INFO-F-302, Cours d'Informatique Fondamentale

Emmanuel Filiot
Département d'Informatique
Faculté des Sciences
Université Libre de Bruxelles

Année académique 2022-2023

Le Problème SAT

3 Rappels

Satisfaisabilité

Soit P un ensemble de propositions. Une formule ϕ de la logique propositionnelle sur P est *satisfaisable* si

3 Rappels

Satisfaisabilité

Soit P un ensemble de propositions. Une formule ϕ de la logique propositionnelle sur P est *satisfaisable* si il existe une interprétation $V : X \to \{1, 0\}$ telle que $V \models \phi$.

3 Rappels

Satisfaisabilité

Soit P un ensemble de propositions. Une formule ϕ de la logique propositionnelle sur P est *satisfaisable* si il existe une interprétation $V : X \to \{1, 0\}$ telle que $V \models \phi$.

Dans ce chapitre, nous allons utiliser des solveurs SAT, qui permettent de tester la satisfiabilité de formules représentées comme des **ensembles de clauses**.

4 Littéraux et Clauses

Définition

- Un littéral est une variable x ou sa négation $\neg x$
- Une clause est une disjonction de littéraux $\ell_1 \vee \cdots \vee \ell_n$. Elle est satisfaite par une valuation V s'il existe i tel que $V(\ell_i) = 1$. Par extension, on appelle $\perp$ la *clause vide*, qui est insatisfaisable.

Donc, p et $\neg p$ sont des littéraux. $x \vee \neg y \vee r$ est une clause, mais pas $x \wedge y$.

4 Littéraux et Clauses

Définition

- Un littéral est une variable x ou sa négation $\neg x$
- Une clause est une disjonction de littéraux $\ell_1 \vee \cdots \vee \ell_n$. Elle est satisfaite par une valuation V s'il existe i tel que $V(\ell_i) = 1$. Par extension, on appelle $\perp$ la *clause vide*, qui est insatisfaisable.

Donc, p et $\neg p$ sont des littéraux. $x \vee \neg y \vee r$ est une clause, mais pas $x \wedge y$.

Satisfaction d'ensemble de clauses

Un ensemble de clauses $A = \{C_1, \ldots, C_n\}$ est satisfait par une valuation V, noté $V \models A$, si pour tout i, $V \models C_i$. En particulier, tout valuation satisfait l'ensemble vide $A = \varnothing$.

$A \models C_1 \wedge C_2 \wedge \ldots \wedge C_n$

Lien avec les formes normales

Une formule est en *forme normale conjonctive* (FNC) ssi c'est une conjonction de disjonctions de littéraux, de la forme : de la forme :

$$\bigwedge_i (\bigvee_j (\neg) x_{i,j})$$

Une formule est en *forme normale disjonctive* (FND) ssi c'est une disjonction de conjonctions de littéraux, de la forme : de la forme :

$$\bigvee_i (\bigwedge_j (\neg) x_{i,j})$$

Lien avec les formes normales

Une formule est en *forme normale conjonctive* (FNC) ssi c'est une conjonction de disjonctions de littéraux, de la forme : de la forme :

$$\bigwedge_i (\bigvee_j (\neg) x_{i,j})$$

Une formule est en *forme normale disjonctive* (FND) ssi c'est une disjonction de conjonctions de littéraux, de la forme : de la forme :

$$\bigvee_i (\bigwedge_j (\neg) x_{i,j})$$

Lemme

Tout ensemble non-vide de clauses $A = \{C_1, \ldots, C_n\}$ est équivalent à la formule en FNC $\phi_A = \bigwedge_{i=1}^{n} C_i$, au sens où pour toute valuation, $V \models A$ ssi $V \models \phi_A$.

$\Rightarrow$ Pour mettre une formule sous forme de clauses, il suffit de la mettre en FNC !

6 Mise sous FNC et FND

On utilise les transformations successives suivantes pour obtenir les formes normales :

1. élimination des connecteurs $\rightarrow$ et $\leftrightarrow$ grâce aux équivalences suivantes :

$$(\phi \rightarrow \psi) \equiv (\neg\phi \vee \psi)$$
$$(\phi \leftrightarrow \psi) \equiv (\neg\phi \vee \psi) \wedge (\phi \vee \neg\psi)$$

2. entrer les négations le plus à l'intérieur possible :

$$\neg(\phi \wedge \psi) \equiv (\neg\phi \vee \neg\psi) \qquad \neg\neg\phi \equiv \phi$$
$$\neg(\phi \vee \psi) \equiv (\neg\phi \wedge \neg\psi)$$

3. utilisation des distributivité de $\vee$ et $\wedge$:

$$(\phi \vee (\psi \wedge \chi)) \equiv (\phi \vee \psi) \wedge (\phi \vee \chi) \text{ (pour mise sous FNC)}$$
$$(\phi \wedge (\psi \vee \chi)) \equiv (\phi \wedge \psi) \vee (\phi \wedge \chi) \text{ (pour mise sous FND)}$$

Mise en FNC de $\neg(x \leftrightarrow (y \rightarrow r))$

$$\equiv \neg([x \rightarrow (y \rightarrow r)] \wedge [(y \rightarrow r) \rightarrow x])$$

$$\equiv \neg([\neg x \vee \neg y \vee r] \wedge [(y \wedge \neg r) \vee x])$$

$$\equiv \neg[\neg x \vee \neg y \vee r] \vee \neg[(y \wedge \neg r) \vee x]$$

$$\equiv \underbrace{x \wedge y \wedge \neg r}_{F_1} \vee (\underbrace{(\neg y \vee r)}_{F_2} \wedge \underbrace{\neg x}_{F_3})$$

$$\equiv (F_1 \vee F_2) \wedge (F_1 \vee F_3)$$

$$\equiv ((x \wedge y \wedge \neg r) \vee (\neg y \vee r)) \wedge ((x \wedge y \wedge \neg r) \vee \neg x)$$

$$\equiv \ldots$$

Mise en FNC de $\neg(x \leftrightarrow (y \rightarrow r))$

1. on retire les équivalences et implications :

$$\neg((\neg x \vee \neg y \vee r) \wedge (\neg(\neg y \vee r) \vee x))$$

2. on pousse les négations à l'intérieur :

$$(x \wedge y \wedge \neg r) \vee ((\neg y \vee r) \wedge \neg x)$$

3. on distribue :

$$((x \wedge y \wedge \neg r) \vee (\neg y \vee r)) \wedge ((x \wedge y \wedge \neg r) \vee \neg x)$$

