

MA122 - Introduction à la logique

Frédéric Herbreteau

`frederic.herbreteau@bordeaux-inp.fr`

18 septembre 2024

1. Connecteurs booléens

Le langage naturel est ambigu

- ▶ “fromage ou dessert”

→ Peut-on avoir fromage **et** dessert ?

- ▶ “si vous pouvez résoudre n'importe quel exercice en TD, alors vous aurez 20/20”

→ Peut-on avoir 20/20 en résolvant **un** exercice ? Ou faut-il **tous** les résoudre ?

- ▶ “la fonction `int find(int t[], int n, int x)` retourne l'indice de `x` dans le tableau `t` à `n` valeurs ou `-1`”

→ La fonction ci-dessous est-elle correcte ?

```
int find(int t[], int n, int x) { return -1; }
```

La logique comme langage formel

- ▶ L'ambiguïté du langage est importante pour la vie quotidienne
- ▶ Mais en **mathématiques** et en **informatique** nous avons besoin d'un langage **non-ambigü**

Logique = mathématiques du raisonnement

- ▶ **Besoin** : **exprimer** et **identifier** des **relations logiques** entre propositions

Exemple

- ▶ **si** “les humains sont mortels” **et** “les Grecs sont humains” **alors** “les Grecs sont mortels”
- ▶ **si** “x apparaît dans t” **alors** “la fonction find retourne son indice”, **et si** “x n'apparaît pas dans t” **alors** “elle retourne -1”

Variables propositionnelles

- ▶ En langage naturel, on combine les propositions avec des mots tels que : **et**, **ou**, etc.

Exemple

si “x apparaît dans t” **alors** “la fonction `find` retourne son indice”
et si “x n’apparaît pas dans t” **alors** “elle retourne -1 ”

- ▶ En logique, on s’intéresse aux **relations logiques** entre les propositions élémentaires qui sont abstraites par des **variables**

Définition

Une **variable propositionnelle** (ou **booléenne**) est une variable à valeurs : 0 (faux) ou 1 (vrai)

Exemple

(si A alors B) et (si non(A) alors C)

Négation (**non**)

Définition

Pour toute proposition A , **non**(A) est également une proposition

Exemple

“ x n’apparaît pas dans t ” $\rightarrow$ **non**(“ x apparaît dans t ”)

Table de vérité :

A	non (A)
0	
1	

“**non**(A)” est **vraie** si ...

Conjonction (**et**)

Définition

Pour toutes propositions A et B , " A **et** B " est également une proposition

Exemple

"les humains sont mortels" **et** "les Grecs sont humains"

Table de vérité :

A	B	A et B
0	0	
0	1	
1	0	
1	1	

" A **et** B " est **vraie** si ...

Disjonction (**ou**)

Définition

Pour toutes propositions A et B , “ A **ou** B ” est également une proposition

Exemple

“la fonction `int find(int t[], int n, int x)` retourne l’indice de x dans le tableau t à n valeurs” **ou** “elle retourne -1”

Table de vérité :

A	B	A ou B
0	0	
0	1	
1	0	
1	1	

“ A **ou** B ” est vraie si ...

Implication (**si ... alors ...**)

Définition

Pour toutes propositions A et B , “**si A alors B** ” est également une proposition

Exemple

si “ x apparaît dans t ” **alors** “la fonction `find` retourne son indice”

Table de vérité :

A	B	si A alors B
0	0	
0	1	
1	0	
1	1	

“**si A alors B** ” est vraie si ...

Ces propositions sont-elles vraies ou fausses ?

- ▶ **si** “la conjecture de Goldbach¹ est vraie” **alors** “ $x^2 \geq 0$ pour tout nombre réel x ”
- ▶ **si** “les cochons volent” **alors** “vous savez prouver la conjecture de Goldbach”
- ▶ **si** “la lune brille” **alors** “la lune est en gruyère”

1. On ne sait pas si la conjecture de Goldbach est vraie ou fausse

Pourquoi “**si A alors B**” est-elle vraie quand A est fausse ?

