# Logique propositionnelle #2

Pierre Karpman

Lycée Champollion MP2I

https://membres-ljk.imag.fr/Pierre.Karpman/CPGE/2025/

# Table des matières

# Formes normales : quoi & pourquoi ?

Une forme normale est une façon « standardisée » de représenter un objet

## Forme normale en logique propositionnelle

Informellement, une forme normale est un sous-ensemble $\mathcal{F}$ des formules t.q. pour toute formule $\varphi$, il existe $\varphi' \in \mathcal{F}$, $\varphi' \equiv \varphi$

- Notion faisant intervenir la sémantique

## Intérêt

- On peut spécialiser des outils, des raisonnements pour une forme normale sans (trop de) perte de généralité…
- Permet parfois de caractériser certains aspects des objets (par ex. dans notre cas, le *degré* d'une formule)

# Forme normale conjonctive, disjonctive

## Littéral, clauses

- Un *littéral* est une formule $v$ ou $\neg v$, avec $v \in \mathcal{V}$ une variable
- Une *clause conjonctive* est une conjonction $\bigwedge_i \ell_i$ de littéraux
  - Aussi appelée *min-terme* si toutes les variables sont distinctes
- Une *clause disjonctive* est une disjonction $\bigvee_i \ell_i$ de littéraux
  - Aussi appelée *max-terme* si toutes les variables sont distinctes

## Forme normale conjonctive

Une formule $\varphi$ est sous *forme normale conjonctive* (FNC, ou CNF) ssi. c'est une conjonction de clauses disjonctives : $\varphi = \bigwedge_i (\bigvee_j \ell_{i,j})$

## Forme normale disjonctive

Une formule $\varphi$ est sous *forme normale disjonctive* (FND, ou DNF) ssi. c'est une disjonction de clauses conjonctives : $\varphi = \bigvee_i (\bigwedge_j \ell_{i,j})$

# Mise sous CNF, DNF

## Théorème

Toute formule $\varphi$ est sémantiquement équivalente à (au moins) une formule $\varphi_\wedge$ (resp. $\varphi_\vee$) sous CNF (resp. DNF)

## Preuve 1 (esquisse) : par interpolation de l'évaluation de $\varphi$

On note $V_\varphi = \{\sigma \mid [\![\varphi]\!]_\sigma = V\} = \mathbb{M}(\varphi)$, $F_\varphi = \{\sigma \mid [\![\varphi]\!]_\sigma = F\} = \mathbb{V} \setminus \mathbb{M}(\varphi)$

- DNF (« primale : on vise les $V$ ») :
  - pour toute $\sigma \in V_\varphi$, on peut construire une clause conjonctive $\psi_\sigma$ t.q. $[\![\psi_\sigma]\!]_{\sigma'} = V$ ssi. $\sigma' = \sigma$
  - $\bigvee_{\sigma \in V_\varphi} \psi_\sigma \equiv \varphi$ et est sous DNF
- CNF (« duale : on évite les $F$ ») :
  - pour toute $\sigma \in F_\varphi$, on peut construire une clause disjonctive $\psi_\sigma$ t.q. $[\![\psi_\sigma]\!]_{\sigma'} = F$ ssi. $\sigma' = \sigma$
  - $\bigwedge_{\sigma \in F_\varphi} \psi_\sigma \equiv \varphi$ et est sous CNF

# Mise sous CNF, DNF *bis*

## Remarques

- Les formes normales construites par interpolation sont uniques à ordre (et répétition) près des clauses
  - En l'absence de répétition, on parle de formes normales *complètes* ou *canoniques* : toute variable apparaît exactement une fois dans chaque clause (chaque clause est un min/max-terme faisant intervenir toutes les variables)
- La construction se fait aisément à partir de la table de vérité de la formule
- Décider si une formule sous DNF est satisfiable peut se faire en temps linéaire en sa taille, mais :
- Le coût de construction pire cas (CNF ou DNF) est exponentiel en le nombre de variable
- La *taille* d'une forme normale (conjonctive ou disjonctive) peut être exponentielle en celle de la formule initiale

## Exemple

# Mise sous CNF, DNF *ter*

## Preuve 2 (esquisse) : par *réécriture*

On applique des règles de réécriture $\rho$ qui préservent la sémantique et convergent vers une forme normale. Exemples pour CNF :

