Séquence 14 - Logique

Table des matières

I. Syntaxe des formules en logique propositionnelle

I. 1. Définition inductive

Définition 1 (Variable propositionnelle ou formule atomique)

Une **variable propositionnelle** (ou **formule atomique**) est une assertion que ne peut prendre que deux états possibles, appelées valeurs de vérité.
Les deux états possibles sont :

- la valeur vraie, notée V, 1 ou `true`
- la valeur fausse, notée F, 0 ou `false`

On notera $\mathcal{P}$ l'ensemble des variables propositionnelles.

Notation 1 (Cas particuliers : $\top$ et $\bot$)

On notera $\top$ la variable propositionnelle particulière, qui prend toujours la valeur V (valeur vraie).
On notera $\bot$ la variable propositionnelle particulière, qui prend toujours la valeur F (valeur fausse)

Attention. Différence entre V et $\top$: V est une valeur booléenne, $\top$ est une variable propositionnelle constante, dont l'image vaut toujours V.

Remarque. Numérotation des variables propositionnelles L'ensemble des variables propositionnelles $\mathcal{P}$ sera la plupart du temps fini (donc dénombrable), c'est-à-dire que :

$$\text{Card}\mathcal{P} = n$$

Ainsi, on pourra parfois être amenés à étiqueter les variables propositionnelles par des entiers dans $[\![1, n]\!]$. On notera alors :

$$\mathcal{V} = \{p_1, \dots p_n\}$$

Une formule logique est formée inductivement à partir de ces éléments de base $\mathcal{P} \cup \{\top, \bot\}$, en les assemblant à l'aide de différents constructeurs appelés connecteurs logiques :

Définition 2 (Formule ou expression logique - Définition inductive)

Soit $\mathcal{P}$ un ensemble de variables propositionnelles. L'ensemble des formules logiques construites à partir de ces variables propositionnelles, noté $\mathcal{F}$, est défini inductivement avec la signature suivante :

- les variables propositionnelles $\mathcal{P}$ et les deux formules logiques constantes $\top$ et $\bot$, qui constituent les éléments de base de cet ensemble (constructeurs d'arité 0)
- un constructeur d'arité 1 noté `not`, qui correspond au NON logique
- trois constructeurs d'arité 2 notés `and`, `or` et `imp` et qui correspondent respectivement au ET logique, au OU logique et à l'opérateur logique d'implication.

Une formule logique $\varphi \in \mathcal{F}$ est un **terme** de cet ensemble inductif.

Définition 3 (Système complet)

On appelle système de complet une signature (et en particulier un ensemble de constructeurs logiques) permettant de générer n'importe quelle formule de $\mathcal{F}$.

Remarque. Ici, par exemple, le constructeur `imp` est surnuméraire : on pourrait l'enlever de la signature et toujours avoir un système de complet.

I. 2. Représentation de la syntaxe sous forme d'arbre

Comme nous l'avons vu pour d'autres ensembles construits de manière inductive, on peut représenter toute formule logique sous la forme d'un arbre :

— les éléments de base (variables propositionnelles et éventuellement formules constantes $\bot$ et $\top$) constituent les feuilles de l'arbre.

— les différents constructeurs apparaissent comme les nœuds internes de l'arbre

Définition 4 (Taille d'une formule logique)

La taille d'une formule logique φ, notée $|\varphi|$ est définie de manière inductive de la façon suivante :

— $|\top| = 0$

— $|\bot| = 0$

— $\forall p \in \mathcal{P}$, $|p| = 0$

— $|\texttt{not}(\varphi)| = 1 + |\varphi|$

— $|\texttt{and}(\varphi_1, \varphi_2)| = 1 + |\varphi_1| + |\varphi_2|$

— $|\texttt{or}(\varphi_1, \varphi_2)| = 1 + |\varphi_1| + |\varphi_2|$

— $|\texttt{imp}(\varphi_1, \varphi_2)| = 1 + |\varphi_1| + |\varphi_2|$

La taille d'une formule correspond au nombre de connecteurs logiques apparaissant dans sa construction, autrement dit au nombre de nœuds internes de son arbre syntaxique.

Définition 5 (Hauteur d'une formule logique)

La hauteur d'une formule logique φ, notée $h(\varphi)$ est définie de manière inductive de la façon suivante :

— $h(\top) = 0$

— $h(\bot) = 0$

— $\forall p \in \mathcal{P}$, $h(p) = 0$

— $h(\texttt{not}(\varphi)) = 1 + h(\varphi)$

— $h(\texttt{and}(\varphi_1, \varphi_2)) = 1 + \max(h(\varphi_1), h(\varphi_2))$

— $h(\texttt{or}(\varphi_1, \varphi_2)) = 1 + \max(h(\varphi_1), h(\varphi_2))$

— $h(\texttt{imp}(\varphi_1, \varphi_2)) = 1 + \max(h(\varphi_1), h(\varphi_2))$

Elle correspond à la hauteur de l'arbre syntaxique, en prenant comme convention qu'une formule vide (non évaluable !) serait de hauteur -1.

I. 3. Écriture infixe linéaire

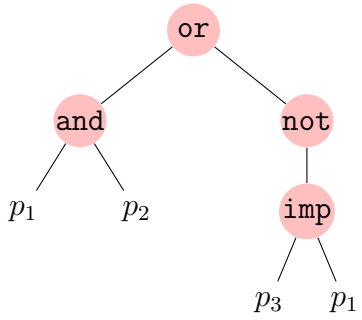
Nous avons vu une situation très similaire avec les expressions algébriques : les éléments de base étaient la ou les indéterminées x, y, z..etc ainsi que les valeurs numériques constantes (nous nous étions limités aux entiers). Nous avions introduit 3 constructeurs d'arité 2 : `add`, `mult` et `sub`.

Et nous avions vu que l'écriture classique des expressions algébrique correspondait à un parcours infixe, en remplaçant `add` par le symbole d'opérateur infixe $+$, `mult` par $\times$...etc

Et bien cela fonctionne de manière analogue pour les formules logiques : on peut écrire une formule logique de manière linéaire infixée : les constructeurs logiques `and`, `or` et `imp` sont remplacés respectivement par $\wedge$, $\vee$ et $\rightarrow$.
Le constructeur `not` étant d'arité 1, les notations infixe et préfixe sont confondues et on note simplement $\neg$.

Exemple 1

Par exemple, la formule logique $\texttt{or}\,(\texttt{and}(p_1, p_2), \text{not}(\texttt{imp}(p_3, p_1)))$ correspond à l'arbre syntaxique suivant :



et son écriture linéaire infixée est :

$$((p_1 \wedge p_2) \vee (\neg\,(p_3 \rightarrow p_1)))$$

Remarque (Parenthèses). Les parenthèses externes sont, en toute rigueur, nécessaires pour pouvoir définir de manière inductive les formules logiques en utilisant cette écriture linéaire infixe. On parle alors de **formules linéaires strictes**.
Par ailleurs, les parenthèses internes sont indispensables pour lever toute ambiguïté sur l'enchaînement des constructeurs. Par exemple, l'écriture $(x \wedge y \vee z)$ est ambiguë car, selon l'ordre d'évaluation, l'arbre syntaxique obtenu sera différent.
$((x \wedge y) \vee z)$ et $(x \wedge (y \vee z))$ n'ont en effet pas le même arbre syntaxique !

Théorème 1 (Théorème de lecture unique des formules strictes)

Si une formule est bien écrite, on doit pouvoir construire sans ambiguïté son arbre syntaxique et associer chaque sous-formule avec un sous-arbre de l'arbre syntaxique.
On appelle ce résultat le **théorème de lecture unique des formules strictes**.

II. Sémantique des formules en logique propositionnelle

Nous utilisons dans cette séquence des mots utilisés dans les études linguistiques et dans la théorie des langages :

— syntaxe = forme = signifiant

— sémantique = fond/sens = signifié

II. 1. Valuations

$\mathbb{B} = \{V, F\}$ ensemble des booléens (2 éléments)

Définition 6 (Valuation (aussi appelée distribution de vérité))

Soit $\mathcal{P}$ un ensemble de variables propositionnelles.
Une **valuation** v (aussi appelée **distribution de vérité**) est une **fonction** qui, à chaque variable propositionnelle $p \in \mathcal{P}$, associe une valeur de vérité V ou F, c'est-à-dire l'un des deux éléments de $\mathbb{B}$:

$$\begin{array}{rccc} v : & \mathcal{P} & \to & \mathbb{B} \\ & p & \mapsto & V \text{ ou } F \end{array}$$

Définir une valuation, c'est donc choisir, pour chacune des variables propositionnelles impliquées dans une expression logique, une valeur de vérité V ou F.

Notation 2

On notera $\mathcal{V}$ l'ensemble des valuations sur $\mathcal{P}$.

Remarque (Dénombrement des valuations possibles). Pour une expression logique faisant intervenir n variables propositionnelles, il y a donc 2^n valuations possibles car il y a 2 choix possibles de valeurs (V ou F) pour la première variable, 2 choix pour la seconde..etc, tous ces choix étant indépendants.

