

Bruno Mery (Doctorant, LaBRI bureau 123, mery@labri.fr)
Université de Bordeaux, année 2007-2008
Informatique Théorique 1
INF154T Groupe CSB5A2

Notes Logique

Ces quelques notes sont destinées à compléter les feuilles d'exercices distribuées en TD en fournissant explications sur les techniques utilisées. Elles sont aussi destinées à pallier une possible annulation des derniers TD de l'année 2007-2008 pour ce groupe et cette UE, en vue de préparer correctement aux examens ; je reste à disposition pour toute question portant sur ces points. Attention : le mécanisme proposé dans le cours pour la logique diffère sensiblement de celui présenté les années précédentes et en deuxième année de Licence (où les règles de déduction sont plus nombreuses, et les axiomes moins) ; il est donc possible que certains exercices provenant d'Annales ou du cours de deuxième année ne conviennent pas à la révision de ce cours en particulier. (Le problème ne se pose pas, ou peu, pour les deux autres parties.)

Ce document ayant été vite rédigé, des erreurs peuvent s'y être glissées. En cas de doute, contactez-moi...

1 Techniques

Suivent quelques brefs rappels sur les techniques utilisées dans les exercices, à maîtriser pour l'examen.

1.1 Manipuler la vérité

La forme peut-être la plus simple de la logique, déjà connue de ceux qui auraient suivi un parcours en électronique, par exemple, est de considérer la logique, ses formules et opérations, comme vériconditionnelles (manipulant des valeurs de vérité assimilables aux booléens 0 ou 1).

La construction de *tables de vérités* pour les formules est un exercice calculatoire facile, pouvant être long selon la taille de la formule considérée. Elle permet de résoudre nombre de problèmes directement : considérons une formule $\phi(a, b, c)$ dont la table est :

a	b	c	ϕ
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Cette table permet immédiatement de répondre aux questions suivantes :

- *La formule ϕ est-elle une tautologie ?* : non (pour certaines valeurs, elle vaut 0). Les tautologies ont 1 pour valeur à chaque ligne.
- *La formule ϕ est-elle satisfaisable ?* : oui (pour certaines valeurs, elle vaut 1). Les formules non satisfaisables sont celles pour lesquelles aucune valeur ne donne 1 comme résultat – autrement dit, chaque ligne est à 0.
- *La formule ϕ est-elle monotone ?* : non. Une fonction¹ booléenne est monotone quand l'augmentation (i.e., le passage de 0 à 1) en valeur d'une quelconque de ses variables ne peut entraîner une diminution (i.e., le passage de 1 à 0), et ici, $\phi(0, 1, 1) = 1$ alors que $\phi(1, 0, 0) = 0$ (considérez la première variable).

¹On confond allègrement formule logique et fonction booléenne quand la valeur de la formule est fonction de variables booléennes, dans ces cas.

1.2 Systèmes de connecteurs complets

Le calcul propositionnel, avec l'ensemble de ses connecteurs (implication, négation, conjonction, disjonction, équivalence), et en logique à deux valeurs, est *complet* : toutes les tautologies sont démontrables à partir des axiomes. Demander si un *système de connecteurs* est complet, dans ce cadre, c'est demander si on peut également démontrer l'ensemble des tautologies à l'aide de ce système. Un argument de non-complétude est de considérer, par exemple :

a	ϕ
0	1
1	1

Ici, ϕ est une tautologie. On ne peut l'écrire à partir de connecteurs tels que \wedge ou \vee seulement : de fait, il est impossible de représenter une fonction telle que $f(0) = 1$ si on ne peut reconstituer la négation (\wedge et \vee sont monotones).

En général, on utilise la complétude du calcul propositionnel : on sait que le système avec tous ses connecteurs est complet. Si on arrive donc à reconstituer les connecteurs manquants à partir d'un ensemble partiel, alors ce dernier était complet. Ainsi, on sait que $\{\rightarrow, \neg\}$ est complet, ainsi que $\{\vee, \neg\}$ ou $\{\wedge, \neg\} : A \rightarrow B \leftrightarrow (B \vee (\neg A \wedge \neg B))$, etc.