- $\rho(\varphi_1 \wedge \varphi_2) \Rightarrow \rho(\varphi_1) \wedge \rho(\varphi_2)$
- $\rho(\varphi_1 \rightarrow \varphi_2) \Rightarrow \rho(\neg\varphi_1 \vee \varphi_2)$
- $\rho(\neg\neg\varphi) \Rightarrow \rho(\varphi)$
- $\rho(\neg(\varphi_1 \wedge \varphi_2)) \Rightarrow \rho(\rho(\neg\varphi_1) \vee \rho(\neg\varphi_2))$
- $\rho(\neg(\varphi_1 \vee \varphi_2)) \Rightarrow \rho(\rho(\neg\varphi_1) \wedge \rho(\neg\varphi_2))$
- $\rho(\varphi_1 \vee \varphi_2) \Rightarrow$ *(utilise la distributivité)*

## Remarques

- L'approche suit la définition d'une forme normale pour un système de réécriture…
- Les règles pour une DNF sont les mêmes *mutatis mutandis*
- La forme normale ainsi obtenue peut être de taille exponentielle en celle de l'entrée, à cause de l'utilisation qui est faite de la distributivité

# CNF par réécriture en OCaml #1

```
let rec cnfy f = match f with
    | V _ -> f
    | And (f1, f2) -> cnfy_fold_and (cnfy f1) (cnfy f2)
    | Or  (f1, f2) -> cnfy_fold_or (cnfy f1) (cnfy f2)
    | Imp (f1, f2) -> cnfy (Or (Neg f1, f2))
    | Neg f' -> (match f' with
                 | V _ -> f
                 | Neg f'' -> cnfy f''
                 | Imp (f1, f2) -> cnfy (And (f1, Neg f2))
                 | And (f1, f2) -> let f1' = cnfy (Neg f1) in
                                   let f2' = cnfy (Neg f2) in
                                   cnfy (Or (f1', f2'))
                 | Or  (f1, f2) -> let f1' = cnfy (Neg f1) in
                                   let f2' = cnfy (Neg f2) in
                                   cnfy (And (f1', f2'))
    )
```

# CNF par réécriture en OCaml #2

```
let rec merge_and cnf1 cnf2 = match cnf1 with
    | V _ | Neg (V _) | Or _ -> And (cnf1, cnf2)
    | And (cnf1head, cnf1tail) -> And (cnf1head, merge_and cnf1tail cnf2)

let cnfy_fold_and = merge_and

let rec merge_or dnf1 dnf2 = match dnf1 with
    | V _ | Neg (V _) | And _ -> Or (dnf1, dnf2)
    | Or (dnf1head, dnf1tail) -> Or (dnf1head, merge_or dnf1tail dnf2)

let rec distor dclause cnf = match cnf with
    | V _ | Neg (V _) -> Or (dclause, cnf)
    | Or _ -> merge_or dclause cnf
    | And ((Or _ as or1), (Or _ as or2)) -> And (merge_or dclause or1,
                                                 merge_or dclause or2)
    | And (fhead, ftail) -> And (merge_or dclause fhead, distor dclause ftail)

let rec cnfy_fold_or cnf1 cnf2 = match cnf1 with
    | V _ | Neg (V _) | Or _ -> distor cnf1 cnf2
    | And (cnf1head, cnf1tail) -> And (distor cnf1head cnf2,
                                       (cnfy_fold_or cnf1tail cnf2))
```

# Démo : explosion des CNFs de $\bigvee_{i=1}^{n}(X_{2i} \wedge X_{2i+1})$

# 3-CNF efficace

## Théorème

Pour toute formule propositionnelle $\varphi$, on peut calculer en temps polynomial en sa taille une formule $\varphi_{3\text{-}CNF}$ de taille linéaire en sa taille telle que :

- $\varphi_{3\text{-}CNF}$ est sous forme *3-CNF*: c'est une conjonction de clauses disjonctives de trois littéraux
- $\varphi_{3\text{-}CNF}$ et $\varphi$ sont *equisatisfiables* (l'une est SAT ssi. l'autre l'est)
- On peut « efficacement » convertir un modèle de $\varphi_{3\text{-}SAT}$ en un modèle de $\varphi$

## Preuve

Par exemple *via* la transformation de Цейтин (admis)

Démo $\bigvee_{i=1}^{n}(X_{2i} \wedge X_{2i+1})$

# 3-CNF → 3-SAT

- De nombreux problèmes algorithmiques se *réduisent* à la résolution de problèmes SAT (*cf.* ci-dessous)
- Par ce qui précède, on peut (si l'on veut) se contenter d'écrire des *SAT-solvers* pour des formules sous 3-CNF, sans perdre radicalement en expressivité
- Mais on ne sait pas efficacement résoudre SAT dans tous les cas, même si certains prétendent que c'est facile