Exemple 2

Soit l'expression logique suivant : $\varphi = (p \wedge q) \vee (r \wedge \neg p)$. Cette expression est construite à l'aide de 3 variables propositionnelles (formules atomiques) : $\mathcal{P} = \{p, q, r\}$ Il y a donc $2^3 = 8$ valuations possibles. Une valuation possible est, par exemple, la fonction définie de manière exhaustive :

$$\begin{array}{rccc} v : & \mathcal{P} & \to & \mathbb{B} \\ & p & \mapsto & V \\ & q & \mapsto & F \\ & r & \mapsto & F \end{array}$$

II. 2. Évaluation sémantique d'une formule logique

II. 2. a. Fonctions booléennes associées aux constructeurs

Définition 7 (Fonction booléenne)

Une fonction booléenne à n arguments est une fonction de $\mathbb{B}^n$ dans $\mathbb{B}$.

A chaque connecteur logique peut être associée une fonction booléenne permettant de calculer la valeur de vérité d'une formule en fonction de la valeur de vérité des sous formules.
Par exemple, le constructeur logique `et`, noté $\wedge$ dans l'écriture linéaire infixe, est associé à la fonction booléenne :

$$\begin{array}{rccl} f_{\text{and}} : & \mathbb{B}^2 & \to & \mathbb{B} \\ & (\varphi_1, \varphi_2) & \mapsto & \begin{cases} f_{\text{and}}(\varphi, \psi) = V & \text{si } \varphi_1 = V \text{ et } \varphi_2 = V \\ f_{\text{and}}(\varphi, \psi) = F & \text{dans tous les autres cas} \end{cases} \end{array}$$

Définition 8 (Table de vérité d'une fonction booléenne)

Une fonction booléenne est souvent représentée par son tableau de valeurs, appelé, dans ce contexte **table de vérité**.

Exemple 3

La table de vérité de la fonction booléenne f_{and} est donnée par :

φ	ψ	$f_{\text{and}}(\varphi, \psi)$
F	F	F
V	F	F
F	V	F
V	V	V

Exercice 1.
Écrire les fonctions booléennes associées aux autres constructeurs logiques et établir leurs tables de vérité.

On prêtera particulièrement attention à la table de vérité de l'implication, qui est peut-être la moins naturelle :

φ	ψ	$f_{\text{imp}}(\varphi, \psi)$
F	F	V
V	F	F
F	V	V
V	V	V

II. 2. b. Fonction d'évaluation d'une formule logique

Une fois définie la fonction booléenne de chaque constructeur logique, on peut définir la fonction d'évaluation de n'importe quelle formule logique par induction (cf séquence sur

l'induction structurelle)

Définition 9 (Valeur d'une formule pour une valuation v donnée)

Soit $\mathcal{P}$ un ensemble de variables propositionnelles.
Soit $\mathcal{F}$ l'ensemble des formules logiques construites inductivement à partir de $\mathcal{P}$ en utilisant les constructeurs logiques `and`, `or`, `imp` et `not`.
Pour une formule $\varphi \in \mathcal{F}$, la valeur sémantique de la formule φ pour une valuation v est notée $[\![\varphi]\!]_v$ et elle est calculée de la manière inductive suivante :

- Si $\varphi = \top$ (donc un élément minimal), alors $[\![\varphi]\!]_v = V$
- Si $\varphi = \bot$ (donc un élément minimal), alors $[\![\varphi]\!]_v = F$
- Si φ est une variable propositionnelle (donc un élément minimal), c'est-à-dire $\varphi = p \in \mathcal{P}$, alors $[\![\varphi]\!]_v = v(p)$
- Si $\varphi = \text{not}(\varphi_1)$, alors $[\![\varphi]\!]_v = f_{\text{not}}\left([\![\varphi_1]\!]_v\right)$
- Si $\varphi = \text{and}(\varphi_1, \varphi_2)$, alors $[\![\varphi]\!]_v = f_{\text{and}}\left([\![\varphi_1]\!]_v, [\![\varphi_2]\!]_v\right)$
- Si $\varphi = \text{or}(\varphi_1, \varphi_2)$, alors $[\![\varphi]\!]_v = f_{\text{or}}\left([\![\varphi_1]\!]_v, [\![\varphi_2]\!]_v\right)$
- Si $\varphi = \text{imp}(\varphi_1, \varphi_2)$, alors $[\![\varphi]\!]_v = f_{\text{imp}}\left([\![\varphi_1]\!]_v, [\![\varphi_2]\!]_v\right)$

Pour une valuation v donnée, on peut donc **évaluer** la valeur de l'expression logique globale en cette valuation.

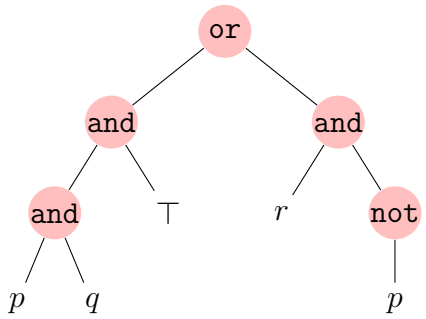
- On commence par remplacer chaque chaque feuille de l'arbre par sa valeur de vérité : $\top$ par V, $\bot$ par F, et chaque variable propositionnelle par la valeur booléenne V ou F qui lui est attribuée par la valuation v.
- Puis on applique inductivement les fonctions booléennes de chaque constructeur logique, en partant du fond de l'arbre et en remontant à la racine de la formule.

Définition 10 (Table de vérité d'une formule logique)

L'ensemble de toutes les valeurs de vérité obtenues pour toutes les valuations est généralement représenté dans une table de vérité.
On y liste les 2^n valuations possibles, puis, pour chaque valuation v, on indique la valeur de vérité $[\![\varphi]\!]_v$ obtenue en évaluant la formule φ pour cette valuation v.

Exemple 4

Soit l'expression logique suivant : $\varphi = ((p \wedge q) \wedge \top) \vee (r \wedge \neg p)$.
Cette expression est construite à l'aide de 3 variables propositionnelles (formules atomiques) : $\mathcal{P} = \{p, q, r\}$ Son arbre syntaxique s'obtient sans ambiguïté à l'aide de l'écriture infixe stricte ci-dessus :



Détaillons le calcul de $[\![\varphi]\!]_v$ pour la valuation suivante :

$$\begin{array}{rccc} v : & \mathcal{P} & \to & \mathbb{B} \\ & p & \mapsto & V \\ & q & \mapsto & F \\ & r & \mapsto & F \end{array}$$

on a : $[\![\varphi]\!]_v = F$
Pour l'ensemble des valuations, la table de vérité est :

p	q	r	$(p \wedge q)$	$\neg p$	$(r \wedge \neg p)$	φ
F	F	F	F	V	F	F
V	F	F	F	F	F	F
F	V	F	F	V	F	F
V	V	F	V	F	F	V
F	F	V	F	V	V	V
V	F	V	F	F	F	F
F	V	V	F	V	V	V
V	V	V	V	F	F	V

II. 3. Satisfiabilité, modèle

Définition 11 (Modèle associé à une expression logique)

On appelle **modèle** associé à une expression logique φ une valuation v telle que l'évaluation de φ sur cette valuation v vaut V (vrai).
Autrement dit :

$$v \text{ est un modèle pour } \varphi \text{ si et seulement si } [\![\varphi]\!]_v = V$$

Notation 3

On notera $\mathrm{M}(\varphi)$ l'ensemble des valuations modèles de φ, c'est-à-dire l'ensemble des valuations qui rendent φ vraie :

$$\mathrm{M}(\varphi) = \{v \in \mathcal{V},\ [\![\varphi]\!]_v = V\} \subseteq \mathcal{V}$$

Définition 12 (Tautologie)

On appelle **tautologie** une formule logique dont l'évaluation a pour résultat V (vrai), quelque soit la valuation choisie.
En d'autres termes, une tautologie est une formule φ telle que toutes les valuations possibles sont des modèles :

$$\mathrm{M}(\varphi) = \mathcal{V}$$

Définition 13 (Antilogie)

On appelle **antilogie** une expression logique dont l'évaluation a pour résultat F (faux), quelque soit la valuation choisie.
En d'autres termes, une antilogie est une formule φ telle que aucune valuation n'est un modèle :

$$\mathrm{M}(\varphi) = \emptyset$$

Définition 14 (Satisfiabilité d'une expression logique)

Une formule φ est **satisfiable** s'il existe au moins une valuation v qui rend φ vraie :

$$\exists v \in \mathcal{V},\ [\![\varphi]\!]_v = V$$

Autrement dit, φ est satisfiable si et seulement si $\mathrm{M}(\varphi) \neq \emptyset$, s'il existe au moins une valuation qui est un modèle.

Exemples 5

1. La formule $(p \wedge q) \vee \neg p$ est satisfiable car si p prend la valeur F, la formule est évaluée à V.
2. La formule $(p \wedge \neg p)$ n'est pas satisfiable car aucune valeur de p ne peut rendre la formule vraie.

Définition 15 (Conséquence sémantique d'une formule)

Une formule ψ est une **conséquence sémantique** d'une formule φ si toute valuation v qui rend φ vraie, rend aussi ψ vraie :

$$\forall v \in \mathcal{V}, \text{ si } [\![\varphi]\!]_v = V \text{ alors } [\![\psi]\!]_v = V$$

On note $\varphi \vDash \psi$ pour dire que ψ est une conséquence sémantique de φ.