1.3 Formes normales

Les *formes normales* de fonctions booléennes sont des manières standardisées de représenter les fonctions, qui peuvent servir pour raisonner sur les preuves de façon abstraite, etc. Toute fonction booléenne peut être représentée sous forme normale à partir de sa table de vérité. On distingue *forme normale disjonctive* (DNF) et *forme normale conjonctive* (CNF), qui sont définies comme suit :

- Les objets de base sont appelés *littéraux* : il s'agit soit de variables atomiques, soit de leur négation (par exemple a ou \neg).
- Il n'y a pas d'autres négations que celles qui accompagnent les littéraux (autrement dit, pas de négations devant des parenthèses).
- Les seuls autres connecteurs présents sont la conjonction (\wedge) et la disjonction (\vee).
- Une formule sous *forme normale conjonctive* est constituée de *conjonctions de clauses disjonctives* : $\phi = E_1 \wedge E_2 \wedge \dots$, avec les E_i (clauses disjonctives) qui sont des disjonctions de littéraux, e.g. $(a_1 \vee \neg a_2 \vee a_3)$.

- Symétriquement, une formule sous *forme normale disjonctive* est constituée de *disjonctions de clauses conjonctives* : $\psi : F_1 \vee F_2 \dots$, avec les F_i (clauses conjonctives) qui sont des conjonctions de littéraux, e.g. $(b_1 \wedge b_2 \wedge \neg b_3)$.

Il existe des méthodes de construction simples et mécaniques pour ces formes normales. Soit une fonction donnée par sa table de vérité : elle est vraie pour toutes les combinaisons de variables qui lui donnent la valeur 1, on sélectionne donc les lignes l_1, l_2, \dots qui donnent 1 en résultat (s'il n'y en a pas, la formule est non-satisfaisable et peut être réduite à la constante 0). Chaque ligne donne une clause conjonctive de la forme normale disjonctive, de la manière suivante : le littéral correspondant à chaque variable est *positif* si la colonne est 1, *négatif* (avec une négation) quand la colonne est 0.

Pour obtenir la forme normale conjonctive, on calcule la forme normale disjonctive de la négation de la fonction de cette façon (en utilisant les valeurs 0 dans la table de vérité). On ajoute ensuite une négation à la formule obtenue ; en utilisant les lois de De Morgan ($\neg(A \vee B) = \neg A \wedge \neg B$), on propage cette négation jusqu'à obtenir la forme normale conjonctive.

Exemple : soit la fonction ϕ suivante (déjà vue) :

a	b	c	ϕ
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

La forme normale disjonctive de cette formule est $(\neg a \wedge b \wedge \neg c) \vee (\neg a \wedge \neg b \wedge c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge b \wedge c)$ (on pourrait simplifier et dire simplement que $\phi = b$, au vu du résultat, mais ce n'est pas le sujet).

La forme normale disjonctive de $\neg\phi$, elle, est $(\neg a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge \neg b \wedge c) \vee (a \wedge \neg b \wedge \neg c) \vee (a \wedge \neg b \wedge c)$. On reste en logique à deux valeurs, donc $\phi = \neg\neg\phi$, soit :

$$\begin{aligned} \phi &= \neg((\neg a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge \neg b \wedge c) \vee (a \wedge \neg b \wedge \neg c) \vee (a \wedge \neg b \wedge c)) \\ \phi &= (\neg(\neg a \wedge \neg b \wedge \neg c) \wedge \neg(\neg a \wedge \neg b \wedge c) \wedge \neg(a \wedge \neg b \wedge \neg c) \wedge \neg(a \wedge \neg b \wedge c)) \\ \phi &= (a \vee b \vee c) \wedge (a \vee b \vee \neg c) \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \end{aligned}$$

1.4 Formules paramétrées

Au lieu de variables, on peut avoir des *assertions paramétrées* comme atomes dans une formule : ainsi, $(n = 1)$ est une assertion vraie ou fausse selon la valeur de n . Pour traiter ces cas, on considère les assertions comme des variables pour la manipulation logique (= n'est pas plus un connecteur que n n'est un booléen, la logique n'a donc rien à en dire), on établit la table de vérité, etc. Ceci posé, on considère ensuite ce que signifie le fait qu'une assertion soit vraie : cela donne un domaine sur lequel il est possible d'agir. Par exemple, $(n = 1)$ (dans \mathbb{N}) vaut 1 (dans \mathbb{B}) si et seulement si $n = 1$, 0 si et seulement si $n \neq 1$.