# Bonus : forme normale algébrique 🙈

### Théorème

Toute formule propositionnelle $\varphi$ en $n$ variables a son évaluation uniquement interpolée par un polynôme $P \in \mathbb{F}_2[X_0, \ldots, X_{n-1}]/\langle X_i^2 - X_i \rangle_{0 \leq i < n}$ : sa *forme normale algébrique* (FNA, ou ANF)

### Preuve

Par exemple *via* la transformation $u|u+v$

### Applications

La forme normale algébrique permet par exemple de définir le *degré* d'une formule. De façon générale, elle a de nombreuses applications en théorie des codes et en cryptographie

# Table des matières

# Problème SAT

## Problème SAT : version décision

Soit $\varphi$ une formule de la logique propositionnelle, admet-elle un modèle pour la sémantique par tables de vérité ?

## Problème SAT : version recherche

— ? Si oui, en renvoyer un

## Exercice

Montrez que la version recherche du problème SAT n'est pas significativement plus dure que la version décision

Un peu plus formellement, montrez qu'étant donné un oracle $\mathcal{S}$ (une fonction « boîte-noire ») de résolution pour SAT-décisionnel, on a un algorithme qui résout SAT-recherche pour $\varphi$ à $n$ variables en temps $O(n \times |\varphi|)$ et au plus $n$ appels à $\mathcal{S}$ sur des formules $\varphi'$ avec $|\varphi'| \leq |\varphi|$

# 3-CNF SAT

- On peut « spécialiser » le problème SAT en imposant que l'entrée soit une formule sous une certaine forme
- Par exemple (typiquement) une formule en 3-CNF
- Dans ce cas on ne perd pas significativement en généralité, puisque pour toute formule $\varphi$ il existe une formule $\varphi_{3-CNF}$ en 3-CNF qui lui est équisatisfiable (*cf.* ci-dessus)

# Réduction à SAT

De nombreux problèmes se *réduisent* « efficacement » au problème SAT

## Principe d'une réduction (à SAT-décision)

Soit une instance $x$ d'un problème de décision $\mathcal{P}$, on calcule une formule $\varphi_x$ telle que $\varphi_x$ est satisfiable ssi. $x$ est une instance positive de $\mathcal{P}$ la réponse du problème $\mathcal{P}$ pour $x$ est « vrai »

- Idéalement, on souhaite que les tailles des instances soient comparables ($|\varphi_x| \approx |x|$)
- On peut dire que $\varphi_x$ *modélise* $x$
- Détails : l'année prochaine

# Exemple de modélisation : la $k$-colorabilité d'un graphe

## Problème initial

Soit $G = S, A$ un graphe et $k \in [\![\#S]\!]$, $G$ est-il $k$-coloriable ?

## Principe d'une modélisation

- On définit $\mathcal{V} = \{c_{i,j}\}_{0 \leq i < \#S, 0 \leq j \leq k}$
  - Dans une valuation, $\sigma$, on interprétera $\sigma(c_{i,j}) = 1$ comme voulant dire que le sommet $i$ est colorié avec la couleur $j$
- Pour toute « contrainte » (condition qu'un coloriage valide doit *nécessairement* satisfaire), on construit une formule qui s'évalue à vrai pour une valuation $\sigma$ ssi. le coloriage modélisé par $\sigma$ satisfait la contrainte
- On modélise l'instance par la conjonction des formules modélisant les contraintes

## Quelles contraintes ?

# $k$-colorabilité : fin

## Un sommet est uniquement colorié

- Pour tout $i, j$ : $c_{i,j} \rightarrow \neg \bigvee_{\ell \neq j} c_{i,\ell}$
  - Ou la formule sémantiquement équivalente $\bigwedge_{\ell \neq j} \neg(c_{i,j} \wedge c_{i,\ell})$
- Pour tout $i$ : $\bigvee_j c_{i,j}$

## Deux sommets adjacents sont de couleurs différentes

- Pour toute arête $(v, w) \in A$, pour tout $j$ : $\neg(c_{v,j} \wedge c_{w,j})$

On obtient une formule de taille polynomiale en celle du graphe

# Un algorithme de résolution (inefficace dans le pire cas)

Soit une formule $\varphi$ sur un ensemble de variable $\mathcal{V}$, on peut utiliser un algorithme de *backtracking* évident, avec un élagage glouton (l'algorithme « de Quine ») pour décider si elle est satisfiable :