Définition 16 (Conséquence sémantique d'un ensemble de formules)

Une formule ψ est une conséquence sémantique d'un ensemble de formule Γ si, pour toute valuation v telle que toutes les formules de Γ sont vraies, alors v rend aussi ψ vraie.

$$\forall v \in \mathcal{V}, \text{ si } , (\forall \varphi \in \Gamma, [\![\varphi]\!]_v = V) \text{ alors } [\![\psi]\!]_v = V$$

On note $\Gamma \vDash \psi$

Remarque (Subtilités...). Différence $\Rightarrow$ $\rightarrow$ et $\vDash$:

- $\rightarrow$: on parle d'implication **matérielle** ; il s'agit d'une pure syntaxe, de langage objet, on se moque du sens, de la sémantique ;
- $\vDash$ sémantique : si une formule ou un ensemble de formule est vraie, alors je peux en conclure qu'une autre formule est elle aussi vraie ;
- $\Rightarrow$: $\varphi \Rightarrow \psi$ ne peut être écrit que si la formule $\varphi \rightarrow \psi$ est une tautologie : on est dans ce que l'on nomme parfois la méta-linguistique...

II. 4. Équivalence sémantique

Définition 17 (Équivalence sémantique)

Deux formules φ et ψ sont dites **équivalentes** lorsque, pour toute valuation v, on a :

$$[\![\varphi]\!]_v = [\![\psi]\!]_v$$

Autrement dit, deux formules sont équivalentes si et seulement si elles ont la même table de vérité.

Notation 4

On note $\varphi \equiv \psi$ pour indiquer que deux formules sont sémantiquement équivalentes. Le symbole $=$ sera réservé à la définition par induction d'une formule.

Tout comme il existe des règles d'équivalence entre expressions algébriques, qui vous permettent de transformer des expressions algébriques en des expressions équivalentes (par exemple quand vous factorisez ou développez), il existe des règles d'équivalences entre formules logiques qui permettent de transformer certaines formules et donc d'effectuer des

calculs sur ces formules.

Définition 18 (Constructeur $\leftrightarrow$)

On voit parfois apparaître le constructeur logique $\leftrightarrow$ dans les formules. Ce constructeur n'est pas indispensable. Il est défini par l'équivalence sémantique suivante :

$$(\varphi \leftrightarrow \psi) \equiv ((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$$

Remarque (Lien et différence entre $\leftrightarrow$ et $\equiv$). $(\varphi \leftrightarrow \psi)$ est une formule, $\varphi \equiv \psi$ n'en est pas une !
$\equiv$ indique juste que, **d'un point de vue sémantique**, ψ et ψ sont indiscernables.
Par contre, $\varphi \equiv \psi$ si et seulement si $(\varphi \leftrightarrow \psi)$ est une tautologie.
Enfin, le symbole $\Leftrightarrow$ a encore un autre sens subtile. On écrit que $\varphi \Leftrightarrow \psi$ que si la formule $\varphi \leftrightarrow \psi$ est évaluée à V.

II. 5. Substitution

Il est courant de souhaiter modifier une formule, de façon à rendre son expression plus simple, ou plus facile à manipuler, et ceci en gardant bien sûr la sémantique de la formule, c'est-à-dire, sans modifier l'ensemble de ses modèles.
La substitution d'une formule ψ par une formule ψ' dans une troisième formuleφ (notée $\varphi^{[\psi \leftarrow \psi']}$ consiste à remplacer chaque occurrence de ψ dans φ par ψ'.

Exemple 6

Prenons par exemple les formules $\varphi = (\neg p \vee q) \wedge (\neg p \vee \neg r)$, $\psi' = \neg p$ et $\psi' = q \rightarrow p$, alors :

$$\varphi^{[\psi \leftarrow \psi']} = ((q \rightarrow p) \vee q) \wedge ((q \rightarrow p) \vee \neg r).$$

Plus formellement, la substitution est définie de manière inductive :

Définition 19 (Substitution)

Soient φ, ψ, et ψ' trois formules du calcul propositionnel :

- si ψ n'est pas une sous-formule de φ alors $\varphi^{[\psi \leftarrow \psi']} = \varphi$
- sinon si $\varphi = \psi$ alors $\varphi^{[\psi \leftarrow \psi']} = \psi'$
- sinon
  - si $\varphi = \neg\varphi_1$ alors $\varphi^{[\psi \leftarrow \psi']} = \neg\left(\varphi_1^{[\psi \leftarrow \psi']}\right)$
  - si $\varphi = \varphi_1 \wedge \varphi_2$ alors $\varphi^{[\psi \leftarrow \psi']} = \varphi_1^{[\psi \leftarrow \psi']} \wedge \varphi_2^{[\psi \leftarrow \psi']}$
  - si $\varphi = \varphi_1 \vee \varphi_2$ alors $\varphi^{[\psi \leftarrow \psi']} = \varphi_1^{[\psi \leftarrow \psi']} \vee \varphi_2^{[\psi \leftarrow \psi']}$
  - si $\varphi = \varphi_1 \rightarrow \varphi_2$ alors $\varphi^{[\psi \leftarrow \psi']} = \varphi_1^{[\psi \leftarrow \psi']} \rightarrow \varphi_2^{[\psi \leftarrow \psi']}$

Propriété 1 (Principe de substitution équivalente)

Soient φ, ψ, et ψ' trois formules du calcul propositionnel.
Si $\psi \equiv \psi'$ alors $\varphi \equiv \varphi^{[\psi \leftarrow \psi']}$

Attention. Attention, dans le cas général, si on substitue par quelque chose qui n'est pas équivalent sémantiquement à ce que l'on remplace, la substitution ne conserve pas la sémantique de la formule !

II. 6. Règles de calculs sur les formules de la logique propositionnelle

Forts de ce principe de substitution, on peut travailler sur les formules en substituant certaines sous formules ψ d'une formule φ par des formules équivalentes. Pour cela, on peut utiliser des équivalences sémantiques classiques, détaillées ci-dessous :

lois de Morgan	$(\neg(\varphi \wedge \psi)) \equiv ((\neg\varphi) \vee (\neg\psi))\,, \quad (\neg(\varphi \vee \psi)) \equiv ((\neg\varphi) \wedge (\neg\psi))$
implication	$(\varphi \to \psi) \equiv (\psi \vee \neg\varphi)$
contraposition	$(\varphi \to \psi) \equiv (\neg\psi \to \neg\varphi)$
curryfication	$((\varphi \wedge \psi) \to \theta) \equiv (\varphi \to (\psi \to \theta))$
non-contradiction	$(\varphi \wedge \neg\varphi) \equiv \bot$
tiers-exclu	$(\varphi \vee \neg\varphi) \equiv \top$
double négation	$(\neg(\neg\varphi)) \equiv \varphi$
$\top$ élément neutre de $\wedge$	$(\varphi \wedge \top) \equiv \varphi$
$\bot$ élément neutre de $\vee$	$(\varphi \vee \bot) \equiv \varphi$
$\bot$ élément absorbant de $\wedge$	$(\varphi \wedge \bot) \equiv \bot$
$\top$ élément absorbant de $\vee$	$(\varphi \vee \top) \equiv \top$
idempotence de $\vee$	$\varphi \vee \varphi \equiv \varphi$
idempotence de $\wedge$	$\varphi \wedge \varphi \equiv \varphi$
absorption	$\varphi \wedge (\varphi \vee \psi) \equiv \varphi\,, \quad \varphi \vee (\varphi \wedge \psi) \equiv \varphi$
commutativité de $\wedge$	$(\varphi \wedge \psi) \equiv (\psi \wedge \varphi)$
commutativité de $\vee$	$(\varphi \vee \psi) \equiv (\psi \vee \varphi)$
associativité de $\wedge$	$(\varphi \wedge \psi) \wedge \theta \equiv (\varphi \wedge (\psi \wedge \theta))$
associativité de $\vee$	$(\varphi \vee \psi) \vee \theta \equiv \varphi \vee (\psi \vee \theta)$
distributivité de $\wedge$ sur $\vee$	$(\varphi \wedge (\psi \vee \theta)) \equiv ((\varphi \wedge \psi) \vee (\varphi \wedge \theta))$
distributivité de $\vee$ sur $\wedge$	$(\varphi \vee (\psi \wedge \theta)) \equiv ((\varphi \vee \psi) \wedge (\varphi \vee \theta))$

Elles sont à connaître sur le bout des doigts et à savoir redémontrer (avec des tables de vérité par exemple) !

II. 7. Formes normales

On cherche souvent à réécrire des formules logique sous une forme normalisée (d'où le terme formes normales), c'est-à-dire sous une forme plus sympathique permettant d'automatiser un certain nombre d'analyses.
On détaille ici trois écritures normalisées utiles en logique propositionnelle :

— écriture sous forme normale négative NNF

— écriture sous forme normale disjonctive DNF

— écriture sous forme normale conjonctive CNF

II. 7. a. Écriture sous forme normale négative (NNF)

Définition 20 (Littéral)

Un littéral est une variable propositionnelle ou la négation d'une variable propositionnelle

Définition 21 (Forme normale négative (NNF))

Une forme normale négative (Normal Negative Form ou NNF) est une écriture sémantiquement équivalente d'une formule logique ne faisant intervenir que des littéraux, des $\wedge$ (conjonctions) et des $\vee$ (disjonctions).