1.5 BDD

Les diagrammes de décisions binaires (BDD) sont une manière graphique de représenter les formules logiques et d'en faire des arbres de décision, donnant ainsi une correspondance intéressante entre arbres, automates et formules logiques. La construction est simple : à chaque niveau correspond un branchement binaire pour une des variables de la fonction considérée, selon que l'on souhaite que cette variable vaille 0 (fils gauche) ou 1 (fils droit).

En cours, on a pu constater que des simplifications sont possibles : par exemple, quand on a une fonction $x \wedge (y \vee \neg(z \vee (t \rightarrow (x \wedge z))))$, il est inutile de détailler le comportement de la fonction à partir du moment où on a déterminé que $x = 0$: autrement dit, sur le BDD et si on prend x pour premier branchement, on peut se contenter d'étiqueter le fils gauche de la racine par 0, ce qui réduit la taille de l'arbre (mais non sa hauteur) : ce principe (qui peut être atteint en minimisant l'automate associé, mais c'est déconseillé) est *l'élagage d'arbres de décisions*. Cela pose le problème de l'ordre dans lequel on considère les variables : ici, si x avait été considéré en dernier, cette simplification n'aurait servi à rien.

1.6 Logique à trois valeurs

Le principe de la logique à trois valeurs est avant tout pratique : certains axiomes de la logique à deux valeurs ("classique") ne correspondent pas à des preuves intuitivement valides. Par exemple, $A \vee \neg A$ est vrai, même si A ne correspond à rien. La logique à trois valeurs est une réponse philosophiquement intéressante, empruntant au constructivisme, et refusant le principe du tiers exclu...

Pour ce qui nous préoccupe (quelques exercices simples), la différence principale est cette troisième valeur. En effet, on a d'une part A (la formule est vraie), $\neg A$ (la formule est fausse) et $\neg\neg A$ (la formule *n'est pas fausse* – autrement dit, possible). Les tables de vérité sont données dans le cours, et sont assez intuitives ($\neg\neg\neg A = \neg A$: une formule pas pas fausse est fausse). Deux différences dans la démarche : les tables de vérité comportent 3^n lignes pour n variables, et le calcul propositionnel n'est plus complet. Autrement dit, on ne demandera en général pas de trouver des systèmes de connecteurs complets, ni de calculer forme normale conjonctive ou disjonctive de formules, ni tout calcul sur des fonctions de 4 variables ou plus ; la plupart des exercices sur la logique à trois valeurs se concentreront sur le calcul des tables de vérité.

2 Notes sur les exercices

À consulter avec les feuilles d'exercices données.

2.1 Exercice 1

Immédiat : construire la table de vérité, vérifier que toutes les valeurs soient 1. La symétrique est $p \rightarrow (p \vee q)$: ce sont des instances des axiomes 3 et 6, respectivement.

2.2 Exercice 2

La question porte, en fait, sur l'interprétation des formules dans un modèle ensembliste. Sans conceptualiser trop, on cherche simplement des opérations sur les ensembles ayant le même comportement.

2.3 Exercice 3

Application de De Morgan, avec une petite subtilité : pourquoi $\phi(\neg p_1, \dots)$ reste-t-elle sensiblement équivalente à $\phi(p_1, \dots)$? Considérons la table de vérité d'une fonction arbitraire, échangeons 1 et 0, remettons dans l'ordre : on obtiendra la même table qu'au départ. En effet, si on considère qu'une variable peut avoir les valeurs 0 et 1, alors sa négation pourra avoir les valeurs 1 et 0. Autrement dit : $f(a) = f(\neg a)$, tant que la valeur de a est quelconque...

2.4 Exercice 4

Méthode de la construction de la forme normale conjonctive, donnée plus haut.

2.5 Exercices 5, 6, 7, 8

Simple application des méthodes données pour déterminer si un système de connecteurs est complet ou non. Les arguments donnés en solution sont amplement suffisants.