4. et encore :

$$(x \vee \neg y \vee r) \wedge (y \vee \neg y \vee r) \wedge (\neg r \vee \neg y \vee r) \wedge (x \vee \neg x) \wedge (y \vee \neg x) \wedge (\neg r \vee \neg x)$$

5. on retire les formules équivalentes à $\top$:

$$(x \vee \neg y \vee r) \wedge (y \vee \neg x) \wedge (\neg r \vee \neg x)$$

9 Exercice

Mettre les formules suivantes sous FNC :

1. $\neg(x \rightarrow q)$
2. $(x \wedge q) \vee r$
3. $x \wedge (y \vee (r \wedge t))$
4. $\neg(x \wedge ((r \vee q) \wedge s))$
5. $\neg(x \wedge (r \leftrightarrow q))$
6. $x \leftrightarrow (y \vee r)$
7. $(x_1 \wedge q_1) \vee (x_2 \wedge q_2) \vee (x_3 \wedge q_3)$
8. généraliser aux formules de la forme
   $(x_1 \wedge q_1) \vee ((x_2 \wedge q_2) \vee \cdots \vee (x_n \wedge q_n))$ pour tout $n \geq 0$

$$\equiv \left[x_1 \vee ((x_2 \wedge q_2) \vee \cdots \vee (x_n \wedge q_n)) \right] \wedge \cdots$$

$$\left[q_1 \vee ((x_2 \wedge q_2) \vee \cdots \vee (x_n \wedge q_n)) \right]$$

10 Question 1

Pensez-vous qu'il existe un algorithme de complexité pire cas en temps polynomiale pour tester la satisfaisabilité d'une formule en forme normale disjonctive ?

10 Question 1

Pensez-vous qu'il existe un algorithme de complexité pire cas en temps polynomiale pour tester la satisfaisabilité d'une formule en forme normale disjonctive ? OUI :

```
FND_SOLVE(Phi : formule en forme normale disjonctive)

    POUR TOUTE conjonction C de Phi
       SI C ne contient pas de litéraux contradictoires
           RETOURNER 'SATISFAISABLE'

    RETOURNER 'INSATISFAISABLE'
```

11 Question 2

Qu'est-ce qui n'est pas correct dans l'énoncé suivant ?

L'algorithme suivant permet de décider en temps polynomial si une formule donnée en entrée est satisfaisable :

```
SAT\_SOLVE(Phi : formule)

    Psi <- forme normale disjonctive équivalente à Phi

    RETOURNER FND\_SOLVE(Psi)
```

11 Question 2

Qu'est-ce qui n'est pas correct dans l'énoncé suivant ?

L'algorithme suivant permet de décider en temps polynomial si une formule donnée en entrée est satisfaisable :

```
SAT\_SOLVE(Phi : formule)

    Psi <- forme normale disjonctive équivalente à Phi

    RETOURNER FND\_SOLVE(Psi)
```

Ce n'est pas correct car la mise sous forme normale disjonctive crée, au pire, une formule exponentiellement plus grande. Par exemple :

$$(a_1 \vee b_1) \wedge (a_2 \vee b_2) \wedge \cdots \wedge (a_n \vee b_n)$$

12 Problème SAT

Définition

ENTREE Un ensemble de clauses S.
SORTIE Est-ce que S est **sat**isfaisable ?

Dans le cas où S est satisfaisable, on aimerait que l'algorithme nous retourne une valuation V qui satisfait S.

13 Remarques sur la complexité du problème SAT

- le problème de satisfaisabilité d'une formule propositionnelle, appelé *problème SAT*, est dans la classe NP (on y reviendra dans le chapitre "Complexité")
- actuellement, on ne sait toujours pas s'il existe un algorithme pour ce problème dont la complexité en temps est polynomiale.
- la plupart des scientifiques pensent que ce n'est pas le cas, mais personne n'a été capable de le montrer.
- ce défit scientifique est plus connu sous la question $P \neq NP$: est-ce qu'il existe un problème résoluble en temps non-déterministe polynomial qui n'est pas résoluble en temps polynomial ?
- le premier qui aura la réponse (et la preuve de son affirmation) gagnera 1.000.000$ [1]

1. Millenium Prize Problems du Clay Mathematics Institute http://www.claymath.org/millennium/

14 Pourquoi SAT est-il si intéressant ?

- beaucoup de problèmes pratiques intéressants se trouvent dans la classe NP, voir http://www.nada.kth.se/~viggo/problemlist/.
- par exemple le problème du *voyageur de commerce* : étant donné une distance d et n villes, est-il possible de construire un cycle de longueur au plus d qui passe par les les n villes exactement une fois (sauf la ville de départ). **Application :** optimiser la trajectoire d'une fraiseuse, d'un bus scolaire, d'un camion de livraison, etc ...
- autre exemple, le *bin-packing* : étant donné k boîtes de contenance différentes, n objets de volumes différents, et $d \geq 0$ un entier, est-il possible de ranger les n objets en utilisant au plus d boîtes. **Application :** rangement de fichiers sur un support informatique, remplissage de camions ou containers ...

15 Pourquoi SAT est-il si intéressant ?

- **tous** les problèmes de la classe NP se *réduisent* au problème SAT en temps polynomial. En d'autre terme, pour résoudre une instance I du problème du voyageur de commerce ou du bin-packing, il est possible de construire, en temps polynomial, une formule de la logique propositionnelle ϕ_I telle que ϕ_I est satisfaisable ssi I a une solution
- il est donc important d'avoir de "bons" algorithmes pour SAT

16 Solveurs SAT

- Un solveur SAT est un programme qui décide le problème SAT.
- Si la formule est satisfaisable, une interprétation qui la satisfait est retournée.
- On ne connaît pas d'algorithme en temps polynomial pour résoudre SAT, et on pense qu'il n'y en a pas (SAT est NP-complet). Les SAT solveurs essaient néanmoins de se comporter efficacement dans la plupart des cas. Cependant, les SAT solveurs ont une complexité **au pire cas** exponentielle. Il est donc possible de ne pas pouvoir prouver la satisfiabilité de certaines formules avec un SAT solveur, dans l'état actuel des capacités calculatoires des ordinateurs.