Méthodologie.
Pour obtenir une telle forme à partir d'une formule, on procède comme ceci :

— On remplace toutes les connecteurs $\rightarrow$ et $\leftrightarrow$ par les formules équivalentes suivantes :

$$\varphi \rightarrow \psi \equiv \psi \vee (\neg\varphi)$$

$$\varphi \leftrightarrow \psi \equiv (\psi \vee (\neg\varphi)) \wedge (\varphi \vee (\neg\psi))$$

— On distribue les $\neg$ en utilisant les lois de Morgan

$$\neg(\varphi \wedge \psi) \equiv (\neg\varphi) \vee (\neg\psi) \quad \neg(\varphi \vee \psi) \equiv (\neg\varphi) \wedge (\neg\psi)$$

— On simplifie les double-négations $\neg\neg$:

$$\neg(\neg\varphi) \equiv \varphi$$

Exercice 2.
Mettre sous forme normale négative la formule suivante :

$$(p \wedge (\neg r)) \rightarrow ((\neg p) \wedge (\neg(q \wedge \neg r)))$$

Corrigé de l'exercice 1. *[Retour à l'énoncé]*

	$(p \wedge (\neg r)) \rightarrow ((\neg p) \wedge (\neg(q \wedge \neg r)))$	
$\equiv$	$(\neg(p \wedge (\neg r))) \vee ((\neg p) \wedge (\neg(q \wedge \neg r)))$	règle d'implication
$\equiv$	$((\neg p) \vee (\neg\neg r)) \vee ((\neg p) \wedge (\neg q \vee (\neg\neg r)))$	loi de Morgan
$\equiv$	$((\neg p) \vee r) \vee ((\neg p) \wedge (\neg q \vee r))$	double négation
$\equiv$	$(\neg p \vee r) \vee (\neg p \wedge (\neg q \vee r))$	allègement des parenthèses facultatives

On a bien une forme normale négative NNF.

II. 7. b. Écriture sous forme normale disjonctive (DNF)

Définition 22 (Clause conjonctive)

Une clause conjonctive est une conjonction de littéraux.

Exemple 7

Si p, q et r sont des variables propositionnelles, $(\neg p) \wedge q \wedge (\neg r)$ est une clause conjonctive à 3 littéraux.

Définition 23 (Forme normale disjonctive (DNF))

Une forme normale disjonctive (***D**isjonctive Normal Form* ou DNF) est une écriture sémantiquement équivalente d'une formule logique sous la forme d'une **disjonction de clauses conjonctives**.

Remarque. Une DNF est une NNF... la réciproque est bien sûr fausse !

Exemple 8

Soit p, q et r des variables propositionnelles. La formule logique suivante est écrite sous forme normale disjonctive :

$$\underbrace{((\neg p) \wedge q \wedge (\neg r))}_{\text{clause à 3 litt.}} \vee \underbrace{(p \wedge (\neg q))}_{\text{clause à 2 litt.}} \vee \underbrace{(\neg r)}_{\text{clause à 1 litt.}}$$

Il y a 3 clauses conjonctives : une clause à 3 littéraux, une à 2 littéraux et une ayant 1 seul littéral.

Exemple 9

Toutes les formules suivantes sont sous forme DNF :

- $(p_1 \wedge \neg p_2 \wedge \neg p_3) \vee (\neg p_4 \wedge p_5 \wedge p_6)$: 2 clauses conjonctives à 3 littéraux
- $(p_1 \wedge p_2) \vee (p_3)$: 2 clauses conjonctives, 1 à 2 littéraux, 1 à 1 seul littéral
- $(p_1 \wedge p_2)$: 1 seule clause conjonctive à 2 littéraux
- (p_1) : 1 seule clause conjonctive à 1 littéral

Exemple 10 (Contre-exemples)

Les formules suivantes **ne sont pas** sous forme DNF :

- $(p_1 \vee p_2) \wedge p_3$
- $\neg(p_2 \wedge p_3)$ car le $\wedge$ est situé à l'intérieur d'une négation.
- $p_1 \vee (p_2 \wedge (p_4 \vee p_5))$, car le $\vee$ est situé à l'intérieur d'un $\wedge$

Méthodologie (Méthode 1 : par réécriture et utilisation des règles d'équivalence).
Pour obtenir une telle forme, on procède comme ceci :

— On écrit la formule sous une forme NNF si ce n'est pas déjà le cas ;

— On utilise la distributivité ;

— On simplifie ce qui peut l'être avec les lois d'équivalences que l'on connaît.

Exemple 11 (Méthode 1)

On repart de la forme NNF de $\varphi = (p \wedge (\neg r)) \rightarrow ((\neg p) \wedge (\neg(q \wedge \neg r)))$

$$\begin{array}{lll} & (p \wedge (\neg r)) \rightarrow ((\neg p) \wedge (\neg(q \wedge \neg r))) & \\ \equiv & (\neg p \vee r) \vee (\neg p \wedge (\neg q \vee r)) & \text{forme NNF} \\ \equiv & ((\neg p) \vee r) \vee ((\neg p \wedge \neg q) \vee (\neg p \wedge r)) & \text{distributivité} \\ \equiv & (\neg p) \vee r \vee (\neg p \wedge \neg q) \vee (\neg p \wedge r) & \text{associativité de } \vee \end{array}$$

On a bien obtenu une DNF à 4 clauses, dont 2 clauses à 1 seul littéral, et 2 clauses à 2 littéraux.

Méthodologie (Méthode 2 : à partir de la table de vérité de la formule).
On repère les valuations modèles v dans la table de vérité (lignes pour lesquelles φ est évaluée à V)
Il suffit alors d'écrire une **clause conjonctive** qui est vraie pour cette valuation.
On obtient une formule équivalente en écrivant finalement la **d**isjonction de ces conjonctions

Exemple 12 (Méthode 2)

Faisons la table de vérité de φ :

p	q	r	$p \wedge (\neg r)$	$(\neg p)$	$(\neg(q \wedge \neg r))$	$((\neg p) \wedge (\neg(q \wedge \neg r))$	φ
F	F	F	F	V	V	V	V
V	F	F	V	F	V	F	F
F	V	F	F	V	F	F	V
F	F	V	F	V	V	V	V
V	V	F	V	F	F	F	F
V	F	V	F	F	V	F	V
F	V	V	F	V	V	V	V
V	V	V	F	F	V	F	V

Il y a 6 valuations qui rendent φ vraie :

Valuation		Clause conjonctive rendant cette valuation vraie
(F, F, F)	$\rightarrow$	$\neg p \wedge \neg q \wedge \neg r$
(F, V, F)	$\rightarrow$	$\neg p \wedge q \wedge \neg r$
(F, F, V)	$\rightarrow$	$\neg p \wedge \neg q \wedge r$
(V, F, V)	$\rightarrow$	$p \wedge \neg q \wedge r$
(F, V, V)	$\rightarrow$	$\neg p \wedge q \wedge r$
(V, V, V)	$\rightarrow$	$p \wedge q \wedge r$

Et on en déduit une DNF :

$$\varphi \equiv (\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (p \wedge q \wedge r)$$

Propriété 2

Toute formule logique admet une formule équivalente écrite sous forme normale disjonctive

Démonstration

Preuve constructive liée à la méthode que l'on vient de voir : toute formule logique possède une table de vérité qui permet d'écrire sa DNF !

Propriété 3

Il n'y a pas unicité de l'écriture DNF d'une formule logique

Exemple 13

Nous avons déjà trouvé deux écritures DNF pour φ, par deux méthodes différentes :

$$\varphi \equiv (\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (p \wedge q \wedge r)$$

et

$$\varphi \equiv (\neg p) \vee r \vee (\neg p \wedge \neg q) \vee (\neg p \wedge r)$$

Et en étant encore plus malin, on peut même voir (règle d'absoption) que :

$$\varphi \equiv (\neg p) \vee r$$

On le voit, selon la méthode choisie et les astuces utilisées, on peut avoir des DNF complexes, avec beaucoup de clauses, ou bien des DNF beaucoup plus simples, avec très peu de clauses conjonctives, ces clauses conjonctives ayant elles-même très peu de littéraux...

II. 7. c. Écriture sous forme normale conjonctive (CNF)

Définition 24 (Clause disjonctive)

Une clause disjonctive est une disjonction de littéraux

Exemple 14

Si p, q et r sont des variables propositionnelles, $(\neg p) \vee q \vee (\neg r)$ est une clause disjonctive à 3 littéraux.

Définition 25 (Forme normale conjonctive (CNF))

Une forme normale conjonctive (*Conjonctive Normal Form* ou CNF) est une écriture sémantiquement équivalente d'une formule logique sous la forme d'une conjonction de clauses disjonctives

Remarque. Une CNF est une NNF... la réciproque est bien sûr fausse !

Exemple 15

Soit p, q et r des variables propositionnelles. La formule logique suivante est écrite sous forme normale conjonctive :

$$((\neg p) \vee q \vee (\neg r)) \wedge (p \vee (\neg q)) \wedge (\neg r)$$

Il y a 3 clauses disjonctives : une clause à 3 littéraux, une à 2 littéraux et une ayant un seul littéral.

Exemple 16

Toutes les formules suivantes sont sous forme CNF :

- $(p_1 \vee \neg p_2 \vee \neg p_3) \wedge (\neg p_4 \vee p_5 \vee p_6)$: 2 clauses disjonctives à 3 littéraux
- $(p_1 \vee p_2) \wedge (p_3)$: 2 clauses disjonctives, une à 2 littéraux, une à un seul littéral
- $(p_1 \vee p_2)$: 1 clause disjonctive à deux littéraux
- (p_1) : 1 clause disjonctive à 1 littéral

Exemple 17 (Contre-exemples)

Les formules suivantes **ne sont pas** sous forme CNF :

- $\neg(p_2 \vee p_3)$ car le $\vee$ est situé à l'intérieur d'une négation.
- $(p_1 \wedge p_2) \vee p_3$
- $p_1 \wedge (p_2 \vee (p_4 \wedge p_5))$, car le $\wedge$ est situé à l'intérieur d'un $\vee$

Méthodologie (Méthode 1 : par réécriture et utilisation des règles d'équivalence). Pour obtenir une telle forme, on procède comme ceci :

— On écrit la formule sous une forme NNF si ce n'est pas déjà le cas ;

— On utilise la distributivité ;

— On simplifie ce qui peut l'être.

