Chapitre 3

Calcul propositionnel classique

Ce chapitre a pour but de présenter, dans un langage proche de celui que nous utiliserons dans le reste du cours, des choses qui sont censées être bien connues de tous.

3.1 Variables propositionnelles, connecteurs et formules

Dans cette section, on définit les objets étudiés, qu'on appellera des *formules propositionnelles* (ou, en abrégé, formules).

3.1.1 Variables propositionnelles et connecteurs

On se donne un ensemble $\mathcal{V}$ dont les éléments sont appelés *variables propositionnelles* ; par convention, ces variables sont souvent notées par des lettres majuscules, par exemple $\mathcal{V} = \{P, Q, R, S\}$.

On se donne également un ensemble de *connecteurs*, typiquement $\{\neg, \wedge, \vee, \rightarrow\}$, dont chacun se voit attribuer une *arité* (un nombre d'arguments) : 1 pour le connecteur $\neg$ (négation), 2 pour les autres. On pourrait définir des connecteurs d'arité plus grande, au prix d'une légère complication des notations.

Prononciation : le symbole $\neg$ (négation) se lit "non" ; $\wedge$ (conjonction) se lit "et" ; $\vee$ (disjonction), "ou". Le symbole $\rightarrow$ (implication) peut se lire "implique" ; on peut également lire $A \rightarrow B$ en "si A, alors B".

3.1.2 Formules et sous-formules

Une fois donné tout cela, on définit une *formule propositionnelle*, ou plus simplement une *formule*, par une définition inductive :

— chaque variable propositionnelle définit une formule (formule atomique) ;

— pour chaque connecteur unaire c et chaque formule F, $(c\ F)$ est une formule ;
— pour chaque connecteur binaire c et chaques deux formules F_1 et F_2, $(F_1\ c\ F_2)$ (ou, en notation préfixe, $c\ F_1\ F_2$) est une formule.

Quand on parle de définition inductive, cela sous-entend deux conditions :
— toute formule est d'une des trois formes ci-dessus, et d'une seule ;
— deux formules ne sont considérées comme "égales" que si elles sont du même type, avec le même connecteur, et, le cas échéant, la ou les mêmes sous-formules.

Essentiellement, on a une définition équivalente à une définition de type en `ocaml` qui serait la suivante :

```
type formule =
P | Q | R | S
| Non of formule
| Et of formule * formule
| Ou of formule * formule
| Implique of formule * formule
```

Une telle définition n'est pas optimale, car elle fige à la fois l'ensemble des variables propositionnelles et celui des connecteurs ; il serait préférable, même si c'est plus verbeux, de définir des types pour les variables et pour les ensembles de connecteurs de chaque arité, avec trois constructeurs de formules :

```
type var = P | Q | R | S

type unaire = Not

type binaire = Et | Ou | Implique

type formula =
| Atome of var
| Unaire of unaire * formula
| Binaire of binaire * formula * formula
```

Notre premier type permet d'écrire l'équivalent de $(P \vee (\neg Q))$ sous la forme

```
Ou(P,Non(Q))
```

au lieu du plus verbeux

```
Binaire(Ou,Atome(P),Unaire(Not, Atome(Q))
```

mais le fait d'avoir un type à part pour les variables rendra plus facile la description d'une valuation.

Exercice 1. *Écrire des fonction* `ocaml` *de conversion (dans les deux sens) entre les types* `formule` *et* `formula`*.*

Il faut donc, à ce stade, voir les formules comme des objets formels ; mais on peut aussi leur attribuer un *sens*, en associant à chaque connecteur une

sémantique définie, pour chaque connecteur, par une fonction booléenne : fonction d'une variable booléenne pour les connecteurs unaires, de deux variables pour les connecteurs binaires. Combinées avec des valeurs de vérité pour chaque variable, cela permettra d'attribuer une valeur de vérité à chaque formule.

La représentation typographique des formules, avec des parenthèses, permet également de voir qu'elles sont équivalentes à des arbres dont les noeuds internes sont d'arité 1 ou 2, dont les noeuds internes sont étiquetés par des opérateurs de même arité et les feuilles par des variables propositionnelles.