17 Motivations

- Beaucoup de problèmes s'expriment naturellement par des formules en FNC.
- Par exemple, un problème de planification où l'on a beaucoup de contraintes (une conjonction) et chaque contrainte représente plusieurs alternatives possibles (disjonction).
- Les problèmes de la classe NP se réduisent tous au problème SAT en temps polynomial.
- **Motivation principale :** Plutôt que d'avoir un algorithme pour chaque problème, écrire un bon algorithme pour SAT et transformer automatiquement chaque instance du problème de départ en une instance de SAT équivalente.
- Si la transformation (aussi appelée réduction) se fait dans un temps raisonnable (idéalement polynomial), alors l'utilisation d'un solveur SAT donne parfois d'aussi bons résultats qu'un algorithme dédié au problème de départ.
- la difficulté est en général de trouver cette réduction, c'est-à-dire de *modéliser* le problème de départ comme une instance de SAT, sous forme normale conjonctive.

18 Histoire des Solveurs SAT

- **Davis & Putnam, 1960 ; Davis, Longemann & Loveland, 1962** Algorithme **DPLL**.
- **Tseitin, 1966** DPLL a une complexité en temps exponentiel au pire cas
- **Cook 1971** SAT est NP-complet
- Seconde Génération de solveurs SAT basés sur DPLL : Posit [1995], SATO [1997], GRASP [1999].
- Compétition SAT depuis 2002 : http://www.satcompetition.org/
- Solveurs SAT très efficaces : Chaff [2001], BerkMin [2002], zChaff [2004], MiniSAT [2005] ... Regroupent différentes techniques d'optimisation telles que l'apprentissage, des structures de données efficaces pour stocker les clauses, des heuristiques etc ...

19 Notations pour les grandes disjonctions et conjonctions

Définition

Disjonction et conjonction

- On appelera *disjonction* une formule de la forme $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_n$, où $n \geq 1$. Attention lorsque $n = 1$, la disjonction est réduite à la seule formule ϕ_1. On utilisera l'abbréviation

$$\bigvee_{i=1}^{n} \phi_i$$

  pour noter cette discjontion.
- On appelera *conjonction* une formule de la forme $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$, où $n \geq 1$. Attention lorsque $n = 1$, la conjonction est réduite à la seule formule ϕ_1. On utilisera l'abbréviation

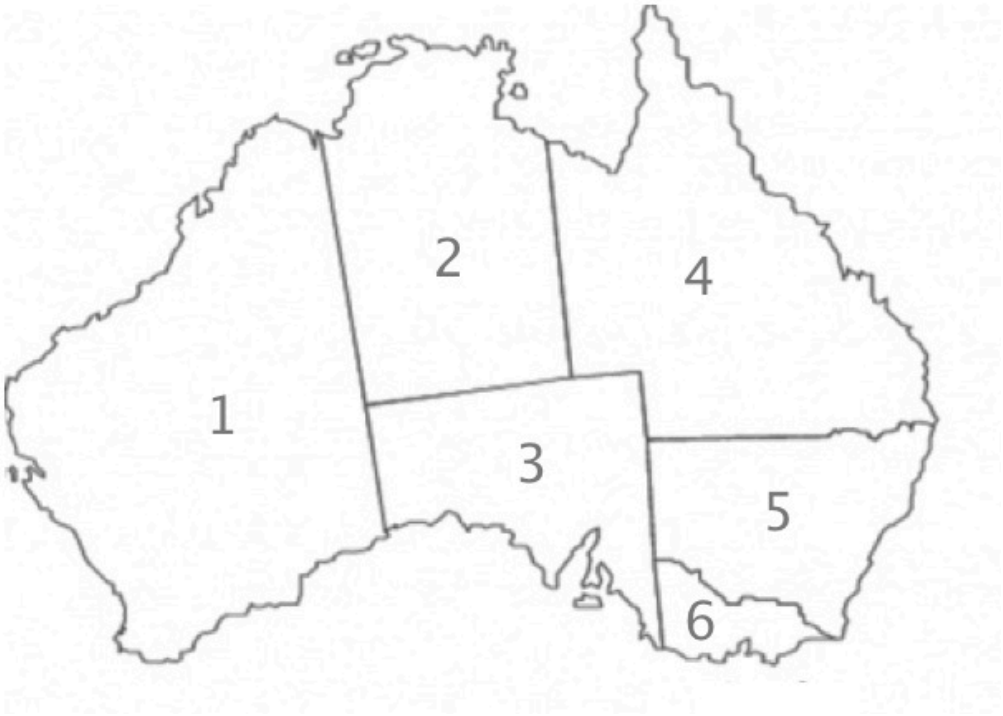
$$\bigwedge_{i=1}^{n} \phi_i$$

20 Modélisation

Nous allons maintenant nous intéresser à la modélisation de plusieurs problèmes en logique propositionnelle :

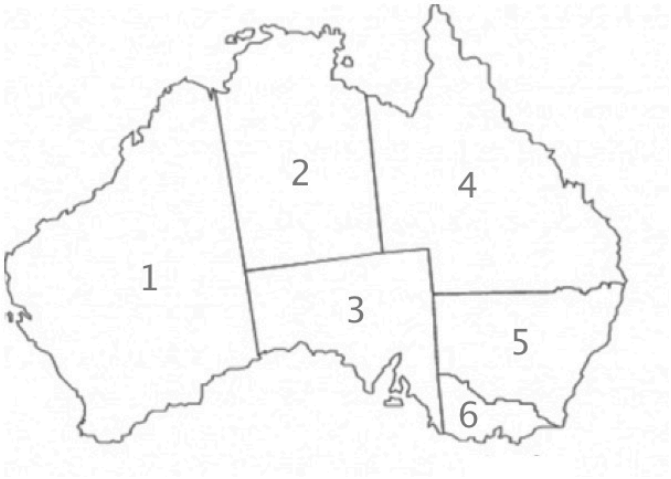
- le coloriage de cartes
- le problème des 8 reines
- le jeu Sudoku

21 Coloriage de Cartes



- colorier chaque région de telle sorte que deux régions ayant une frontière commune n'aient jamais la même couleur
- but : minimiser le nombre de couleurs