- ▶ Parce que “**si A alors B**” exprime une condition sur B **uniquement lorsque A est vraie**
- ▶ Cette sémantique est nécessaire pour écrire des **spécifications**

Exemple

La fonction `int find(int t[], int n, int x)` est **correcte** si la proposition suivante est **vraie dans tous les cas**.

(**si** “x apparaît dans t” **alors** “la fonction `find` retourne son indice”) **et** (**si** “x n’apparaît pas dans t” **alors** “elle retourne `-1`”)

- ▶ Supposons que x apparaisse dans t :
- ▶ Supposons que x n’apparaisse pas dans t :

Équivalence (**si et seulement si**)

Définition

Pour toutes propositions A et B , " A si et seulement si B " est également une propositions

Exemple

"il pleut" si et seulement si "il y a des nuages"

Table de vérité :

A	B	A si et seulement si B
0	0	
0	1	
1	0	
1	1	

" A si et seulement si B " est vraie si ...

Logique et langages de programmation

Les propositions et connecteurs booléens sont une part essentielles des **programmes informatiques**

Exemple

- Langages C, C++, Java :

```
if (x > 0 || (x <= 0 && y > 100))  
    ...
```

- Langage Python :

```
if x > 0 or (x <= 0 and y > 0):  
    ...
```

- En posant A la proposition " $x > 0$ " et B la proposition " $y > 0$ " :

$A \text{ or } (\text{non}(A) \text{ and } B)$

Logique propositionnelle

Définition

Une **formule (ou proposition)** est

- ▶ soit une **variable propositionnelle** (proposition élémentaire)
- ▶ soit la **combinaison** de deux formules A et B : “**non**(A)”, “ A et B ”, “ A ou B ”, “**si** A **alors** B ”, “ A **si et seulement si** B ”

Exemple

(**si** “ x apparaît dans t ” **alors** “la fonction `find` retourne son indice”) **et** (**si** “ x n’apparaît pas dans t ” **alors** “elle retourne -1 ”)

Les connecteurs booléens

Textuelle	Abrégée	Mathématique
non (A)		$\neg A$
A et B		$A \wedge B$
A ou B		$A \vee B$
si A alors B	A implique B	$A \implies B$
A si et seulement si B	A ssi B	$A \iff B$

Priorité : du plus fort (gauche) au moins fort (droite)

non **et** **ou** **implique** **ssi**

Exemple

Parenthéser la formule en respectant la priorité des connecteurs :

A **implique** B **et** $\text{non}(A)$ **implique** C

2. Validité

Valeur d'une formule

Définition

Une **valuation** donne une valeur (0 ou 1) à chaque variable propositionnelle

Exemple

$$v(A) = 0, v(B) = 1, v(C) = 1$$

Définition

La **valeur** d'une formule dans une valuation v est obtenue en appliquant les tables de vérité aux valeurs des variables propositionnelles

Exemple

Valeur de : “(A implique B) et (non(A) implique C)” dans la valuation v ci-dessus.

Table de vérité d'une formule

La **table de vérité** consiste à calculer la valeur d'une formule pour toutes les valuations.

Exemple

Pour la formule " A ou ($\text{non}(A)$ et B)"

A	B	A ou ($\text{non}(A)$ et B)
0	0	
0	1	
1	0	
1	1	

Exemple

Combien de valuations avec 2 variables ? 3 variables ? n variables ?

Validité

Définition

Une formule est **valide** si elle est vraie (elle vaut 1) pour toute valuation

Pour prouver la validité :

- ▶ Méthode #1 : table de vérité
- ▶ Méthode #2 : preuve sémantique

Exemple

À l'aide des tables de vérité, indiquer si les formules ci-dessous sont valides

- ▶ $A \text{ ou } (\text{non}(A) \text{ et } B)$
- ▶ $A \text{ implique } (B \text{ implique } A)$

Méthode #2 : preuve sémantique

Pour montrer qu'une formule est valide :

- ▶ on fixe une valuation v quelconque
- ▶ par disjonction de cas (le plus souvent), on montre que la formule est vraie pour v

Exemple

Montrer que la formule " A implique (B implique A)" est valide.

Supposons : v une valuation quelconque.