Cette interprétation en termes d'arbres permet de définir la notion de *sous-formule* d'une formule donnée : une formule B est une sous-formule d'une formule A, si B est la formule correspondant à un des sous-arbres [1] de A.

Exemple : la formule $(P{\to}Q){\vee}(P{\wedge}((\neg Q){\wedge}R))$ a 8 sous-formules distinctes :

— P, Q, R, qui correspondent aux feuilles de l'arbre ;
— $(P{\to}Q)$, $(\neg Q)$, $((\neg Q) \wedge R)$, $(P \wedge ((\neg Q) \wedge R))$, qui correspondent aux noeuds internes autres que la racine ;
— la formule elle-même.

Deux de ces sous-formules apparaissent plus d'une fois comme sous-formules.

On peut également alléger la typographie en supprimant certaines parenthèses, en définissant des règles de priorité :

— $\neg$ est prioritaire sur les autres connecteurs ;
— $\wedge$ est prioritaire sur $\vee$ et $\to$;
— $\vee$ est prioritaire sur $\to$;
— chaque connecteur binaire associe de droite à gauche, de sorte que $P{\to}Q{\to}R$ se lit comme $P{\to}(Q{\to}R)$ et non comme $(P{\to}Q){\to}R$.

Ces règles ne sont pas exactement celles utilisées par `ocaml`, où `||` et `&&` ont la même priorité et l'association se fait de gauche à droite ; ce sont les règles de priorité standard de `coq`.

Attention, quand la formule n'est pas totalement parenthésée, certaines séquences de symboles de la formule, qui décrivent des formules, ne sont pas pour autant des sous-formules : ainsi $P{\to}Q$ n'est pas une sous-formule de $P{\to}Q{\to}R$ (qui, sous forme parenthésée, s'écrit $(P{\to}(Q{\to}R))$: sous cette forme, la confusion n'est normalement pas possible).

Exercice 2. *Parenthéser, et représenter sous forme d'arbres, les formules suivantes :*

— $P{\to}\neg Q \wedge R$
— $P{\to}Q \wedge Q{\to}R$
— *d'autres*

3.2 Interprétation standard des connecteurs

Définissons pour chaque connecteur la fonction booléenne correspondante. Nous utilisons la lettre V pour "vrai" (`true` ou `True` dans beaucoup de langages

1. Au sens usuel de sous-arbre : l'arbre composé d'un noeud de l'arbre et de tous ses descendants

de programmation) et F pour "faux" (`false` ou `False` dans divers langages) ; la seule chose que nous nous interdirons systématiquement, c'est d'utiliser des valeurs entières 1 et 0 [2]. Nous les représentons par des "tables de vérité", avec une ligne pour chaque combinaison de valeurs de vérité des sous-formules.

La fonction correspondant au connecteur $\neg$ ("non") est simplement la fonction "l'autre valeur booléenne" :

A	$\neg A$
V	F
F	V

Les connecteurs $\wedge$ ("et") et $\vee$ ("ou") ne posent généralement pas de problème ; il faut se souvenir que le "ou" des informaticiens, des logiciens et des mathématiciens est systématiquement inclusif :

A	B	$A \wedge B$
V	V	V
V	F	F
F	V	F
F	F	F

A	B	$A \vee B$
V	V	V
V	F	V
F	V	V
F	F	F

L'interprétation de l'implication $\rightarrow$ nécessite de se souvenir de ce que "faux implique n'importe quoi" est vrai :

A	B	$A \rightarrow B$
V	V	V
V	F	F
F	V	V
F	F	V

Exercice 3. *Écrire deux fonctions `ocaml` d'interprétation des connecteurs unaires et binaires : la première sera de type `unaire→var→bool`, la seconde sera de type `binaire →var →var→bool`.*

Exercice 4. *Que serait la table de vérité pour un connecteur $\leftrightarrow$ d'équivalence ? Comment faudrait-il modifier les définitions `ocaml` précédentes pour ajouter un tel connecteur aux formules considérées ?*

3.2.1 Valuations et valeurs de vérité

Une *valuation* est l'affectation, à chaque variable propositionnelle, d'une valeur booléenne, ou valeur de vérité – donc, techniquement, c'est une fonction de l'ensemble $\mathcal{V}$ des variables propositionnelles, vers l'ensemble $\{V, F\}$.