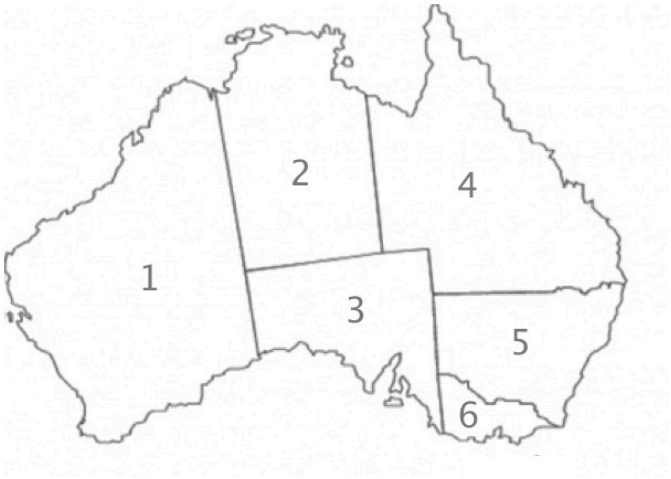
22 Coloriage de Cartes - Modélisation



Questions

- combien faut-il de couleur minimum pour la carte du dessus ?

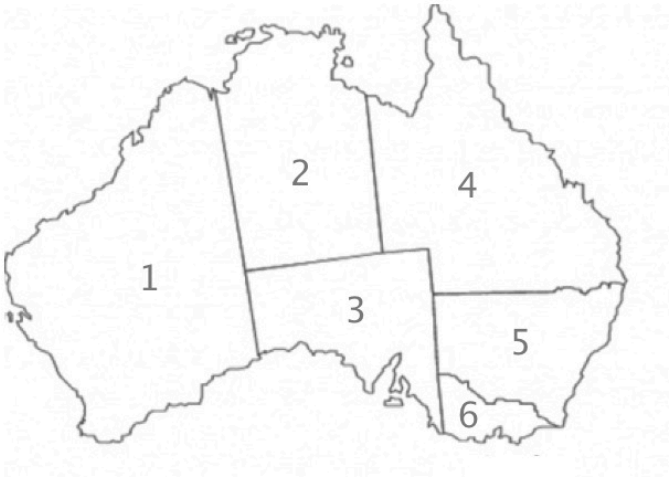
22 Coloriage de Cartes - Modélisation



Questions

- combien faut-il de couleur minimum pour la carte du dessus ? 3 couleurs

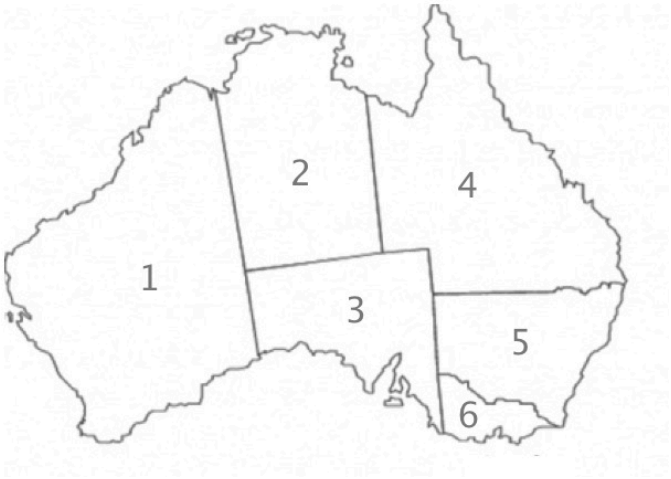
22 Coloriage de Cartes - Modélisation



Questions

- combien faut-il de couleur minimum pour la carte du dessus ? 3 couleurs
- pourriez-vous donner une carte où il faut au moins 4 couleurs ?

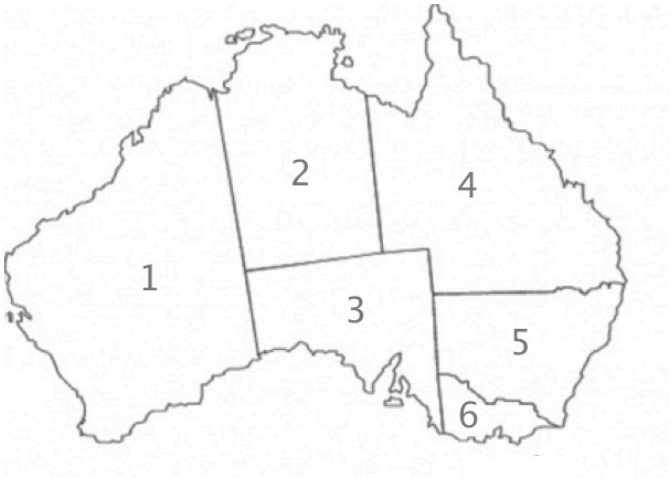
22 Coloriage de Cartes - Modélisation



Questions

- combien faut-il de couleur minimum pour la carte du dessus ? 3 couleurs
- pourriez-vous donner une carte où il faut au moins 4 couleurs ?
- et 5 couleurs ?