2.6 Exercice 9

Cette démonstration difficile part en fait de la définition de la monotonie : *une fonction est monotone si l'augmentation de valeur d'une de ses variables ne peut pas diminuer la valeur de la fonction*. Isolons donc cette variable et raisonnons sur elle seule, en supposant que le reste se comporte de façon attendue, vérifions que l'on puisse écrire la fonction en distinguant simplement cette variable sans négation, conclure. La difficulté du raisonnement est peut-être le "si ça marche pour ce cas très particulier, ça marche toujours" qui fait le grand charme des démonstrations par induction abrégées (très communes en Master). Ici, l'ordre est "la fonction est fonction de n variables" – donc une extension linéaire de l'ordre naturel sur \mathbb{N} , et donc bien fondé...

2.7 Exercices 10 et 11

Applications numériques. Il y a autant de fonctions de n variables que ce que l'on peut écrire comme combinaisons de valeurs sur une table de vérité de n variables. Or, cette table fait 2^n lignes. Il s'agit donc du nombre de mots sur l'alphabet $\{0, 1\}$ que l'on puisse écrire, de longueur 2^n : il y a donc 2^{2^n} fonctions de n variables (soit une complexité hyperexponentielle). Dans le même ordre d'idée : il y a 2^n valeurs à calculer pour une fonction de n variables, et $n2^n$ littéraux à stocker pour une forme normale disjonctive (au plus 2^n clauses de n littéraux chacun). Le reste est une histoire d'approximations calculatoires.

2.8 Exercice 12

La preuve de réductibilité de SAT à 3-SAT est un des principes fondamentaux de la théorie de la complexité, vue en général en Master ; il est donc normal de ne pas trouver la solution à cet exercice. On peut par exemple consulter <http://www.liafa.jussieu.fr/~carton/Enseignement/Complexite/MasterInfo/Cours/time.html>, mais la preuve reste délicate.

2.9 Exercice 13

Fonctions paramétrées : on considère simplement $a \rightarrow b$, avec $a = (n = 1)$, $b = (n = 2)$. C'est vrai quand $a = 0$ et $b = 0$, $a = 0$ et $b = 1$, $a = 1$ et $b = 1$. Donc $n \in \{\mathbb{N} - 1 \cap \mathbb{N} - 2\} \cup \{\mathbb{N} - 1 \cap \{2\}\} \cup \{\{1\} \cap \{2\}\}$, soit $n \in \{\mathbb{N} - 1\}$. On procède de même pour l'équivalence (vraie quand n est différent de 1 et de 2).

2.10 Exercice 14

On peut largement se fonder sur des tables de vérité...

1. Oui.
2. Non : supposons $G = 0$ (la constante "faux"), $F \rightarrow G$ est satisfaite quelle que soit F .
3. **Oui** (rectification du 14 Décembre) : si F est une tautologie, les deux premières lignes de la table de vérité ne sont plus accessibles ; si $F \rightarrow G$ est satisfaisable, on ne peut exclure la dernière ligne, où G est vraie.
4. Oui (même raisonnement).
5. Oui : si $\neg(G \wedge H)$ est satisfaisable, alors G ou H peuvent être fausses ; si c'est G , comme $F \rightarrow G$ est vraie, F sera alors fausse ; sinon, H est fausse (ou les deux), autrement dit, $\neg F \vee \neg G$ est satisfaisable.

2.11 Exercice 15

Voir le cours pour la construction des BDD : simple application. En réalisant une construction avec simplification, on se rend compte qu'un des deux BDD est beaucoup plus simple que l'autre...

2.12 Exercice 16

Fait en cours : il s'agit de vérifier que le calcul propositionnel, avec seulement deux règles, est très complexe à manipuler. . .

2.13 Exercices 17, 20, 21, 24

Logique à trois valeurs : dans tous les cas, la technique est la même. . . tables de vérité. On vérifie d'abord que $A \rightarrow B$ n'a pas la même table que $\neg A \vee B$, puis que celle de $\neg A \rightarrow (A \rightarrow B)$ ne comporte que des 1, qu'il n'y a que des 1 dans la colonne de B quand $A \rightarrow B$ est vraie, que la table de $A \vee \neg A$ ne comporte pas que des 1. Enfin, on calcule les valeurs possibles pour les trois dernières formules de l'exercice 24 (les deux premières sont clairement fausses, sinon on reviendrait au tiers exclu). Il faut tout de même une table de neuf lignes à chaque essai. . . Les trois formules sont bien des 3-tautologies : la contraposée et les loi de De Morgan sont donc toujours valides en logique à trois valeurs.