Une valuation v, définie *a priori* seulement sur les variables propositionnelles, s'étend naturellement à n'importe quelle formule propositionnelle par une définition récursive qui suit la structure de la formule elle-même : en conservant la notation v pour la valuation étendue,

2. C'est l'influence néfaste du langage C que nous combattons ici.

— si A est une formule atomique, $v(A)$ est déjà connue ;
— si A est une formule de la forme $c(B)$, où B est une formule quelconque et c un connecteur unaire dont la fonction d'interprétation est f_c, alors $v(A) = f_c(v(B))$;
— si A est une formule de la forme $c(B, C)$, où B et C sont des formules quelconques, et c un connecteur binaire dont la fonction d'interprétation est f_c, alors $v(A) = f_c(v(B), v(C))$.

On se convainc aisément que cette description inductive de v à partir ses valeurs sur les variables propositionnelles (et des interprétations des connecteurs) est bien définie : en effet, si on considère que la *taille* d'une formule est donnée par son nombre de connecteurs (nombre de noeuds internes de l'arbre correspondant), la définition récursive de v n'utilise les valeurs de v que sur des formules de taille strictement inférieure, avec la taille 0 (les formules atomiques) comme cas de base. C'est même ce qu'on appelle une induction *structurelle* : la valeur de $v(A)$ est définie à partir des valeurs de v sur des *sous-formules* de A.

Exercice 5. *Écrire une fonction* `ocaml` *prenant en entrée une formule propositionnelle et une fonction valuation (donc de type* `var→bool`*), et retournant la valeur de la valuation étendue sur la formule donnée.*

On peut représenter l'ensemble des valuations possibles d'une formule et de ses variables au moyen d'une table de vérité : cette table aura une ligne pour chaque valuation possible de l'ensemble des variables propositionnelles (donc, pour une formule à k variables, 2^k lignes). Typiquement, on prévoit une colonne dans la table pour chaque sous-formule, ce qui facilite le remplissage. Si la colone d'une formule est toujours à droite de celle de ses sous-formules, on peut remplir ligne par ligne et de gauche à droite, en consultant à chaque fois des valeurs de la même ligne déjà remplies.

Voici par exemple la table de vérité complète pour la formule $((P \to Q) \to P) \to P$:

P	Q	$P \to Q$	$(P \to Q) \to P$	$((P \to Q) \to P) \to P$
V	V	V	V	V
V	F	F	V	V
F	V	V	F	V
F	F	V	F	V

Les deux premières colonnes sont posées *a priori* : on veut les quatre possibilités. Puis, on peut remplir le reste de la table soit ligne par ligne, soit colonne par colonne, en prenant seulement soin de procéder de gauche à droite dans chaque ligne.

3.3 Formules équivalentes, tautologies

Deux formules propositionnelles sont dites *équivalentes* si, pour *toute* valuation, elles ont la même valeur de vérité. Autrement dit, si leurs colonnes dans une table de vérité sont identiques ligne à ligne.

Deux formules équivalentes ne pas être considérées comme *égales*, mais du point de vue des valeurs de vérité, elles "disent la même chose" : quelque soit le sens qu'on donne aux diverses variables propositionnelles qui apparaissent dans les deux formules, les deux formules sont soit toutes les deux vraies, soit toutes les deux fausses.

Par exemple, les formules $P \to Q$ et $\neg Q \to \neg P$ sont deux formules équivalentes, ce qu'on peut vérifier en calculant leurs tables de vérité. Il en est de même des formules P et $\neg lnotP$.

Une formule est une *tautologie* si *toute* valuation lui donne la valeur V. C'est, en ce sens, une formule "toujours vraie" (dans le langage courant, le terme de tautologie a tendance à prendre un sens un peu plus restrictif, pour désigner une affirmation qui est "évidemment vraie").

Par exemple, les formules $P \to P$, $P \vee \neg P$, $(P \to Q) \vee (Q \to P)$, $((P \to Q) \to P) \to P$ sont des tautologies.