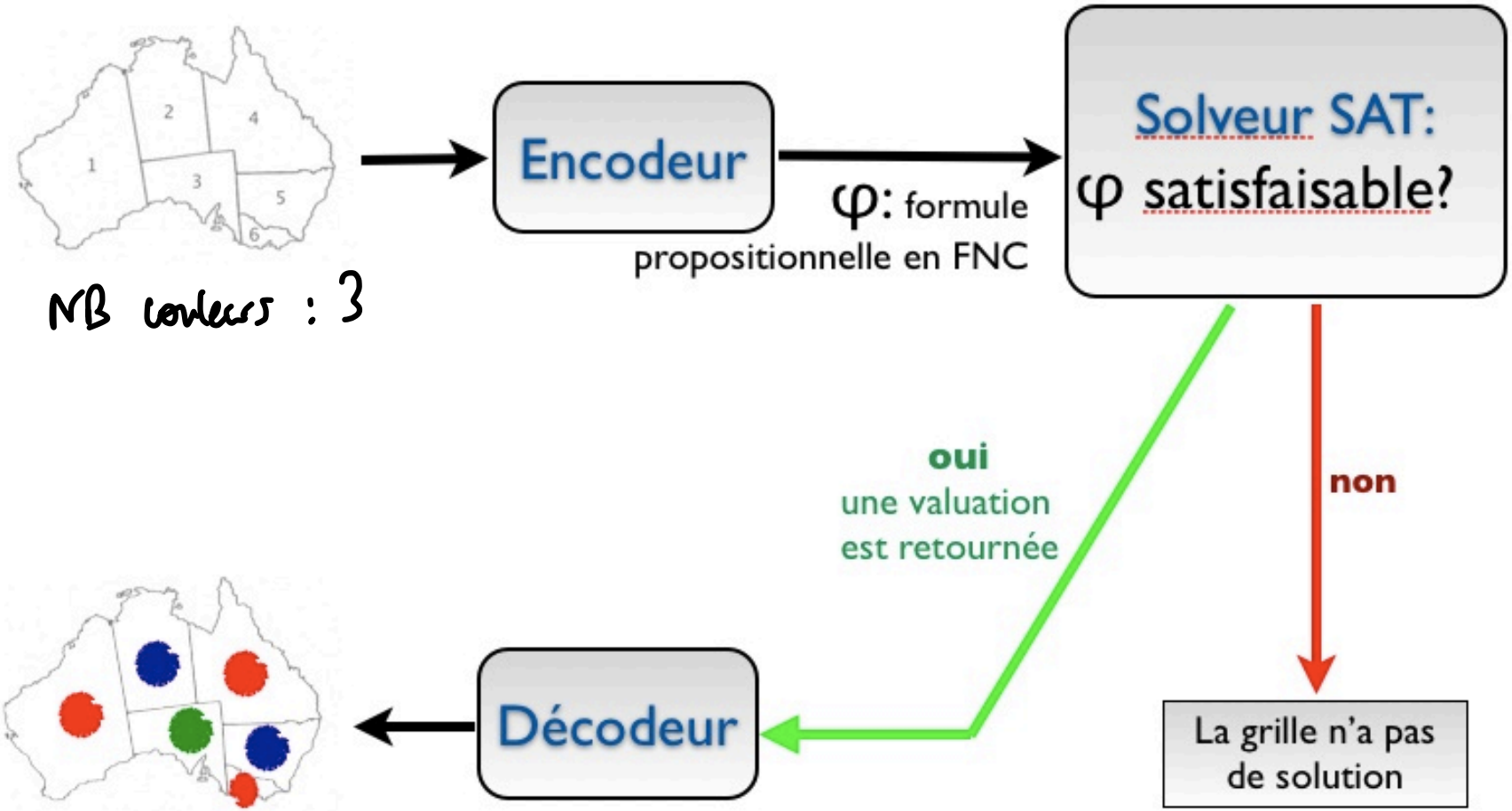
22 Coloriage de Cartes - Modélisation



Questions

- combien faut-il de couleur minimum pour la carte du dessus ? 3 couleurs
- pourriez-vous donner une carte où il faut au moins 4 couleurs ?
- et 5 couleurs ?
- **Remarque :** lorsque les régions ne sont pas discontinues (i.e. qu'une région est nécessairement en un seul morceau), alors 4 couleurs suffisent toujours (*Théorème de 4 couleurs*)

Modélisation : Rappel



Coloriage de Cartes - Modélisation

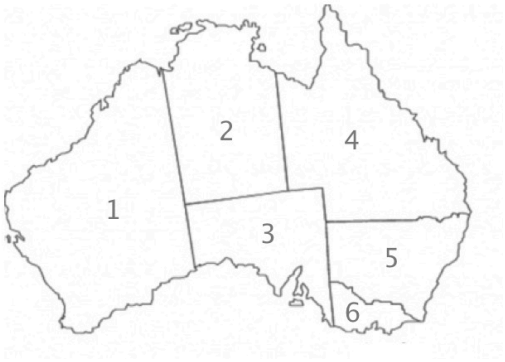
Choix des Propositions

- fixons un nombre de couleurs (3) et exprimons l'existence d'un coloriage avec 3 couleurs en logique propositionnelle.
- la première étape consiste à définir l'ensemble des propositions et leur sémantique
- ici, l'ensemble des régions est $\{1, 2, 3, 4, 5, 6\}$ et l'ensemble des couleurs est $\{R, V, B\}$. Pour chaque région r et chaque couleur c, nous introduisons la proposition $x_{r,c}$. Soit

$$X = \{x_{r,c} \mid r \in \{1, \ldots, 6\},\ c \in \{R, V, B\}\}$$

- Nous posons la sémantique suivante : si l'interprétation V trouvée par le solveur SAT est telle que $V(x_{r,c}) = 1$, alors cela signifie que la région r est coloriée avec la couleur c

Coloriage de Cartes - Modélisation



Expression des contraintes

- les régions voisines doivent avoir des couleurs différentes, ici les régions voisines sont données par l'ensemble des paires $Voisines = \{(1,2),(1,3),(2,4),(3,4),(4,5),(3,5),(5,6),(3,6)\}$.

Coloriage de Cartes - Modélisation

Expression des contraintes

- les régions voisines doivent avoir des couleurs différentes, ici les régions voisines sont données par l'ensemble des paires $Voisines = \{(1,2),(1,3),(2,4),(3,4),(4,5),(3,5),(5,6),(3,6)\}$.
- comment exprimer que 1 et 2 ont des couleurs différentes ?

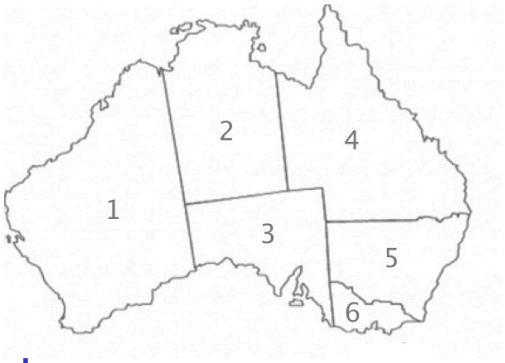
Coloriage de Cartes - Modélisation

Expression des contraintes

- les régions voisines doivent avoir des couleurs différentes, ici les régions voisines sont données par l'ensemble des paires $Voisines = \{(1,2),(1,3),(2,4),(3,4),(4,5),(3,5),(5,6),(3,6)\}$.
- comment exprimer que 1 et 2 ont des couleurs différentes ?