- on fixe un certain ordre (arbitraire) sur les variables ;
- on définit une valuation partielle $\sigma$ de $\mathcal{V}$, initialement définie nulle part
- pour chaque variable $v$ dans l'ordre, et soit $\varphi'$ la formule courante :
  - on affecte $\sigma(v) = 1$
  - on considère la formule $\varphi''$ obtenue en *simplifiant* la formule $\varphi'[\top/v]$ obtenue en substituant $\top$ à $v$ dans $\varphi'$
  - si l'on peut déterminer que cette formule est insatisfiable, on affecte $\sigma(v) = 0$ et recommence
    - si l'on peut déterminer que $\varphi'[\bot/v]$ est insatisfiable, on *backtrack*
    - sinon on itère en passant à la variable suivante, et *backtrack* si nécessaire
  - sinon on itère en passant à la variable suivante, et considère $\sigma(v) = 0$ si nécessaire

# Simplifications, décision anticipée

- Pour être intéressant, l'algorithme précédent doit pouvoir « simplifier » ou déterminer l'insatisfiabilité de certaines formules
- Ceci peut se faire par des transformations *syntaxiques* qui dépendent de la sémantique (ici par tables de vérité)

## Petit catalogue de règles

On ne donne pas de règles faisant intervenir $\rightarrow$, puisque l'on peut supposer sans perte de généralité une formule n'utilisant pas ce connecteur

- $\top$ est satisfiable
- $\bot$ est insatisfiable
- $\top \vee \varphi, \varphi \vee \top, \neg\bot \Rightarrow \top$
- $\bot \wedge \varphi, \varphi \wedge \bot, \neg\top \Rightarrow \bot$
- $\top \wedge \varphi, \varphi \wedge \top, \bot \vee \varphi, \varphi \vee \bot, \Rightarrow \varphi$

# Table des matières

# Sémantique des ouverts 

## Préliminaires

- On appelle *ouvert* de la droite réelle $\mathbb{R}$ toute union d'intervalles ouverts $]a, b[$, avec $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$
- On note $\mathcal{O}(\mathbb{R})$ l'ensemble de ces ouverts
- On identifie $\mathbb{R}$ à l'ouvert $]-\infty, +\infty[$
- On définit l'*intérieur* $\text{int}(S)$ d'un sous-ensemble $S$ de $\mathbb{R}$ comme l'union de tous les ouverts $X$ de $\mathbb{R}$ inclus dans $S$
  - $\text{int}(S)$ est un ouvert
  - $\text{int}(S) \subseteq S$, avec égalité ssi. $S$ est ouvert

## Exemples

- $]0, 1[$, $]0, 1[ \cup ]1, 2[$, $\varnothing$, $\mathbb{R}$ sont des ouverts
- $[0, 1]$, $\{0\}$ ne sont pas des ouverts
- $\text{int}([0, 1]) = ]0, 1[$, $\text{int}(\{0\}) = \varnothing$

# Sémantique des ouverts 

- Une valuation de la sémantique des ouverts est une fonction $\sigma : \mathcal{V} \to \mathcal{O}(\mathbb{R})$ (en fait pas nécessairement $\mathbb{R}$)
- On définit inductivement l'évaluation $[\![\cdot]\!]_\sigma$ pour cette sémantique relativement à $\sigma$ (ou $[\![\cdot]\!]$ si $\sigma$ est clair dans le contexte ou non important) comme :
  - $[\![\top]\!] = \mathbb{R}$
  - $[\![\bot]\!] = \varnothing$
  - $[\![v]\!]_\sigma = \sigma(v)$ pour toute variable $v \in \mathcal{V}$
  - $[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \cap [\![\varphi_2]\!]$
  - $[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$
  - $[\![\varphi_1 \to \varphi_2]\!] = \mathrm{int}(\overline{[\![\varphi_1]\!]} \cup [\![\varphi_2]\!])$

## Remarques

- On peut comprendre $[\![\varphi_1 \to \varphi_2]\!]$ comme étant le plus grand ouvert $X$ de $\mathbb{R}$ t.q. $([\![\varphi_1]\!] \cap X) \subseteq ([\![\varphi_2]\!] \cap X)$
  - Un ouvert de $[\![\varphi_1 \to \varphi_2]\!]$ est ou bien non inclus dans $[\![\varphi_1]\!]$, ou bien inclus dans $[\![\varphi_1]\!]$ et $[\![\varphi_2]\!]$
- $[\![\neg\varphi]\!]$ est définie *via* $[\![\varphi \to \bot]\!]$ comme $\mathrm{int}(\overline{[\![\varphi]\!]})$