3.3.1 Prouver une équivalence ou une tautologie

Il y a un lien entre la notion d'équivalence entre deux formules, et la notion de tautologie : en effet, deux formules A et B sont équivalentes si et seulement si la formule $A \leftrightarrow B$ est une tautologie (ou, si on n'a pas de connecteur $\leftrightarrow$: si la formule $(A \to B) \wedge (B \to A)$ est une tautologie). Inversement, une formule est une tautologie si et seulement si elle est équivalente à une formule déjà connue pour être une tautologie, comme par exemple $P \to P$.

À ce stade, nous avons un moyen (et un seul) de prouver qu'une formule est une tautologie, ou que deux formules sont équivalentes : établir des tables de vérité, ce qui revient à vérifier la définition ("toute valuation…" : il faut vérifier toutes les lignes de la table de vérité). Cela peut être très fastidieux (rappelons qu'une formule faisant intervenir k variables, possède 2^k valuations différentes), mais il n'y a pas de doute sur le fait que c'est une tâche qu'on peut confier à un ordinateur. Reconnaissons tout de même qu'une longue table de vérité n'est pas très éclairante sur "ce qui se passe" : si l'on cherche à se forger une intuition sur le *pourquoi*, une table de vérité à 16 ou 32 lignes n'est pas d'une grande aide.

En revanche, quand il s'agit de prouver qu'une formule n'est **pas** une tautologie, ou que deux formules ne sont **pas** équivalentes, c'est potentiellement plus simple : il suffit d'exhiber **une** valuation (une ligne de la table de vérité) particulière, du moment que cette valuation donne une valeur F à la formule testée (ou des valeurs distinctes aux deux formules, quand il s'agit de prouver qu'elles ne sont pas équivalentes). Ceci étant dit, cette remarque n'aide pas particulièrement quand il s'agit de **trouver** une telle valuation : nous n'avons pas *a priori* de méthode plus intelligente que de tester toutes les valuations, à moins d'avoir une compréhension intuitive du "sens" de la formule qui nous mette sur la piste d'une valuation lui attribuant la valeur F.

3.3.2 Formules et discours en langue naturelle

Il est naturel de se demander ce que représentent les variables propositionnelles et les formules que l'on écrit avec. Si l'on convient d'interpréter les différents connecteurs par leur équivalent en langue naturelle ($\neg$, par la négation ; $\wedge$, par un *et* ; $\vee$, par un *ou* inclusif ; $\rightarrow$, par un "si…alors"), on peut potentiellement associer à chaque variable propositionnelle une affirmation donnée, et "traduire" la formule – ce qui a vite tendance à donner des phrases un peu alambiquées et qu'on a envie de reformuler.

Par exemple, si on considère la formule $(P{\rightarrow}Q){\rightarrow}(Q{\rightarrow}R){\rightarrow}P{\rightarrow}R$, et qu'on associe

— à la variable P, l'affirmation "il pleut" ;
— à la variable Q, l'affirmation "le sol est mouillé" ;
— à la variable R, l'affirmation "je glisse",

les formules se traduisent ainsi :

— $P{\rightarrow}Q$: "s'il pleut, alors le sol est mouillé" ;
— $Q{\rightarrow}R$: "si le sol est mouillé, alors je glisse" ;
— $P{\rightarrow}R$: "s'il pleut, alors je glisse"
— la formule complète peut se reformuler en "si on considère que, s'il pleut, le sol est mouillé ; et que, si le sol est mouillé, je glisse ; alors, s'il pleut, je glisse" (ici, on a intégré le fait que $A{\rightarrow}B{\rightarrow}C$ peut s'interpréter en "si A et B, alors C" au lieu de "si A, alors si B, alors C").

Mais le fait que la formule en question soit une tautologie se traduit surtout par le fait que, *quelles que soient* les traductions qu'on donne aux différentes variables propositionnelles (à condition bien sûr de toujours interpréter la même variable de la même manière), on obtient une affirmation "vraie".

C'est à la lumière de cette remarque qu'il faut comprendre la notion de substitution.

3.3.3 Substitution uniforme

Considérons une formule A, qui peut être arbitrairement complexe ; ainsi qu'une variable P, et une formule B. Nous voulons définir, de manière non ambigüe, la formule obtenue en "remplaçant" partout dans A, les occurrences de P par B (il est fortement conseillé, pour ne pas se tromper quand on fait l'opération à la main, de parenthéser complètement la formule B).