$$\begin{aligned} & \neg((x_{1,V} \wedge x_{2,V}) \vee (x_{1,B} \wedge x_{2,B}) \vee (x_{1,R} \wedge x_{2,R}) \\ \equiv\ & (\neg x_{1,V} \vee \neg x_{2,V}) \wedge (\neg x_{1,B} \vee \neg x_{2,B}) \wedge (\neg x_{1,R} \vee \neg x_{2,R}) \\ \equiv\ & \bigwedge_{c \in \{R,V,B\}} \neg x_{1,c} \vee \neg x_{2,c} \end{aligned}$$

Coloriage de Cartes - Modélisation

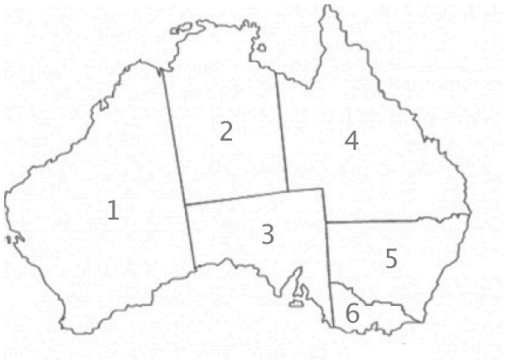


Expression des contraintes

- Il faut exprimer la contrainte pour toutes les paires voisines de $Voisines = \{(1,2),(1,3),(2,4),(3,4),(4,5),(3,5),(5,6),(3,6)\}$:

$$
\begin{array}{ll}
 & (\neg x_{1,V} \vee \neg x_{2,V}) \wedge (\neg x_{1,B} \vee \neg x_{2,B}) \wedge (\neg x_{1,R} \vee \neg x_{2,R}) \\
\wedge & (\neg x_{1,V} \vee \neg x_{3,V}) \wedge (\neg x_{1,B} \vee \neg x_{3,B}) \wedge (\neg x_{1,R} \vee \neg x_{3,R}) \\
\wedge & (\neg x_{2,V} \vee \neg x_{4,V}) \wedge (\neg x_{2,B} \vee \neg x_{4,B}) \wedge (\neg x_{2,R} \vee \neg x_{4,R}) \\
\wedge & (\neg x_{3,V} \vee \neg x_{4,V}) \wedge (\neg x_{3,B} \vee \neg x_{4,B}) \wedge (\neg x_{3,R} \vee \neg x_{4,R}) \\
\wedge & (\neg x_{4,V} \vee \neg x_{5,V}) \wedge (\neg x_{4,B} \vee \neg x_{5,B}) \wedge (\neg x_{4,R} \vee \neg x_{5,R}) \\
\wedge & (\neg x_{3,V} \vee \neg x_{5,V}) \wedge (\neg x_{3,B} \vee \neg x_{5,B}) \wedge (\neg x_{3,R} \vee \neg x_{5,R}) \\
\wedge & (\neg x_{5,V} \vee \neg x_{6,V}) \wedge (\neg x_{5,B} \vee \neg x_{6,B}) \wedge (\neg x_{5,R} \vee \neg x_{6,R}) \\
\wedge & (\neg x_{3,V} \vee \neg x_{6,V}) \wedge (\neg x_{3,B} \vee \neg x_{6,B}) \wedge (\neg x_{3,R} \vee \neg x_{6,R})
\end{array}
$$

Coloriage de Cartes - Modélisation



Expression des contraintes

- Il faut exprimer la contrainte pour toutes les paires voisines de $Voisines = \{(1,2),(1,3),(2,4),(3,4),(4,5),(3,5),(5,6),(3,6)\}$:

$$\phi = \bigwedge_{(i,j)\in Voisines} \bigwedge_{c\in\{R,V,B\}} \neg x_{i,c} \vee \neg x_{j,c}$$

"deux régions voisines n'ont jamais la même couleur"

CONTRAINTE SUPPLÉMENTAIRE.

TOUTE REGION EST COLORIÉE :

" $\boxed{\text{POUR TOUTE}}$ RÉGION, $\boxed{\text{IL EXISTE}}$ UNE COULEUR QUI LA COLORIE"

$$\bigwedge_{r \in \{1, \dots, 6\}} \left(\bigvee_{c \in \{R, V, B\}} X_{r,c} \right)$$

<u>REMARQUE</u> : SANS CETTE CONTRAINTE, NE RIEN COLORIER (METTRE TOUTES LES VARIABLES À FAUX), SATISFAIT LA FORMULE JUSQUE LÀ CRÉÉE

CONTRAINTE "AU PLUS UNE COULEUR PAR REGION"

" POUR TOUTE REGION r ET TOUTE COULEUR c, SI r EST COLORIÉE PAR c, ALORS POUR TOUTE COULEUR $c' \neq c$, r N'EST PAS COLORIÉE PAR c' "

$$\bigwedge_{r \in \{1,\dots,6\}} \bigwedge_{c \in \{R,V,B\}} \left(X_{r,c} \longrightarrow \bigwedge_{\substack{c' \in \{R,V,B\} \\ c \neq c'}} \neg X_{r,c'} \right)$$