# Sémantique des ouverts 

## Modèles, tautologies

Un modèle de $\varphi$ pour la sémantique des ouverts est une valuation $\sigma$ telle que $[\![\varphi]\!]_\sigma = \mathbb{R}$
Une tautologie est (toujours) une formule dont toutes les valuations (ici *en nombre infini non dénombrable*) sont des modèles

## Utilité

La sémantique des ouverts est *correcte* et *complète* (notions définies l'année prochaine) pour la logique intuitionniste, tandis que la sémantique par tables de vérité est correcte et complète pour la logique classique, mais seulement correcte pour la logique intuitionniste :

- Une formule est un théorème de la logique classique (resp. intuitionniste) ssi. c'est une tautologie pour la sémantique par tables de vérité (resp. des ouverts)
- Si une formule est un théorème de la logique intuitionniste, alors c'est une tautologie pour la sémantique par tables de vérité

On peut (notamment) utiliser la sémantique des ouverts pour montrer qu'une formule (qui est éventuellement un théorème de la logique classique) n'est *pas* un théorème de la logique intuitionniste

# Exercices

Montrez que pour la sémantique des ouverts, on a pour toute formule $\varphi$ :

- $\vDash \neg(\varphi \wedge \neg\varphi)$
- $\varphi \vee \neg\varphi$ n'est pas une tautologie
- $(\neg\varphi \rightarrow \varphi) \rightarrow \varphi$ n'est pas une tautologie

Dans les deux derniers cas, les formules sont des tautologies pour la sémantique par tables de vérité

## $\neg(\varphi \wedge \neg\varphi)$

On considère une valuation arbitraire $\sigma$, et note $X = [\![\varphi]\!]_\sigma$, alors :

- $[\![\varphi \wedge \neg\varphi]\!]_\sigma = X \cap \mathrm{int}(\overline{X}) = \varnothing$
  - puisque $\mathrm{int}(\overline{X}) \subseteq \overline{X}$ d'intersection vide avec $X$
- $[\![\neg(\varphi \wedge \neg\varphi)]\!]_\sigma = \mathrm{int}(\mathbb{R}) = \mathbb{R}$

## $\varphi \vee \neg\varphi$

Soit $\varphi, \sigma$ t.q. $[\![\varphi]\!]_\sigma = ]0,1[$, alors :

- On a:

$$[\![\neg\varphi]\!]_\sigma = \text{int}(]-\infty, 0] \cup [1, +\infty[) = ]-\infty, 0[ \cup ]1, +\infty[$$

- Et :

$$[\![\varphi \vee \neg\varphi]\!]_\sigma = ]0,1[ \cup ]-\infty, 0[ \cup ]1, +\infty[ = \mathbb{R} \setminus \{0,1\} \neq \mathbb{R}$$

# $(\neg\varphi \to \varphi) \to \varphi$

Soit $\varphi, \sigma$ t.q. $[\![\varphi]\!]_\sigma = ]0,1[ \cup ]1,2[$, alors :

- $\overline{[\![\varphi]\!]_\sigma} = ]-\infty, 0] \cup \{1\} \cup [2, +\infty[$
- $[\![\neg\varphi]\!]_\sigma = ]-\infty, 0[ \cup ]2, +\infty[$
- $\overline{[\![\neg\varphi]\!]_\sigma} = [0, 2]$
- $[\![\neg\varphi \to \varphi]\!]_\sigma = \text{int}([0,2] \cup ]0,1[ \cup ]1,2[) = ]0,2[$
- $\overline{[\![\neg\varphi \to \varphi]\!]_\sigma} = ]-\infty, 0] \ \cup \ [2, +\infty[$
- Et alors :

$$[\![(\neg\varphi \to \varphi) \to \varphi]\!]_\sigma = \text{int}(]-\infty, 0] \ \cup \ [2, +\infty[ \cup ]0,1[ \cup ]1,2[)$$

- Soit :

$$[\![(\neg\varphi \to \varphi) \to \varphi]\!]_\sigma = \text{int}(]-\infty, 1[ \cup ]1, +\infty[) = \mathbb{R} \setminus \{1\} \neq \mathbb{R}$$