Par exemple, si A est la formule $(P \wedge Q){\rightarrow}(P \vee Q)$, et B la formule $P{\rightarrow}Q$, le résultat (pour la variable P) sera la formule $((P{\rightarrow}Q) \wedge Q){\rightarrow}((P{\rightarrow}Q) \vee Q)$.

Le plus simple est de donner une définition récursive, qui suive la structure de la formule A.

Définition 1. *On définit, pour toute formule A, toute variable propositionnelle v, et toute formule B, la formule $A[v/B]$, par :*

— *Si A est une formule atomique, réduite à une variable, alors*
  — *si cette variable est v, $A[v/B] = B$;*
  — *si c'est une autre variable, $A[v/B] = A$.*

— *Si A est de la forme $c(A')$, où c est un connecteur unaire, alors $A[v/B] = c(A'[v/B])$;*
— *Si A est de la forme $c(A', A'')$, où c est un connecteur binaire, alors $A[v/B] = c(A'[v/B], A''[v/B])$.*

Il n'est pas difficile d'étendre la définition pour substituer *simultanément* à plusieurs variables, chacune une formule donnée ; la notation serait $A[v_1/B_1, \ldots, v_k/B_k]$ (attention : on ne peut pas se permettre de faire les substitutions variable par variable : le résultat dépendrait de l'ordre si, par exemple, les variables substituées apparaissent dans les formules qu'on leur substitue).

Exercice 6. *En s'inspirant de la définition de $A[v/B]$, donner une définition récursive de $A[v_1/B_1, \ldots, v_k/B_k]$; on supposera que les variables $v_1, \ldots, v_k$ sont bien distinctes.*

Exercice 7. *Écrire une fonction* `ocaml`*, de type* `var →formula →formula →formula`*, qui réalise la substitution. Vous pouvez également écrire une fonction pour la substitution multiple, par exemple de type* `(var * formula) list →formula →formula`*, mais c'est un peu plus complexe, et il faut choisir ce que l'on retourne si la liste contient deux couples avec la même variable.*

L'équivalent formaliste de la remarque de la sous-section précédente, serait alors le théorème suivant :

Théorème 1 (de substitution uniforme)**.** *Soient $x_1, \ldots, x_k$ des variables distinctes quelconques, et $B_1, \ldots, B_k$ des formules quelconques.*

Alors, si A est une tautologie, $A[x_1/B_1, \ldots, x_k/B_k]$ est également une tautologie.

De plus, pour des formules A et A' quelconques, si A et A' sont équivalentes, alors $A[x_1/B_1, \ldots, x_k/B_k]$ et $A'[x_1/B_1, \ldots, x_k/B_k]$ sont aussi équivalentes.

Avant de donner une preuve rapide de ce théorème, remarquons une chose : une formule obtenue par substitution peut être beaucoup plus grosse que la formule dans laquelle on a fait la substitution, et peut avoir beaucoup plus de variables distinctes. Repérer qu'une formule est obtenue par substitution dans une formule simple peut ainsi faciliter grandement la tâche si l'on cherche à montrer qu'il s'agit d'une tautologie.

À titre d'exemple, $P{\to}P$ est une tautologie, et ne fait intervenir qu'une seule variable (donc sa table de vérité n'a que deux lignes). Mais si l'on cherche à se convaincre, en écrivant sa table de vérité, que la formule $((P \wedge S) \vee (Q{\to}R)){\to}((P \wedge S) \vee (Q{\to}R))$ est une tautologie, la table de vérité aura 16 lignes (4 variables), alors que cette formule n'est autre que $(P{\to}P)[P/((P \wedge S) \vee (Q{\to}R))]$, et donc le théorème nous permet d'affirmer sans peine qu'il s'agit d'une tautologie.

Démonstration. Tout d'abord, remarquons que le second énoncé est une conséquence du premier : en effet, dire que A et A' sont équivalentes revient à dire

que $A \leftrightarrow A'$ est une tautologie, et $(A \leftrightarrow A')[x_1/B_1, \ldots, x_k/B_k]$ est la formule $(A[x_1/B_1, \ldots, x_k/B_k]) \leftrightarrow (A'[x_1/B_1, \ldots, x_k/B_k])$ d'après la définition de la substitution. Il suffit donc de prouver le premier énoncé.