Conclusion : la formule est vraie pour toute valuation v

3. Équivalence logique

Équivalence logique

Définition

Deux formules A et B sont **logiquement équivalentes**, noté $A \equiv B$, si $v(A) = v(B)$ pour toute valuation v .

Exemple

- ▶ “ A implique B ” est logiquement équivalente à “**non**(A) ou B ”
- ▶ “**non**(A et B)” est logiquement équivalente à “**non**(A) ou **non**(B)”
- ▶ “ A ou (**non**(A) et B)” est logiquement équivalente à “ A ou B ”

Montrer l'équivalence logique :

- ▶ Méthode #1 : comparer les tables de vérité
- ▶ Méthode #2 : preuve sémantique

Méthode #1 : comparaison des tables de vérité

Pour montrer que deux formules sont logiquement équivalente :

- ▶ Calculer les tables de vérité des deux formules
- ▶ Vérifier qu'elles coïncident pour chaque valuation

Exemple

A	B	$A \text{ ou } (\text{non}(A) \text{ et } B)$	$A \text{ ou } B$
0	0		
0	1		
1	0		
1	1		

Méthode #2 : preuve sémantique

Pour montrer que deux formules F et G sont logiquement équivalentes :

- ▶ montrer la double implication : “si F est vraie alors G est vraie” d’une part, et “si G est vraie alors F est vraie d’autre part”
- ▶ pour montrer une implication “si F est vraie alors G est vraie”, on prend une valuation v telle que F est vraie, puis on montre que G est vraie pour v

Exemple

Montrer que “ A ou (**non**(A) et B)” est logiquement équivalente à “ A ou B ”

- ▶ on prend une valuation v telle que “ A ou (**non**(A) et B)” est vraie
- ▶ on prend une valuation v telle que “ A ou B ” est vraie

4. Algèbre et formes normales

Forme normale disjonctive

Définition

Une formule est en **forme normale disjonctive (FND)** si elle est une disjonction (**ou**) de conjonctions (**et**) de littéraux (variables propositionnelles ou leurs négations)

Exemple

$(A \text{ et } B \text{ et } C) \text{ ou } (A \text{ et } B \text{ et non}(C)) \text{ ou } (A \text{ et non}(B) \text{ et } C)$

Théorème

*Toute formule est logiquement équivalente à une formule en **forme normale disjonctive***

Exemple

- ▶ obtenir une FND de la formule " $A \text{ et } (B \text{ ou } C)$ " depuis sa table de vérité
- ▶ en déduire une technique générale

Forme normale conjonctive

Définition

Une formule est en **forme normale conjonctive (FNC)** si elle est une conjonction (**et**) de disjonctions (**ou**) de littéraux (variables propositionnelles ou leurs négations)

Exemple

$(\text{non}(A) \text{ ou } B \text{ ou } C) \text{ et } (A \text{ ou } \text{non}(B) \text{ ou } \text{non}(C)) \text{ et } (A \text{ ou } \text{non}(B) \text{ ou } C) \text{ et } (A \text{ ou } B \text{ ou } \text{non}(C)) \text{ et } (A \text{ ou } B \text{ ou } C)$

Théorème

*Toute formule est logiquement équivalente à une formule en **forme normale conjonctive***

Exemple

- ▶ obtenir une FNC de la formule " $A \text{ et } (B \text{ ou } C)$ " depuis sa table de vérité
- ▶ en déduire une technique générale

Propriétés des connecteurs (1/2)

Simplification :

- ▶ “**non(non(*A*))**” est équivalente à “*A*”
- ▶ “*A* **et** *A*” et “*A* **et** *true*” sont équivalentes à *A*
- ▶ “*A* **et** *false*” et “*A* **et** **non(*A*)**” sont équivalentes à “*false*”
- ▶ “*A* **ou** *A*” et “*A* **ou** *false*” sont équivalentes à “*A*”
- ▶ “*A* **ou** *true*” et “*A* **ou** **non(*A*)**” sont équivalentes à “*true*”

Associativité :

- ▶ “*A* **et** (*B* **et** *C*)” est équivalente à “(*A* **et** *B*) **et** *C*”
- ▶ “*A* **ou** (*B* **ou** *C*)” est équivalente à “(*A* **ou** *B*) **ou** *C*”
- ▶ “*A* **implique** *B* **implique** *C*” se lit “*A* **implique** (*B* **implique** *C*)”

