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

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

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

Exercice 4 - 35

Dans $\mathbb{N}^*$ on considère la relation notée « | » 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

$a \mid b$ si, et seulement si, il existe $k \in \mathbb{N}^*$ 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 $\mathbb{N}^*$.
3. On considère la relation divise dans E = $\{2,4,6,8,10,12\}$.
   a. Construire le diagramme sagittal de cette relation.
   b. Construire le diagramme de Hasse de cette relation.
   c. E admet-il un élément minimum ?
   d. E admet-il un élément maximum ?
   e. E admet-il des éléments minimaux ?
   f. E admet-il des éléments maximaux ?
   g. V = $\{2,4\}$, donner trois majorants de V dans E.
   h. T = $\{8,10\}$ admet-il des majorants dans E ? Donner des majorants de T dans $\mathbb{N}^*$.
   i. Donner des minorants de T dans E.
   j. Donner tous les minorants de T dans $\mathbb{N}^*$.
   k. U = $\{4,6,8\}$ admet-il une borne supérieure dans E ?
   l. U = $\{4,6,8\}$ admet-il une borne supérieure dans $\mathbb{N}^*$?
   m. U = $\{4,6,8\}$ admet-il une borne inférieure dans E ?
   n. U = $\{4,6,8\}$ admet-il une borne inférieure dans $\mathbb{N}^*$?

Exercice 4 - 36

$(E,\leq)$ 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 4 - 37

$\mathbb{R}$ est muni de sa relation d'ordre habituelle et $\mathbb{R}^2$ est ordonné comme produit direct des ensembles ordonnés $(\mathbb{R},\leq)$ et $(\mathbb{R},\leq)$. 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 :

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

$\mathcal{C}$ 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 4 - 38

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 4 - 39

$\mathbb{N}_{10}$ 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 :
   a. $A = \{1, 3, 6\}$
   b. $B = \{2, 3\}$
   c. $C = \{2, 3, 4\}$
   d. $D = \{2, 4\}$

CAML

Le type relation

On reprend la représentation donnée dans le cours utilisant le type ens construit au début du module.

```
type ('a,'b) relation =
  Rel of ('a * 'b ens)  ens ;;
```

On a donc une construction arborescente d'une relation, plus précisément sous forme de peigne :

```
let r =  Rel
   (Ens
      ((1, Ens ('a', Ens ('b', Vide))),
       Ens((2, Ens ('c', Vide)),
           Ens((3, Ens ('b', Vide)),
               Ens((4, Vide),
                   Ens ((5, Ens ('b', Vide)),
                        Vide))))));;
```

Il sera malgré tout plus pratique de rentrer une relation sous la forme d'une liste de couples `(sommet,liste des images)` mais nous aurons d'abord besoin d'une fonction qui calcule la réunion de deux relations qui sera calculée plus tard.
Pour faciliter la **lecture** d'une relation, nous allons créer une fonction qui « affiche » la relation sous forme d'une liste. On utilise `fst` et snd qui donnent respectivement la première et la seconde composante d'un couple.

```
let rec liste_of_ensemble = function
  |Vide -> []
  |Ens(t,q) -> t::(liste_of_ensemble q);;


let rec liste_of_relation rel =
  match rel with
  |Rel(Vide) -> []
  |Rel(Ens(tete,queue)) -> (fst tete,
       liste_of_ensemble (snd tete))::(
       liste_of_relation (Rel(queue)));;
```

Par exemple dans l'interpréteur :

```
# liste_of_relation r;;
- : (int * char list) list =
[(1, ['a'; 'b']); (2, ['c']); (3, ['b']); (4, [])
    ; (5, ['b'])]
```

Images et antécédents

Créez une fonction `domaine` qui renvoie le domaine d'une relation :

```
# domaine r;;
- : int ens = Ens (1, Ens (2, Ens (3, Ens (4, Ens
     (5, Vide)))))
```

Faites de même pour le codomaine :

```
# codomaine r;;
- : char ens = Ens ('a', Ens ('c', Ens ('b', Vide
    )))
```

Vous utiliserez les fonctions `ajoute` et `union` mises au point pour le type ens.
Créez une fonction `ens_images relation sommet` qui renvoie l'ensemble des images de `sommet` par `relation` :

```
# ens_images r 1;;
- : char ens = Ens ('a', Ens ('b', Vide))
```

Créez une fonction `a_images relation s1 s2` qui teste si `s2` est une image de `s1` par `rel` puis une fonction `a_antecedent s2 s1` qui teste si `s1` est un antécédent de `s2` par `rel`.

```
# a_antecedent r 'a' 1;;
- : bool = true
```