Contrairement à ce qu'on pourrait penser, la preuve n'est pas par induction sur la structure de la formule A. En effet, une telle preuve nous amènerait à raisonner sur les sous-formules de la tautologie considérée, or les sous-formules d'une tautologie ne sont en général pas des tautologies.

Supposons que A soit une tautologie, et posons, pour simplifier les notations, $A' = A[x_1/B_1, \ldots, x_k/B_k]$. Considérons une valuation v pour les variables de A' ; nous devons prouver que l'on a $v'(A') = V$.

Notons $q_1, \ldots, q_k$ les valeurs de vérité de v sur les formules $B_1, \ldots, B_k$ ($q_i = v(B_i)$). Alors d'après la façon dont est définie la valuation, $v(A') = v'(A)$, où v' est une nouvelle valuation, où les valeurs des variables $x_1, \ldots, x_k$ sont remplacées par $q_1, \ldots, q_k$ respectivement (les autres valeurs étant inchangées) [3]. Mais on sait que toute valuation donne à la formule A la valeur V (c'est la définitin d'une tautologie), donc c'est le cas en particulier de v'. Par conséquent, on a bien $v(A') = V$.

Résultat, A' est bien une tautologie. □

3.3.4 Remplacement

Le théorème de substitution uniforme manque un peu de subtilité : il ne "marche" que si on remplace *toutes* les occurrences d'une même variable par la même formule.

Nous énonçons ici un second théorème, beaucoup plus chirurgical dans les transformations qu'il fait subir aux formules.

Prenons deux formules équivalentes, par exemple $\neg Q \to \neg P$ et $P \to Q$. Nous prenons volontairement une seconde formule "plus simple" que la première.

Maintenant, prenons une formule A, possiblement assez complexe, et dont $\neg Q \to \neg P$ soit une sous-formule. Par exemple, $((P \vee Q) \wedge (\neg Q \to \neg P)) \to (\neg Q \to \neg P)$. En l'occurrence, notre formule A possède deux occurrences de $\neg Q \to \neg P$ comme sous-formule.

On a un peu envie de voir $P \to Q$ comme une "forme simplifiée" de $\neg Q \to \neg P$, et de faire des simplifications dans la formule A, pour obtenir, par exemple, A' qui serait $((P \vee Q) \wedge (P \to Q)) \to (\neg Q \to \neg P)$ (une simplification) ou A'' qui serait $((P \vee Q) \wedge (P \to Q)) \to (P \to Q)$ (deux simplifications).

Ces trois formules A, A' et A'' ne sont pas *égales*, mais elles sont *équivalentes* (on peut le vérifier : les tables de vérité n'ont que 4 lignes). C'est essentiellement ce que nous garantit le théorème de remplacement.

Théorème 2 (de remplacement)**.** *Soient B et C deux formules équivalentes, et soit A une formule quelconque.*

3. Techniquement, c'est cette affirmation qui devrait être prouvée, et qu'on prouverait par induction structurelle sur la formule A. Nous laissons la rédaction propre de cette preuve au lecteur à la recherche d'un exercice pour s'occuper.

Alors, si A' est n'importe quelle formule obtenue en remplaçant, dans A, une ou plusieurs occurrences de B (comme sous-formules) par C, les formules A et A' sont équivalentes.

Nous ne donnons pas de preuve de ce théorème, mais l'idée est la suivante : dans le calcul de la valuation pour une formule le long de son arbre, l'influence d'une sous-formule sur les valeurs calculées pour ses ancêtres se limite à sa valuation ; donc deux formules différant seulement par des sous-arbres qui représentent des formules équivalentes, ne "verront" plus la différence au niveau de la valuation à la racine.

3.4 Lois de De Morgan et réécritures de formules

3.4.1 Lois de De Morgan

On appelle souvent *lois de De Morgan* les deux équivalences suivantes : pour toutes formules A et B,

— la formule $\neg(A \vee B)$ est équivalente à la formule $(\neg A) \wedge (\neg B)$;

— la formule $\neg(A \wedge B)$ est équivalente à la formule $(\neg A) \vee (\neg B)$.