Exemple 18 (Méthode 1)

Toujours sur le même exemple, on repart de la forme NNF de $(p \wedge (\neg r)) \rightarrow ((\neg p) \wedge (\neg(q \wedge \neg r)))$

	$(p \wedge (\neg r)) \rightarrow ((\neg p) \wedge (\neg(q \wedge \neg r)))$	
$\equiv$	$(\neg p \vee r) \vee (\neg p \wedge (\neg q \vee r))$	forme NNF
$\equiv$	$((\neg p \vee r) \vee \neg p)) \wedge ((\neg p \vee r) \vee (\neg q \vee r))$	distributivité
$\equiv$	$(\neg p \vee r \vee \neg p) \wedge (\neg p \vee r \vee \neg q \vee r)$	associativité de $\vee$
$\equiv$	$(\neg p \vee r) \wedge (\neg p \vee \neg q \vee r)$	idempotence et commutativité de $\vee$ (on a une CNF !)
$\equiv$	$((\neg p \vee r) \vee \bot) \wedge ((\neg p \vee r) \vee (\neg q))$	$\bot$ neutre pour $\vee$
$\equiv$	$(\neg p \vee r) \vee (\bot \wedge \neg q)$	distributivité
$\equiv$	$(\neg p \vee r) \vee \bot$	$\bot$ absorbant pour $\wedge$
$\equiv$	$(\neg p \vee r)$	$\bot$ neutre pour $\vee$

Exercice 3.
Mettez les contre-exemples ci-dessous sous forme CNF !

Méthodologie (Méthode 2 : à partir de la table de vérité de la formule).
On écrit la DNF de $\neg\varphi$ à partir de la table de vérité de $\neg\varphi$
On écrit la CNF de φ en prenant la négation de la DNF et en appliquant les lois de Morgan, qui vont transformer les clauses conjonctives en clauses disjonctives et transformer les $\vee$ de la DNF en $\wedge$.

Exemple 19

Faisons la table de vérité de $\neg\varphi$:

p	q	r	φ	$\neg\varphi$
F	F	F	V	F
V	F	F	F	V
F	V	F	V	F
F	F	V	V	F
V	V	F	F	V
V	F	V	V	F
F	V	V	V	F
V	V	V	V	F

Il y a 2 valuations qui rendent $\neg\varphi$ vraie :

Valuation		Clause conjonctive rendant cette valuation vraie
(V, F, F)	$\rightarrow$	$p \wedge \neg q \wedge \neg r$
(V, V, F)	$\rightarrow$	$p \wedge q \wedge \neg r$

Et on en déduit une DNF pour $\neg\varphi$:

$$\neg\varphi \equiv (p \wedge \neg q \wedge \neg r) \vee (p \wedge q \wedge \neg r)$$

Et on trouve une CNF pour φ en écrivant la négation de cette DNF et en utilisant les lois de Morgan :

$$\varphi \equiv \neg\left((p \wedge \neg q \wedge \neg r) \vee (p \wedge q \wedge \neg r)\right)$$

$$\varphi \equiv \neg\left(p \wedge \neg q \wedge \neg r\right) \wedge \neg\left(p \wedge q \wedge \neg r\right)$$

$$\varphi \equiv (\neg p \vee \neg(\neg q) \vee \neg(\neg) r) \wedge (\neg p \vee \neg q \vee \neg(\neg r))$$

$$\varphi \equiv (\neg p \vee q \vee r) \wedge (\neg p \vee \neg q \vee r)$$

Et on montre que cette formules est bien équivalente à celle trouvée par l'autre méthode :

	$(\neg p \vee q \vee r) \wedge (\neg p \vee \neg q \vee r)$	
$\equiv$	$((\neg p \vee r) \vee q) \wedge ((\neg p \vee r) \vee \neg q)$	commutativité de $\vee$
$\equiv$	$(\neg p \vee r) \wedge (q \vee \neg q)$	distributivité
$\equiv$	$(\neg p \vee r) \wedge \top$	tiers-exclu
$\equiv$	$(\neg p \vee r)$	$\top$ neutre pour $\wedge$

Propriété 4

Il n'y a pas unicité de l'écriture CNF d'une formule logique

Méthodologie (Méthode 3 : distributivité).
Si on a déjà une DNF, on peut obtenir une CNF en utilisant la distributivité et réciproquement, si on a une CNF, on peut obtenir une DNF en utilisant la distributivité

Exemple 20

Imaginons que l'on a obtenu la DNF suivante :

$$(p_1 \wedge \neg p_2) \vee (p_3 \wedge \neg p_4 \wedge p1)$$

En utilisant la distributivité :

$$\begin{array}{lll} & (p_1 \wedge \neg p_2) \vee (p_3 \wedge \neg p_4 \wedge p1) & \\ \equiv & (p_1 \vee p_2) \wedge (p_1 \vee p_4) \wedge (p_1 \vee p_1) \wedge (\neg p_2 \vee p_3) \wedge (\neg p_2 \vee p_4) \wedge (\neg p_2 \vee p_1) & \text{distributivité} \\ \equiv & (p_1 \vee p_2) \wedge (p_1 \vee p_4) \wedge p_1 \wedge (\neg p_2 \vee p_3) \wedge (\neg p_2 \vee p_4) \wedge (\neg p_2 \vee p_1) & \text{idempotence} \end{array}$$

et on a bien une CNF

II. 7. d. Problème de l'explosion combinatoire

Les méthodes systématiques présentées pour transformer une formules logiques et obtenir une CNF ou une DNF marchent toujours, mais ont un gros problème : la taille de la formule (nombre de nœuds internes de l'arbre syntaxique) a tendance à croître de manière exponentielle en fonction du nombre de variables propositionnelles !

La méthode 2, utilisant la table de vérité, peut produire jusqu'à 2^n clauses pour une écriture DNF, et par dualité, pour une CNF.

La méthode 3, consistant à utiliser la distributivité sur une forme existante, amène également une explosion combinatoire du nombre de clauses, de manière mécanique :

$$\left(\ell_1^1 \wedge \ell_2^1 \wedge \ldots \ell_{k_1}^1\right) \vee \left(\ell_1^2 \wedge \ell_2^2 \wedge \ldots \ell_{k_2}^2\right) \vee \ldots \vee \left(\ell_1^c \wedge \ell_2^c \wedge \ldots \ell_{k_c}^c\right) \equiv \bigwedge_{(i_1,\ldots,i_c) \in [\![1,k_1]\!] \times \ldots \times [\![1,k_c]\!]} \left(\ell_{i_1}^1 \vee \ldots \vee \ell_{i_c}^c\right)$$

On passe d'une DNF à c clauses (1ère clause à k_1 littéraux, 2ème à k_2 littéraux..etc) à une CNF à $k_1 \times k_2 \times \ldots \times k_c$ clauses. La taille (= nombre de connecteurs logiques) de la DNF est $(k_1 - 1) + \ldots + (k_c - 1) + (c - 1) = k_1 + \ldots k_c - 1$, celle de la CNF obtenue par distributivité est $k_1 \times \ldots k_c \times (c - 1)$
Mais attention, tous les k_i, ainsi que le nombre de clauses c, dépendent du nombre n de variables propositionnelles et grandit lui aussi généralement avec n en l'absence de simplification ! On se retrouve donc souvent avec des croissances exponentielles !

Remarque (Transformation de Tseitin). Il existe une méthode de transformation en CNF qui garantit une maîtrise de la taille de la formule.

II. 8. Format DIMACS

Il s'agit d'un format classique permettant de stocker les informations associées à une formule logique dans un fichier texte. Voici un exemple :

```
c Fichier DIMACS pour la formule
p cnf 3 4
-2 0
-3 -1 2 0
1 -2 0
```

```
3 -2 0
```

Les lignes commençant par `c` sont des lignes de commentaires.
La ligne commençant par `p` (comme problème) décrit les informations essentielles concernant la formule logique :

- est-elle donnée sous forme CNF ou DNF ?
- nombre de variables propositionnelles n, dans l'exemple $n = 3$
- nombre de clauses de la formule (clauses disjonctives si CNF, clauses conjonctives si DNF)

Ensuite, chaque ligne représente une clause. Le `0` en fin de ligne sert à indiquer la fin de la description de la clause.
Les variables propositionnelles sont numérotées à partir de 1 jusqu'à n, 0 étant réservé pour indiquer la fin d'une ligne.
Une variable propositionnelle est simplement désignée par son entier i, sa négation par l'entier $-i$.
Ainsi, le fichier exemple, sert à stocker l'information relative à la formule logique :

$$(\neg p_2) \wedge (\neg p_3 \vee \neg p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (p_3 \vee \neg p_2)$$

Nous verrons en TP un type OCaml personnalisé permettant de stocker une formule sous ce format.

III. Problème SAT

III. 1. Définition

Le problème SAT est un problème de décision : ce problème pose la question suivante, en apparence simple :

Pour une formule de logique propositionnelle, la formule φ est-elle satisfiable ?
Autrement dit, existe-t-il au moins un modèle pour φ ?