Réunion de relations

Analysez les fonctions suivantes :

```
let rec fusionne_image_ens couple ens_couples =
  match ens_couples with
  |Vide -> Ens(couple,Vide)
  |Ens(t,q) ->
    if fst t = fst couple then Ens((fst t,union (
        snd t) (snd couple)) ,q)
    else Ens(t,fusionne_image_ens couple q);;

let fusionne_image_rel couple relation =
    match relation with
    |Rel(Vide) -> Rel(Ens(couple,Vide))
    |Rel(Ens(a,b)) -> Rel(ajoute couple (Ens(a,b)
        ));;

let rec rel_union r1 r2 =
    match r1 with
    |Rel(Vide) -> r2
    |Rel(Ens(tete,queue)) -> rel_union (Rel(queue
        )) (fusionne_image_rel tete r2);;

let (+@) r1 r2 = rel_union r1 r2;;
```

On aura besoin d'obtenir la liste fusionnée des images par $\mathcal{R}$ des éléments d'un ensemble de sommets :

```
let rec images_d_un_ens rel ens =
...
```

Par exemple :

```
# images_d_un_ens r (Ens(1,Ens(2,Ens(3,Vide))));;
- : char ens = Ens ('a', Ens ('c', Ens ('b', Vide
    )))
```

On peut alors créer une fonction qui renvoie la relation $\mathcal{R}$ suivie de $\mathcal{S}$:

```
let suivie_de r s =
...
```

Puis sous forme infixe :

```
let (@@) r s = suivie_de r s;; (* opérateur
    infixe de la composition *)
```

On peut itérer la composition :

```
let rec iter_compo r n =
...
```

Inclusion de relations

Tester l'inclusion de deux relations c'est tester celle de leurs graphes. Nous avons donc besoin de créer le produit cartésien de deux ensembles. À quoi peuvent servir ces deux fonctions :

```
let rec couples el ens =
  match ens with
    |Vide -> Vide
    |Ens(t,q) -> Ens((el,t),couples el q);;


let rec produit_cart ens1 ens2 =
  match ens1 with
    |Vide -> Vide
    |Ens(t,q) -> union (couples t ens2) (
        produit_cart q ens2);;
```

Comment les utiliser pour le test d'inclusion ?

Propriétés des relations

test « pour tout »

Nous aurons besoin de tester si tous les éléments du domaine vérifient un certain prédicat. Vous allez donc créer une fonction `pour_tout pred ens` qui teste si tous les éléments de ens vérifient le prédicat `pred`.
Vous obtiendrez alors :

```
# let positif nb  =
  nb >= 0;;
  val positif : int -> bool = <fun>

# let ensemble_1 = Ens(-1,Ens(1,Ens(2,Ens(3,Vide)
    )));;
val ensemble_1 : int ens = Ens (-1, Ens (1, Ens
    (2, Ens (3, Vide))))

# pour_tout positif ensemble_1;;
- : bool = false
```

Test de réflexivité

On teste si chaque sommet appartient à son image :

```
let est_reflexive1 r =
...
```

```
# let u = relation ([
                (1,[1]);
                (2,[1;2]);
                (3,[2]);
                (4,[]);
                (5,[2;4])
                ]);;
val u : (int, int) relation

# est_reflexive u;;
- : bool = false
```

Test de transitivité

On teste si $\mathcal{R}^2 \subseteq \mathcal{R}$:

```
let est_transitive r =
...
```

Test de symétrie

Teste si chaque sommet appartient à la liste des images de ses images par $\mathcal{R}$:

```
let est_symetrique r =
...
```

```
# let v = relation ([
                (1,[1;2]);
                (2,[1;2;3;5]);
                (3,[2]);
                (4,[5]);
                (5,[2;4])
                ]);;
val v : (int, int) relation

# est_symetrique v;;
- : bool = true
```

On peut tester aussi l'antisymétrie : aucun sommet a n'appartient à la liste des images de ses images par $\mathcal{R}$:

```
let est_antisymetrique r =
...
```

Fonctions

La relation est une fonction si, et seulement si, la longueur de l'ensemble des images de chaque élément de son domaine est inférieure à 1 :

```
let est_fonction r =
...
```

Créez ensuite une fonction qui teste si une fonction est totale :

```
let est_totale r =
...
```

On aura par la suite besoin de la fonction suivante : que fait-elle ?....

```
let trouve_tout pred ens  =
  let  rec remplit pred ens panier =
  match ens with
    |Vide -> panier
    |Ens(t,q) ->
      if pred t then remplit pred q (ajoute t
          panier)
      else remplit pred q panier
  in remplit pred ens Vide ;;
```

et celle-ci ?...

```
let transposee r =
  let rec recherche_antecedents ens =
    match ens with
    |Vide -> Rel(Vide)
    |Ens(tete,queue) ->
      fusionne_image_rel (tete, trouve_tout (
          a_antecedent r tete) (domaine r)) (
          recherche_antecedents queue)
  in recherche_antecedents (codomaine r);;
```

Une relation est injective si, et seulement si, sa transposée est une ... :

```
let est_injective r =
...
```

Une relation est surjective si, et seulement si, sa transposée est ... :

```
let est_surjective r =
...
```

ou si, et seulement si, sa transposée suivie d'elle-même est ... :

```
let est_surjective2 r =
....
```

Enfin :

```
let est_bijective r =
...
```