Ces deux équivalences se vérifient aisément, en calculant à chaque fois les deux tables de vérité (à 4 lignes chacune). On calcule ici comme si A et B étaient des variables et non pas des formules plus complexes ; c'est le théorème de substitution uniforme qui nous assure que cela suffit.

Ajoutons à cela deux équivalences supplémentaires :

— la formule $\neg(\neg A)$ est équivalente à la formule A ;

— la formule $A \to B$ est équivalente à la formule $(\neg A) \vee B$.

Ces quatre équivalences peuvent s'utiliser pour définir des *règles de réécriture* qui permettent, partant d'une formule quelconque, d'en obtenir une nouvelle qui lui est équivalente et qui suit une certaine forme.

On considère ces quatre équivalences comme orientées : partant d'une formule A,

— chaque sous-formule de la forme $\neg(X \vee Y)$ (un noeud négation ayant comme fils un noeud disjonction) est remplacé par $(\neg X) \wedge (\neg Y)$ (un noeud conjonction dont les deux fils sont des noeuds négation)

— chaque sous-formule de la forme $\neg(X \wedge Y)$ (noeud négation ayant comme fils un noeud disjonction) est remplacé par $(\neg X) \vee (\neg Y)$ (noeud disjonction dont les deux fils sont des noeuds négation)

— chaque sous-formule de la forme $X \to Y$ est remplacé par $(\neg X) \vee Y$;

— chaque double négation $\neg(\neg X)$ est remplacée par X (disparition de deux noeuds négation dont l'un est le parent de l'autre.

Ces règles peuvent être appliquées de manière répétée, tant que l'une d'elles peut l'être ; chaque application d'une règle donne une nouvelle formule équivalente. Il n'est pas très compliqué de voir que le processus termine forcément : on finit forcément par atteindre une formule où aucune des quatre règles ne peut

s'appliquer. (Il est un peu moins évident de voir que le résultat final ne dépend pas de l'ordre dans lequel on fait les transformations ; nous ne nous servirons pas de cette propriété.)

La formule obtenue en fin de ce processus n'a plus que des connecteurs $\wedge$, $\vee$ et $\neg$ (s'il restait des connecteurs $\rightarrow$, on pourrait appliquer la dernière règle), et de plus, le connecteur $\neg$ n'est plus appliqué que sur des variables propositionnelles (car une fois qu'il n'y a plus d'implication, un connecteur $\neg$ appliqué à autre chose qu'une variable permettrait l'utilisation d'une des trois premières règles).

Au final, on obtient des formules qui peuvent se représenter sous la forme d'arbres binaires dont chaque noeud interne est étiqueté $\wedge$ ou $\vee$, et chaque feuille, soit par une variable propositionnelle, soit par la négation d'une variable propositionnelle.

On parle parfois de "mettre la formule A sous une forme donnée", mais il est important de retenir que, de notre point de vue, on parle en fait de deux formules distinctes, qui sont "seulement" équivalentes :

— la formule $\neg\neg P$ n'est pas égale à la formule P, elle lui est seulement équivalente ;
— la formule $P \rightarrow \neg Q$ n'est pas égale à la formule $\neg P \vee \neg Q$, ele lui est seulement équivalente ;
— on peut inventer à loisir des exemples de plus en plus compliqués.

3.4.2 Associativité et commutativité

La conjonction ($\wedge$) et la disjonction ($\vee$) ont des propriétés qui s'apparentent à celles de la multiplication et de l'addition des réels, si l'on s'arrête à l'équivalence entre formules :

— les formules $A \wedge B$ et $B \wedge A$ sont équivalentes (commutativité de la conjonction) ;
— les formules $A \vee B$ et $B \vee A$ sont équivalentes (commutativité de la disjonction) ;
— les formules $(A \wedge B) \wedge C$ et $A \wedge (B \wedge C)$ sont équivalentes (associativité de la conjonction) ;
— les formules $(A \vee B) \vee C$ et $A \vee (B \vee C)$ sont équivalentes (associativité de la disjonction).

Ces équivalences, dont la vérification est laissée en exercice (mécanique), ont pour conséquence que, dès qu'on ne se préoccupe pas trop de formalisme, on a tendance à oublier les différences entre ces formules équivalentes, et à écrire simplement $(P \vee Q \vee \neg R) \wedge (Q \vee \neg P \vee \neg R) \wedge (\neg P \vee \neg Q \vee R)$ sans se préoccuper de la façon dont s'agencent les parenthèses manquantes : les formules parenthésées correspondantes sont, de toute manière, toutes équivalentes.