Définition 26 (Problème SAT)

Le problème SAT est un problème de décision qui pose la question de savoir si une formule de la logique propositionnelle φ est satisfiable ou non.
φ est appelée une **instance** du problème.

- $SAT(\varphi) = V$ si φ est satisfiable
- $SAT(\varphi) = F$ si φ est une antilogie (aucune valuation ne rend φ vraie)

Exemple 21

Donnée d'entrée : une formule de la logique propositionnelle :

$$\varphi = (\neg p_1 \vee p_2 \vee p_3) \wedge (p_1 \vee p_2) \wedge (\neg p_2 \vee p_3) \wedge (\neg p_3 \vee \neg p_1).$$

Question : Est-ce que la formule φ admet au moins un modèle ?
Réponse : Pour cet exemple, la réponse est oui : la valuation définie par $v(p_1) = F$, $v(p_2) = V$, $v(p_3) = V$ rend φ vraie, $[\![\varphi]\!]_v = V$, c'est-à-dire $v \in \text{Modele}(\varphi)$.

Remarque. Souvent, en plus de dire si oui ou non une formule φ est satisfiable, on va chercher à **déterminer toutes les valuations v qui rendent φ vraie**. En d'autres termes, on va chercher à déterminer l'ensemble $\text{Mod}(\varphi)$

III. 2. Utilité de SAT : exemples de réduction (ou dit aussi modélisation) SAT

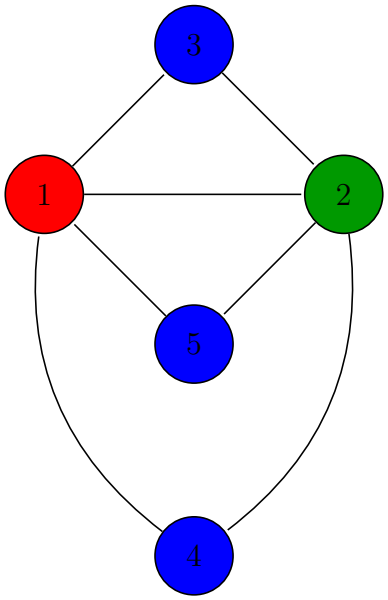
Le problème SAT est un problème qui apparaît dans de nombreux domaines, apparemment sans lien avec la théorie de la logique propositionnelle : pour résoudre des problèmes de satisfaction de contraintes, de planification, pour vérifier les propriétés d'un modèle et même en cryptographie.

III. 2. a. Un premier exemple : coloration d'un graphe

On s'intéresse au problème de la coloration d'un graphe : soit un ensemble de couleurs numérotées de 1 à C et un graphe g dont les sommets sont étiquetés de 1 à V.
On note $e = (i, j)$ une arête qui relie les sommets i et j, si elle existe, et on étiquette également toutes les arêtes existantes pour des entiers de 1 à E.
On se demande s'il est possible de colorer chaque sommet de ce graphe avec une couleur de 1 à C de sorte que deux sommets reliés par une arête n'aient jamais la même couleur.
Le problème d'optimisation classique est de trouver une coloration avec un nombre de couleurs minimum (on appelle ce nombre minimal de couleurs **nombre chromatique** associé au graphe)

Exemple 22

Cette coloration à 3 couleurs est valide :



On comprend assez intuitivement que 3 couleurs au minimum sont nécessaires pour éviter que deux nœuds reliés par une arête n'aient la même couleur. Le nombre chromatique de ce graphe est 3.

Montrons à présent le lien avec un problème SAT. Nous allons montrer que résoudre ce

problème pour un graphe quelconque revient à résoudre le problème SAT associé à une formule logique définie à partir de la structure du graphe.

On introduit les variables propositionnelles $p_{i,c}$ signifiant « Le sommet i est de couleur c », avec $i \in [\![1,\, V]\!]$ et $c \in [\![1,\, C]\!]$. Le nombre couleurs C est fixé et on se demande s'il est possible de colorier le graphe, au sens explicité ci-dessus, avec C couleurs.

Contrainte 1 : La première chose à écrire et que tout sommet (ET) doit avoir une couleur (celle-ci OU celle-la) :

$$\bigwedge_{i \in [\![1,\, V]\!]} \left(\bigvee_{c \in [\![1,\, C]\!]} p_{i,c} \right)$$

Contrainte 2 : La seconde chose à écrire et que tout sommet (ET) ne doit avoir qu'une seule couleur attribuée (NON (une couleur ET une autre)) :

$$\bigwedge_{i \in [\![1,\, V]\!]} \left(\bigwedge_{(c,c') \in [\![1,\, C]\!]^2, c \neq c'} \neg (p_{i,c} \wedge p_{i,c'}) \right)$$

Contrainte 3 : Et enfin, on écrit que deux sommets reliés par une arête ne peuvent avoir la même couleur. On écrit cela en disant que pour toutes les couleurs (ET), et pour tout couple de sommets reliés par une arête, il est impossible que ces deux sommets aient la même couleur :

$$\bigwedge_{c \in [\![1,\, C]\!]} \left(\bigwedge_{(i,j) \in [\![1,\, E]\!]} \neg (p_{i,c} \wedge p_{j,c}) \right)$$

Au final, on a **réduit** le problème de savoir si un graphe donné peut être coloré avec C couleurs à un problème de satisfiabilité d'une formule logique φ réunissant les 3 contraintes :

$$\varphi = \underbrace{\left(\bigwedge_{i \in [\![1,\, V]\!]} \left(\bigvee_{c \in [\![1,\, C]\!]} p_{i,c} \right) \right)}_{\text{contrainte 1}} \wedge \underbrace{\left(\bigwedge_{i \in [\![1,\, V]\!]} \left(\bigwedge_{(c,c') \in [\![1,\, C]\!]^2, c \neq c'} \neg (p_{i,c} \wedge p_{i,c'}) \right) \right)}_{\text{contrainte 2}} \wedge \underbrace{\left(\bigwedge_{c \in [\![1,\, C]\!]} \left(\bigwedge_{(i,j) \in [\![1,\, E]\!]} \neg (p_{i,c} \wedge p_{j,c} \right) \right)}_{\text{contrainte 3}}$$

et avec les lois de Morgan, on peut écrire :

$$\varphi = \underbrace{\left(\bigwedge_{i \in V} \left(\bigvee_{c \in [\![1,\, C]\!]} p_{i,c} \right) \right)}_{\text{contrainte 1}} \wedge \underbrace{\left(\bigwedge_{i \in [\![1,\, V]\!]} \left(\bigwedge_{(c,c') \in [\![1,\, C]\!]^2, c \neq c'} \neg p_{i,c} \vee \neg p_{i,c'} \right) \right)}_{\text{contrainte 2}} \wedge \underbrace{\left(\bigwedge_{c \in [\![1,\, C]\!]} \left(\bigwedge_{(i,j) \in [\![1,\, E]\!]} \neg p_{i,c} \vee \neg p_{j,c} \right. \right.}_{\text{contrainte 3}}$$

Cette formule équivalente est sous forme CNF.

Cette formule possède $V + V \times \dfrac{C \times (C-1)}{2} + C \times E$ clauses !

Exemple 23

Pour le petit graphe dessiné au début de cette section, on a :

- 5 sommets : $I = 5$
- 7 arêtes, $E = 7$: $e_1 = (1,2)$, $e_2 = (1,3)$, $e_3 = (1,4)$, $e_4 = (1,5)$, $e_5 = (2,3)$, $e_6 = (2,4)$, $e_7 = (2,5)$,
- 3 couleurs $C = 3$: bleu, rouge et vert, que l'on numérote dans cet ordre par exemple.

On a $I \times C = 5 \times 3 = 15$ variables propositionnelles différentes. On les numérote de 1 à 15 :

$$p_{i,c} = q_{3\times(i-1)+c}$$

La formule s'écrit :

$$\begin{aligned}
\varphi \quad &= (q_0 \vee q_1 \vee q_2) \\
\wedge \quad &(q_3 \vee q_4 \vee q_5) \\
\wedge \quad &(q_6 \vee q_7 \vee q_8) \\
\wedge \quad &(q_9 \vee q_{10} \vee q_{11}) \\
\wedge \quad &(q_{12} \vee q_{13} \vee q_{14}) \\
\\
\wedge \quad &(\neg q_0 \vee \neg q_1) \wedge (\neg q_0 \vee \neg q_2) \wedge (\neg q_1 \vee \neg q_2) \\
\wedge \quad &(\neg q_3 \vee \neg q_4) \wedge (\neg q_3 \vee \neg q_5) \wedge (\neg q_4 \vee \neg q_5) \\
\wedge \quad &(\neg q_6 \vee \neg q_7) \wedge (\neg q_6 \vee \neg q_8) \wedge (\neg q_7 \vee \neg q_8) \\
\wedge \quad &(\neg q_9 \vee \neg q_{10}) \wedge (\neg q_9 \vee \neg q_{11}) \wedge (\neg q_{10} \vee \neg q_{11}) \\
\wedge \quad &(\neg q_{12} \vee \neg q_{13}) \wedge (\neg q_{12} \vee \neg q_{14}) \wedge (\neg q_{13} \vee \neg q_{14}) \\
\\
\wedge \quad &(\neg q_0 \vee \neg q_3) \wedge (\neg q_1 \vee \neg q_4) \wedge (\neg q_2 \vee \neg q_5) \\
\wedge \quad &(\neg q_0 \vee \neg q_6) \wedge (\neg q_1 \vee \neg q_7) \wedge (\neg q_2 \vee \neg q_8) \\
\wedge \quad &(\neg q_0 \vee \neg q_9) \wedge (\neg q_1 \vee \neg q_{10}) \wedge (\neg q_2 \vee \neg q_{11}) \\
\wedge \quad &(\neg q_0 \vee \neg q_{12}) \wedge (\neg q_1 \vee \neg q_{13}) \wedge (\neg q_2 \vee \neg q_{14}) \\
\wedge \quad &(\neg q_3 \vee \neg q_6) \wedge (\neg q_4 \vee \neg q_7) \wedge (\neg q_5 \vee \neg q_8) \\
\wedge \quad &(\neg q_3 \vee \neg q_9) \wedge (\neg q_4 \vee \neg q_{10}) \wedge (\neg q_5 \vee \neg q_{11}) \\
\wedge \quad &(\neg q_3 \vee \neg q_{12}) \wedge (\neg q_4 \vee \neg q_{13}) \wedge (\neg q_5 \vee \neg q_{14})
\end{aligned}$$

C'est une CNF qui possède bien $V + V \times \dfrac{C \times (C-1)}{2} + C \times E = 5 + 5 \times \dfrac{3 \times 2}{2} + 3 \times 7 = 5 + 15 + 21 = 41$ clauses...