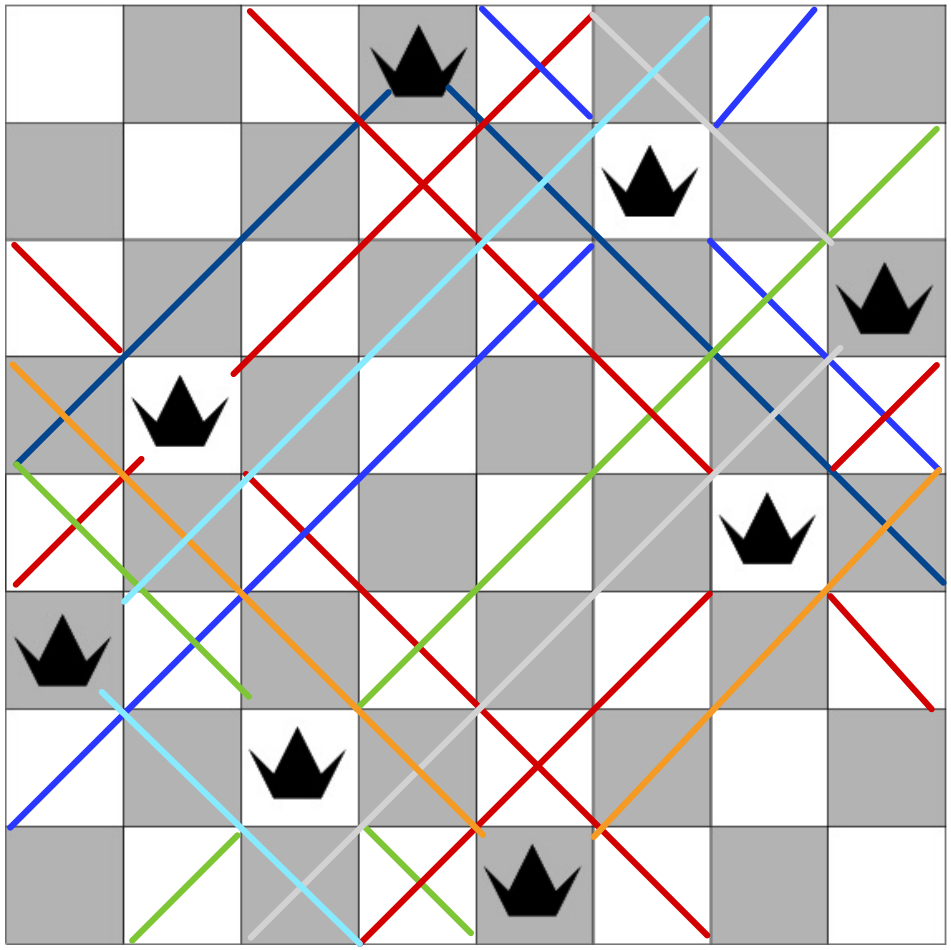
EN FNC :

$$\bigwedge_{\substack{r \in \{1,\dots,6\} \\ c,c' \in \{R,V,B\} \\ c \neq c'}} \left(\neg X_{r,c} \vee \neg X_{r,c'} \right)$$

REMARQUE : CETTE CONTRAINTE N'EST PAS NECESSAIRE. SI UNE REGION REÇOIT PLUS QU'UNE COULEUR, ALORS LES CONTRAINTES DE VOISINAGE SONT DE TOUTE MANIÈRE RESPECTÉES QUELQUES SOIENT LES COULEURS CHOISIES POUR CETTE RÉGION.

29 Problème des 8 reines (Au Tableau)

AUCUNE REINE NE DOIT EN ATTAQUER UNE AUTRE.



CHOIX DES VARIABLES

$$X = \{ x_{i,j} \mid i,j \in \{1, \dots, 8\} \}$$

avec la sémantique "$x_{i,j}$ est vraie <u>ssi</u> il y a une reine en case (i,j)".

EXPRESSION DES CONTRAINTES

1. Pas d'attaque horizontale
2. " " verticale
3. " " diagonale
4. Au moins une reine par ligne

① PAS D'ATTAQUE HORIZONTALE

" Pour toute ligne, il n'existe pas deux reines sur cette ligne "

$$\bigwedge_{i \in \{1,\dots,8\}} \neg \left(\bigvee_{\substack{j,j' \in \{1,\dots,8\} \\ j \neq j'}} X_{i,j} \wedge X_{i,j'} \right)$$

$$\equiv \bigwedge_{i,j,j' \in \{1,\dots,8\}} \left(\neg X_{i,j} \vee \neg X_{i,j'} \right)$$

$j \neq j'$ → POUR OPTIMISER, IL EST SUFFISANT DE PRENDRE $j' > j$. CAR LE CAS $j' < j$ A DÉJÀ ÉTÉ TRAITÉ

② PAS D'ATTAQUE VERTICALE

$$\bigwedge_{\substack{j,i,i' \in \{1,\dots,8\} \\ i \neq i'}} \left(\neg X_{i,j} \vee \neg X_{i',j} \right)$$

⟹ ON DIVISE LE NOMBRE DE CLAUSES PAR 2.

PAS D'ATTAQUE DIAGONALE

$$\left(\neg X_{i_1, j_1} \vee \neg X_{i_2, j_2} \right)$$

$i_1, j_1, i_2, j_2 \in \{1, \dots, 8\}$

(i_1, j_1) et (i_2, j_2) sont sur la même diagonale

$$|i_1 - i_2| = |j_1 - j_2|$$

AU	MOINS	UNE	REINE	PAR	LIGNE

$$\bigwedge_{i \in \{1,\dots,8\}} \left(\bigvee_{j \in \{1,\dots,8\}} x_{i,j} \right)$$

8 Reines - Codage en PySAT (1)

- codons le problème dans la librairie PySAT
- PySAT est une librairie Python qui fournit une interface permettant d'uiliser différents solveurs SAT.

8 Reines - Codage en PySAT (1)

```
from pysat.solvers import Minisat22
from pysat.solvers import Glucose4
from pysat.formula import CNF
from pysat.formula import IDPool


nb = 8 # nombre de reines

affichage_sol = True # affichage d'une solution

vpool = IDPool(start_from=1) # pour le stockage des identifiants entiers des couples (i,j)

cnf = CNF()  # construction d'un objet formule en forme normale conjonctive (Conjunctive Normal Form)
```

- Pour PySAT, une variable booléenne est un entier positif, par exemple 7, et sa négation un entier négatif, par exemple, -7
- La classe `IDPool` permet de créer automatiquement un codage des objets qu'on veut manipuler, par exemple des paires d'entiers, par des entiers positifs
- `cnf` est un objet qui représente une formule en FNC. Chaque clause est une liste de litéraux, et une formule est une liste de clauses.

8 Reines - Codage en PySAT (2)

Le code suivant ajoute un ensemble de clauses qui code la contrainte qu'il existe au moins une reine par ligne `i` :

```
print("Au moins un reine par ligne")

# au moins un reine par ligne
for i in range(nb):
    d = []
    for j in range(nb):
        d.append(vpool.id((i,j)))
    cnf.append(d)
```

- la variable `d` représente une disjonction de litéraux
- `vpool.id((i,j))` retourne un entier qui code la paire `(i,j)`, et l'ajoute à `vpool` si c'est la première fois que cette paire est rencontrée
- `cnf.append(d)` ajoute la clause `d` à l'ensemble des clauses

8 Reines - Codage en PySAT (3)

Le code suivant encode les contraintes qui expriment qu'il n'y a ni d'attaque en ligne ni d'attaque en colonne :

```
# pas d'attaque en ligne
print("Pas d'attaque en ligne")

for i in range(nb):
    for j in range(nb):
        for k in range(nb):
            if k > j:
                cnf.append([-vpool.id((i,j)),-vpool.id((i,k))])

# pas d'attaque en colonne
print("Pas d'attaque en colonne")

for i in range(nb):
    for j in range(nb):
        for k in range(nb):
            if k > j:
                cnf.append([-vpool.id((j,i)),-vpool.id((k,i))])
```

- `cnf.append([-vpool.id((i,j)),-vpool.id((i,k))])` ajoute une clause à deux éléments, en négation.