Exemple

Calculer les tables de vérité de :

- ▶ “*A* **implique** (*B* **implique** *C*)”
- ▶ “(*A* **implique** *B*) **implique** *C*”

Propriétés des connecteurs (2/2)

Commutativité :

- ▶ “A et B” est équivalente à “B et A”
- ▶ “A ou B” est équivalente à “B ou A”

Distributivité :

- ▶ “A et (B ou C)” est équivalente à “(A et B) ou (A et C)”
- ▶ “A ou (B et C)” est équivalente à “(A ou B) et (A ou C)”

Lois de De Morgan :

- ▶ “non(A et B)” est équivalente à “non(A) ou non(B)”
- ▶ “non(A ou B)” est équivalente à “non(A) et non(B)”

Exemple

Calculer une FND et une FNC de “A et (B ou C)” à l’aide des propriétés précédentes. Comparer aux formes obtenues depuis la table de vérité.

5. Le problème SAT

Satisfiabilité

Définition

Une formule est **satisfiable** s'il existe une valuation pour laquelle elle est vraie

Sinon la formule est **insatisfiable**.

Exemple

La formule " A ou ($\text{non}(A)$ et B)" est satisfiable car ...

Théorème

Une formule F est valide si et seulement si sa négation " $\text{non}(F)$ " est insatisfiable

Exemple

La formule " A ou ($\text{non}(A)$ et B)" n'est pas valide car ...

Complexité du problème SAT et algorithme

Problème SAT :

ENTRÉE : une formule propositionnelle F

QUESTION : existe-t-il une valuation v telle que $v(F) = 1$?

- ▶ De nombreux problèmes algorithmique **se réduisent à SAT** (Sudoku, coloration de graphes, simplification de circuits, vérification de programmes, ...)
- ▶ Le **calcul de la table de vérité** de F permet de déterminer si F est satisfiable, en temps ...

Exemple

Étudier la satisfiabilité de “ A ou (non(A) et B)” depuis sa table de vérité, depuis une FNC et depuis une FND calculées précédemment.

Théorème

Le problème SAT est NP-complet

6. Logique des prédicats

Prédicats

Définition

Un **prédicat** est une proposition dont la véracité dépend d'une ou plusieurs variables

Exemple

- ▶ “ $5x^2 - 7 = 0$ ” est vrai **pour certaines** valeurs : $x = \pm\sqrt{\frac{7}{5}}$
- ▶ “ $x^2 \geq 0$ ” est vrai **pour toute** valeur de $x \in \mathbb{R}$
- ▶ Nous avons besoin de **quantifier** les valeurs des variables pour lesquelles un prédicat est vrai

Exemple

“si vous pouvez résoudre **n'importe quel** exercice de TD, alors vous aurez 20/20”

- ▶ “si vous pouvez résoudre **au moins un** exercice ...” ?
- ▶ “si vous pouvez résoudre **tous les** exercices ...” ?

Quantificateurs

Définition

Si F est une formule (et D un domaine) :

Universel : “**pour tout** $x \in D$, F ” est une formule

Existentiel : “**il existe** $x \in D$, F ” est une formule

- ▶ Notations mathématiques : $\forall x \in D$ (**pour tout**) et $\exists x \in D$ (**il existe**)

Exemple

- ▶ “**pour tout** $x \in \mathbb{R}$, $x^2 \geq 0$ ” est une formule vraie
- ▶ “**il existe** $x \in \mathbb{R}$, $x^2 \geq 0$ ” est également une formule vraie
- ▶ “**il existe** $x \in \mathbb{R}$, $5x^2 - 7 = 0$ ” est une formule vraie
- ▶ “**pour tout** $x \in \mathbb{R}$, $5x^2 - 7 = 0$ ” est une formule fausse

NB : lorsque toutes les variables sont quantifiées (formule close), on obtient une proposition

Combiner les quantificateurs

- ▶ Il est souvent nécessaire de **combiner plusieurs quantificateurs**

Exemple (Conjecture de Goldbach)

- ▶ “tout entier pair $n > 2$ est la somme de deux nombres premiers”
 - ▶ “**pour tout** $n \in \text{Pairs}$, (**si** $n > 2$ **alors il existe** $p, q \in \text{Premiers}$, $n = p + q$)”
 - ▶ “ $\forall n \in \text{Pairs}, (n > 2 \implies \exists p, q \in \text{Premiers}, n = p + q)$ ”
-
- ▶ L'ordre des quantificateurs est significatif

Exemple

La formule ci-dessous est-elle équivalente aux formules ci-dessus ?