Remarque. Déterminer ce nombre chromatique pour n'importe quel graphe est un problème très difficile. Il a même été montré en 1979 que décider si un certain nombre C de couleurs suffit pour colorier un graphe quelconque est un problème NP-complet ! Toutefois, en pratique, il existe des méthodes pour le résoudre : algorithmes gloutons, résolutions SAT astucieuses...etc

III. 2. b. Un second exemple : résolution d'un Sudoku

Exercice 4.
Proposez une modélisation SAT du problème de résolution d'un Sudoku 9×9.
Évaluer le nombre de clauses disjonctives de la formule obtenue.
Généraliser ce dénombrement pour un Sudoku de taille $n \times n$ où $n = d^2$

Corrigé de l'exercice 2. *[Retour à l'énoncé]*

On introduit $9 \times 9 \times 9$ variables propositionnelles : $p_{i,j,c}$: « La case (i, j) contient le chiffre c »

— C_0 Certaines cases ont une valeur déjà attribuée
— C_1 Toute case a un nombre entre 1 et 9
— C_2 Une case ne peut pas avoir 2 nombres associés
— C_3 Sur chaque ligne, chaque nombre est présent une fois
— C_4 Sur chaque colonne, chaque nombre est présent une fois
— C_5 Dans chaque carré, chaque nombre est présent une fois

III. 3. Un problème très coûteux à résoudre !

Définition 27 (Solveur SAT)

On appelle solveur SAT (*SAT solver*) un programme informatique capable de résoudre ce problème pour n'importe quelle formule de logique propositionnelle φ.

III. 3. a. Méthode exhaustive naïve

Une première méthode naïve de résolution SAT consiste à évaluer la formule φ sur les 2^n valuations possibles et à :

— s'arrêter dès que l'on trouver une valuation modèle
— ou bien continuer à tester toutes les valuations possibles pour renvoyer l'ensemble Modèle(φ) complet.

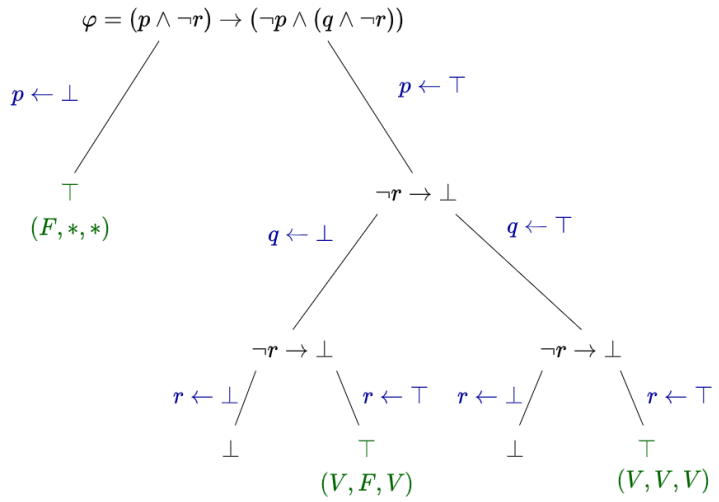
Dans la première version minimaliste, l'algorithme peut être très rapide si on a un gros coup de chance, mais il peut être exponentiel (en fait pire que ça, car le coût d'une évaluation n'est certainement pas constant !) si seule la dernière valuation est une modèle.
Dans la deuxième version, le coût est systématiquement au moins exponentiel car toutes les valuations doivent être testées.
Dans les deux cas, ces méthodes sont inenvisageables dès que l'on a plus d'une centaine de variables propositionnelles

III. 3. b. Algorithme de Quine

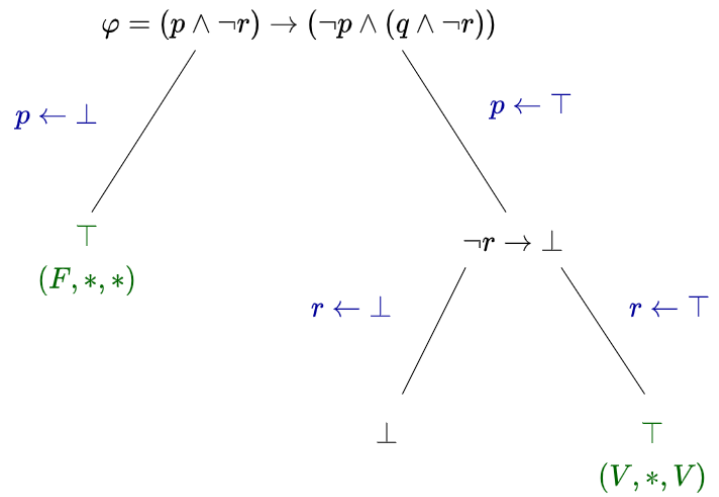
L'algorithme de Quine est un algorithme de résolution du problème SAT général qui procède en construisant un arbre de décision.

A chaque étape, on substitue l'une des variables propositionnelles par $\top$ (branche de gauche) et $\bot$ (branche de droite) et on regarde la formule équivalente obtenue, qui correspond à une évaluation partielle de la formule de départ.
Si on dessine l'arbre dans son ensemble, cela revient à la méthode exhaustive car on a dessiné 2^n branches et effectué 2^n évaluations. Tout l'intérêt de l'algorithme de Quine réside dans l'espérance que l'on n'aura pas à explorer toutes les branches de l'arbre. En effet, si, à un nœud de l'arbre, on constate que l'évaluation partielle est déjà fausse, alors, il est inutile de poursuivre sur cette branche. A l'inverse, si l'évaluation partielle est déjà vraie, on a trouvé un ensemble de valuations modèles et la formule est déclarée satisfiable. L'algorithme reste toutefois, dans le pire des cas, au moins en 2^n, donc exponentiel ! Tout son intérêt réside dans le fait que l'on évite certaines évaluations inutiles. Par exemple, dans le cas où j'aboutis à une formule très compliqué qui commence par $\bot \to \ldots$, inutile de m'intéresser à la partie droite de l'implication, je suis certain que toutes les valuations qui correspondent aux sous-arbres de mon arbre de décision sont des valuations modèles.
Illustrons l'algorithme de Quine en le déroulant pas à pas sur un exemple : on cherche à résoudre $SAT(\varphi)$ où ϕ est définie par :

$$\varphi = (p \wedge \neg r) \to (\neg p \wedge \neg(q \wedge \neg r))$$



On obtient bien 6 valuations modèles, ce qui est cohérent avec la table de vérité :
Remarquons que, en effectuant les substitutions dans un ordre différent, l'algorithme termine plus rapidement :

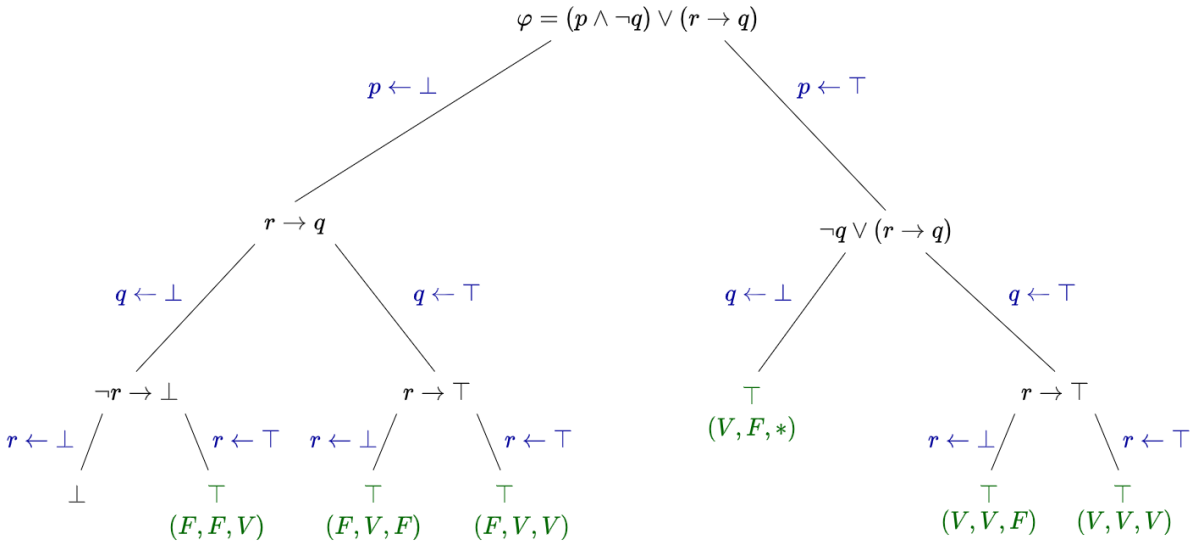


Plusieurs solvers SAT, dont DPLL, utilisent cette idée sous forme d'heuristique pour choisir intelligemment l'ordre dans lequel les substitutions sont effectuées, le but étant de terminer le plus rapidement possible certains embranchements.