(Les praticiens de la programmation savent bien que l'addition des nombres en virgule flottante, elle, n'est pas associative, et sont habitués, ou devraient l'être, à parenthéser leurs expressions pour éviter les mauvaises surprises.)

Attention : si on n'a pas écrit ici que l'implication $\rightarrow$ était associative, ou commutative, ce n'est pas un oubli : ce n'est pas le cas ! En revanche, pour l'implication, on a une forme de commutativité des prémisses.

Exercice 8. — *Les formules $P \to Q$ et $Q \to R$ sont-elles équivalentes ?*
— *Les formules $(P \to Q) \to R$ et $P \to (Q \to R)$ sont-elles équivalentes ?*
— *Les formules $P \to Q \to R$ et $Q \to P \to R$ sont-elles équivalentes ?*

3.4.3 Distributivité

La propriété usuelle de distributivité de la multiplication par rapport à l'addition, se traduit par le fait que, pour n'importe quels nombres x, y, z, les expressions $(x + y) \times z$ et $(x \times z) + (y \times z)$ donnent le même résultat.

Conjonction et disjonction possèdent le même type de propriété, à la différence notable que "ça marche dans les deux sens" (chaque connecteur est distributif par rapport à l'autre) :

— les formules $(A \wedge B) \vee C$ et $(A \vee C) \wedge (B \vee C)$ sont équivalentes ;
— les formules $(A \vee B) \wedge C$ et $(A \wedge C) \vee (B \wedge C)$ sont équivalentes.

3.4.4 Forme normale conjonctive

Les propriétés précédentes (lois de De Morgan, distributivités, élimination de la double négation, transformation de l'implication en disjonction) permettent, de manière systématique, pour n'importe quelle formule de départ A, d'obtenir une formule A', équivalente à A, et qui a la forme suivante, dite *forme normale conjonctive* :

— la formule prend la forme générale d'une conjonction de sous-formules, appelées *clauses* ;
— chaque clause prend la forme générale d'une disjonction de sous-formules, appelées des *littéraux* ;
— chaque littéral est soit une formule atomique réduite à une variable, soit la négation d'une variable.

Les mêmes conditions, en tenant compte des règles de priorité des connecteurs, deviennent, sur les arbres des formules (qui n'utilisent pas le connecter $\to$) :

— un noeud étiqueté $\wedge$ a un fils gauche qui est soit une feuille, soit étiqueté $\vee$ ou $\neg$; et un fils droit qui peut être étiqueté $\wedge$, $\vee$ ou $\neg$, ou être une feuille ;
— un noeud étiqueté $\vee$ a un fils gauche qui est soit une feuille, soit étiqueté $\neg$; et un fils droit qui doit être étiqueté $\vee$ ou $\neg$, ou être une feuille ;
— un noeud étiqueté $\neg$ a un fils qui ne peut qu'être une feuille.

Les formules en forme normale conjonctive ont une grande importance en théorie de la complexité : le problème consistant, à partir d'une formule en forme normale conjonctive, à déterminer s'il existe une valuation qui lui attribue la valeur V, est un exemple emblématique de ce qu'on appelle un problème NP-complet. Il ne sera pas dit un mot de plus à ce sujet dans ce cours.

3.5 Ce qu'il faut retenir de ce chapitre

Ce chapitre n'est pas censé contenir de choses très nouvelles pour un étudiant de Licence, sauf peut-être un peu de vocabulaire. Toute personne qui a eu à écrire des conditions de boucles s'est normalement posé la question de comment reformuler la négation d'une condition.

— **Vocabulaire :** variable propositionnelle, connecteur, formule propositionnelle.

— Notions de valuation et de valeur de vérité, calcul d'une table de vérité qui décrit toutes les valuations pour une formule.

— Réécritures dans une formule ; substitution des variables, remplacement des sous-formules. Théorèmes de substitution uniforme et de remplacement, et leur utilisation pour montrer, sans calcul de tables de vérité, que des formules sont équivalentes.