“ $\exists p, q \in \text{Premiers}, \forall n \in \text{Pairs}, (n > 2 \implies n = p + q)$ ”

Propriétés des quantificateurs (1/2)

Priorité : de la plus haute (gauche) à la plus basse (droite)

pour tout/**il existe** **non** **et** **ou** **implique** **ssi**

Exemple

Parenthéser la formule en respectant la priorité des connecteurs :

“ $\forall n \in \text{Pairs}, n > 2 \implies \exists p, q \in \text{Premiers}, n = p + q$ ”

Domaine vide :

- ▶ “**il existe** $x \in \emptyset, F$ ” est fausse, peu importe F
- ▶ “**pour tout** $x \in \emptyset, F$ ” est vraie, indépendamment de F

Propriétés des quantificateurs (2/2)

Négation :

- ▶ “**non**($\forall x, F$)” est équivalent à “ $\exists x, \text{non}(F)$ ”
- ▶ “**non**($\exists x, F$)” est équivalent à “ $\forall x, \text{non}(F)$ ”

Commutativité :

- ▶ “ $\forall x, \forall y, F$ ” est équivalent à “ $\forall y, \forall x, F$ ”
- ▶ “ $\exists x, \exists y, F$ ” est équivalent à “ $\exists y, \exists x, F$ ”

Distributivité :

- ▶ $\exists x, (F \text{ et } G)$ implique $(\exists x, F) \text{ et } (\exists x, G)$
- ▶ $\forall x, (F \text{ et } G)$ est équivalent à $(\forall x, F) \text{ et } (\forall x, G)$
- ▶ $\exists x, (F \text{ ou } G)$ est équivalente à $(\exists x, F) \text{ ou } (\exists x, G)$
- ▶ $(\forall x, F) \text{ ou } (\forall x, G)$ implique $\forall x, (F \text{ ou } G)$

Validité et satisfiabilité

Définition

Une formule avec prédicats est **valide** si elle est vraie pour toute valeur des variables et toute interprétation des prédicats.

Une formule avec prédicats est **satisfiable** si elle est vraie pour (au moins) une valeur des variables et une interprétation des prédicats

Exemple

- ▶ “ $\exists x, \forall y, P(x, y)$ implique $\forall y, \exists x, P(x, y)$ ”
est valide : elle est vraie **pour toute interprétation** du prédicat P .
- ▶ “ $\forall x, \exists y, P(x, y)$ implique $\exists x, \forall y, P(x, y)$ ”
est satisfiable : **il existe une interprétation** du prédicat P pour laquelle elle est vraie

Théorème

La validité et la satisfiabilité des formules avec prédicats sont indécidables.

Techniques de preuves avec quantificateurs

Méthodologie : “**pour tout** x , F ”

- ▶ on montre que F est vraie pour une **valeur quelconque** de x
(un paramètre de la preuve)

Méthodologie : “**il existe** x , F ”

- ▶ on montre que F est vraie pour une **valeur choisie** de x

Exemple

“ $\forall x, P(x)$ implique $\exists x, P(x)$ ”

Supposons : $\forall x, P(x)$ est vraie.

Conclusion : $\exists x, P(x)$ est vraie.

7. Validité des déductions logiques

Prouver les règles de déduction logique

La logique permet de **formaliser** et de **prouver** les règles de déduction logique

Exemple

$$\text{► } \frac{A \quad A \text{ implique } B}{B}$$

$$\text{► } \frac{A \text{ implique } B}{\text{non}(B) \text{ implique } \text{non}(A)}$$