Exercice 5.
Dérouler l'algorithme de Quine

$$\varphi = (p \wedge \neg q) \vee (r \to q)$$

Corrigé de l'exercice 3. *[Retour à l'énoncé]*



III. 3. c. NP-complétude du problème SAT

Le problème SAT est un problème assez étonnant :

- Il est très simple à énoncer
- Il est très facile de vérifier si une valuation est un modèle, cela se fait avec un coût temporel raisonnable (polynomial)
- On a du mal à trouver des algorithmes déterministes qui ne sont pas exponentiels ou même pire

Ceci est caractéristique des problèmes dits NP-complets que vous étudierez en deuxième année.

Théorème 2 (Théorème de Cook-Levin (1971))

Le problème SAT est NP-complet

Démonstration

Programme de MPI, séquence sur les classes de complexité.

III. 3. d. État des lieux des solveurs SAT en 2023

Ainsi, le problème SAT, pris dans toute sa généralité, est un problème extrêmement difficile à résoudre : on ne connaît que des algorithmes dont la complexité dans le pire des cas est exponentielle. Toutefois, en pratique, des progrès considérables ont été faits, notamment sur des problèmes un peu moins généraux que SAT

Définition 28 (Problème n-SAT)

On appelle problème n-SAT une restriction du problème SAT aux formules logiques déjà écrites sous forme CNF et dont les clauses comportent au plus n littéraux.

Il existe aujourd'hui des réponses pratiques satisfaisantes sur ces formulations un peu plus restrictive, parmi lesquelles

Le problème 2-SAT est la restriction du problème SAT aux formes normales conjonctives (CNF) avec au plus 2 littéraux par clause. Une instance de ce problème est, par exemple :

$$\varphi = (p_1 \vee p_3) \wedge (\neg p_1 \vee p_4) \wedge (p_1 \vee \neg p_2)$$

Ce problème est à part car il existe un algorithme **de complexité polynomiale** qui résout totalement ce problème à l'aide de la théorie des graphes. Nous le coderons en TP !

Le problème CNF-SAT est la restriction du problème SAT aux formes normales conjonctives. Beaucoup de solveurs se restreignent aux formules logiques écrites sous forme CNF, par ce solver DPLL [1], version améliorée de l'algorithme de Quine, disponible en ligne :

DPLL SAT solver `https://www.inf.ufpr.br/dpasqualin/d3-dpll/`

qui utilise un format de définition de l'instance φ du problème proche du format DIMACS vu plus haut. Cet algorithme est une amélioration de l'algorithme de Quine utilisant des heuristiques pour optimiser le choix de la variable propositionnelle traitée à chaque étape.

1. Davis–Putnam–Logemann–Loveland (DPLL)

Exemple 24

On va essayer de faire résoudre ce problème par un solver SAT : on renumérote de manière linéaire les variables propositionnelle : $p_{i,c} = q_{3\times(i-1)+c}$

$$
\begin{aligned}
\varphi \quad & = (q_1 \vee q_2 \vee q_3) \\
\wedge \quad & (q_4 \vee q_5 \vee q_6) \\
\wedge \quad & (q_7 \vee q_8 \vee q_9) \\
\wedge \quad & (q_{10} \vee q_{11} \vee q_{12}) \\
\wedge \quad & (q_{13} \vee q_{14} \vee q_{15})
\end{aligned}
$$

$$
\begin{aligned}
\wedge \quad & (\neg q_1 \vee \neg q_2) \wedge (\neg q_1 \vee \neg q_3) \wedge (\neg q_2 \vee \neg q_3) \\
\wedge \quad & (\neg q_4 \vee \neg q_5) \wedge (\neg q_4 \vee \neg q_6) \wedge (\neg q_5 \vee \neg q_6) \\
\wedge \quad & (\neg q_7 \vee \neg q_8) \wedge (\neg q_7 \vee \neg q_9) \wedge (\neg q_8 \vee \neg q_9) \\
\wedge \quad & (\neg q_{10} \vee \neg q_{11}) \wedge (\neg q_{10} \vee \neg q_{12}) \wedge (\neg q_{11} \vee \neg q_{12}) \\
\wedge \quad & (\neg q_{13} \vee \neg q_{14}) \wedge (\neg q_{13} \vee \neg q_{15}) \wedge (\neg q_{14} \vee \neg q_{15})
\end{aligned}
$$

$$
\begin{aligned}
\wedge \quad & (\neg q_1 \vee \neg q_4) \wedge (\neg q_2 \vee \neg q_5) \wedge (\neg q_3 \vee \neg q_6) \\
\wedge \quad & (\neg q_1 \vee \neg q_7) \wedge (\neg q_2 \vee \neg q_8) \wedge (\neg q_3 \vee \neg q_9) \\
\wedge \quad & (\neg q_1 \vee \neg q_{10}) \wedge (\neg q_2 \vee \neg q_{11}) \wedge (\neg q_3 \vee \neg q_{12}) \\
\wedge \quad & (\neg q_1 \vee \neg q_{13}) \wedge (\neg q_2 \vee \neg q_{14}) \wedge (\neg q_3 \vee \neg q_{15}) \\
\wedge \quad & (\neg q_4 \vee \neg q_7) \wedge (\neg q_5 \vee \neg q_8) \wedge (\neg q_6 \vee \neg q_9) \\
\wedge \quad & (\neg q_4 \vee \neg q_{10}) \wedge (\neg q_5 \vee \neg q_{11}) \wedge (\neg q_6 \vee \neg q_{12}) \\
\wedge \quad & (\neg q_4 \vee \neg q_{13}) \wedge (\neg q_5 \vee \neg q_{14}) \wedge (\neg q_6 \vee \neg q_{15})
\end{aligned}
$$

Au format DIMACS (CNF), cela donne :

```
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
-1 -2
-1 -3
-2 -3
-4 -5
-4 -6
-5 -6
-7 -8
-7 -9
-8 -9
-10 -11
-10 -12
-11 -12
-13 -14
-13 -15
-14 -15
-1 -4
-2 -5
-3 -6
-1 -7
-2 -8
-3 -9
-1 -10
-2 -11
-3 -12
-1 -13
-2 -14
-3 -15
-4 -7
-5 -8
-6 -9
-4 -10
-5 -11
-6 -11
-4 -13
-5 -14
-6 -14
```

Le solver DPLL nous répond qu'il existe une solution à ce problème SAT :

```
SATISFIABLE 1 5 9 12 15 -2 -3 -4 -7 -10 -13 -6 -8 -11 -14
```

Cela signifie que :

Pour le sommet 1 : 1 -2 -3, ce qui signifie qu'on lui attribue la 1ere couleur

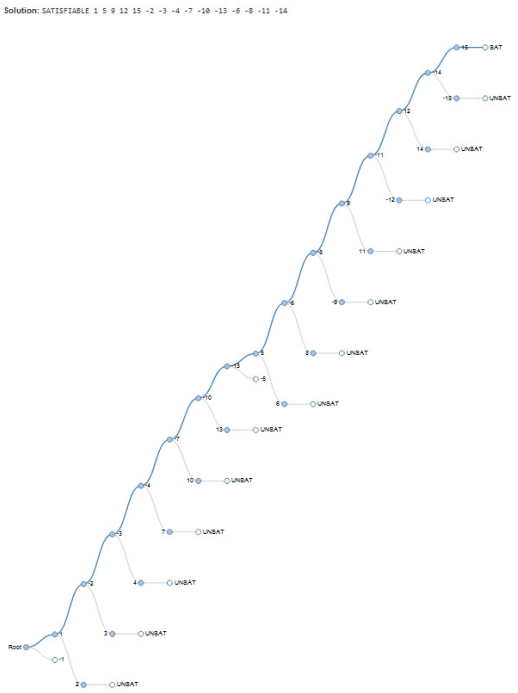
Pour le sommet 2 : -4 5 -6, ce qui signifie qu'on lui attribue la 2eme couleur

Pour le sommet 3 : -7 -8 9, ce qui signifie qu'on lui attribue la 3eme couleur

Pour le sommet 4 : -10 -11 12, ce qui signifie qu'on lui attribue la 3eme couleur

Pour le sommet 5 : -13 -14 15, ce qui signifie qu'on lui attribue la 3eme couleur

C'est exactement le solution de coloriage que nous avons trouvée !
On nous donne même l'arbre de recherche utilisé !



Bien sûr, pour des graphes plus grands, il faut un solveur SAT très performant !

Remarque. Il existe même des compétitions internationales de programmation qui évaluent et récompensent les meilleurs propositions pour résoudre ce problème :

`http://www.satcompetition.org/`