8 Reines - Codage en PySAT (4)

Ici on encode la contrainte "pas d'attaque diagonale" :

```
# pas d'attaque en diagonale

print("Pas d'attaque en diagnale")

# version lente
for i in range(nb):
    for j in range(nb):
        for k in range(nb):
            for l in range(nb):
                if abs(i-k)==abs(j-l) and i!=k:
                    cnf.append([-vpool.id((i,j)),-vpool.id((k,l))])
```

- Pour toute paire de cases `(i,j)` et `(k,l)`, le code `abs(i-k)==abs(j-l)` teste si elles sont sur la même diagonale
- ce code est lent quand `nb` est grand, on va voir une version optimisée.

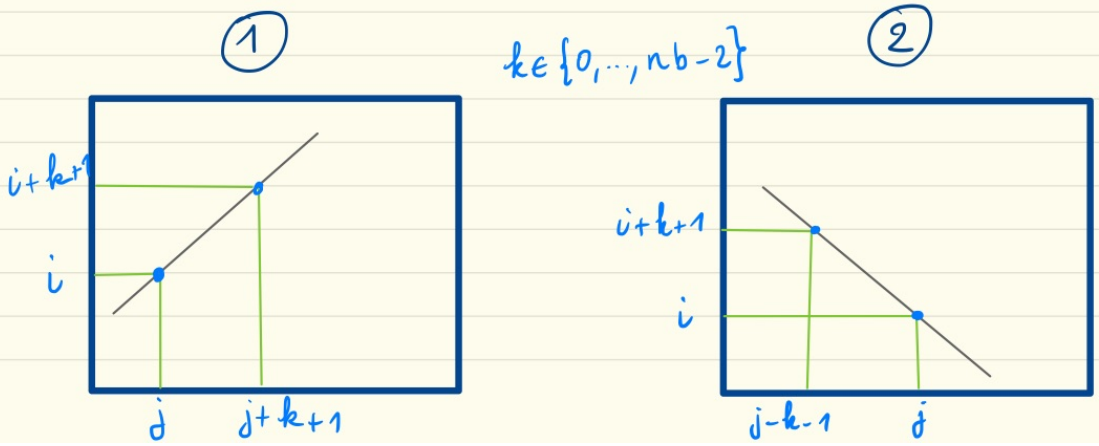
8 Reines - Codage en PySAT (5)

Version optimisée :

```
# version optimisee
for i in range(nb):
    for j in range(nb):
        for k in range(nb-1):
                if i+k+1<nb and j+k+1<nb:
                    cnf.append([-vpool.id((i,j)),-vpool.id((i+k+1,j+k+1))])


for i in range(nb):
    for j in range(nb):
        for k in range(nb-1):
                if i+k+1<nb and j-k-1>=0:
                    cnf.append([-vpool.id((i,j)),-vpool.id((i+k+1,j-k-1))])
```

Deux cas :



8 Reines - Codage en PySAT (6)

Lancement d'un solveur :

```
# phase de resolution

s = Minisat22(use_timer=True) # pour utiliser le solveur MiniSAT
# s = Glucose4(use_timer=True) # pour utiliser le solveur Glucose
s.append_formula(cnf.clauses, no_return=False)

print("Resolution...")
resultat = s.solve()
print("satisfaisable : " + str(resultat))

print("Temps de resolution : " + '{0:.2f}s'.format(s.time()))
```

- `s = Minisat22(use_timer=True)` crée un objet solveur MiniSAT (on pourrait utiliser un autre solveur)
- `use_timer=True` permet de compter le temps passé à la résolution
- `s.append_formula(cnf.clauses, no_return=False)` ajoute la formule au solveur
- `s.solve()` lance le solveur et retourne le résultat dans `resultat`
- `s.time()` retourne le temps passé à la résolution (depuis le dernier appel à la méthode `solve()`).

8 Reines - Codage en PySAT (7)

Remarque : La résolution est incrémentale : si on ajoute de nouvelles clauses à `s` et qu'on appelle une deuxième fois la méthode `solve()`, la résolution réutilise les calculs déjà faits lors du premier appel.

8 Reines - Codage en PySAT (8)

Affichage de la solution trouvée :

```
# affichage solution

if affichage_sol and resultat:

    print("\nVoici une solution: \n")

    for i in range(nb):
        for j in range(nb):
            if vpool.id((i,j)) in s.get_model():
                print("R ",end='')
            else:
                print("* ",end='')
        print("")
```

- `s.get_model()` retourne une interprétation qui satisfait la formule, sous forme de liste de litéraux.
- Par exemple, $V(1) = V(2) = 0$ et $V(3) = V(4) = V(5) = 1$ est représentée par la liste `[-1,-2,3,4,5]`.

OBTENIR UNE AUTRE SOLUTION (SI ELLE EXISTE)

Si la solution trouvée à la première étape est une valuation $V : X \to \{0,1\}$ de l'ensemble X des variables, alors on ajoute une clause qui demande à ce que la nouvelle solution V' satisfasse $V'(x) \neq V(x)$ pour au moins un $x \in X$:

$$\left(\bigvee_{\substack{x \in X \\ V(x)=1}} \neg x \right) \vee \left(\bigvee_{\substack{x \in X \\ V(x)=0}} x \right)$$

d contient représente cette clause

on ajoute d à l'ensemble existant

```python
print("calcul d'une autre solution si possible")
d = []
for v in s.get_model():
    d.append(-v)
cnf.append(d)
s.append_formula(cnf.clauses, no_return=False)

resultat = s.solve()
print("satisfaisable : " + str(resultat))
```

CODE PYTHON

on appelle le solveur de nouveau : il repart des calculs déjà effectués, on dit qu'il est <u>incrémental</u>

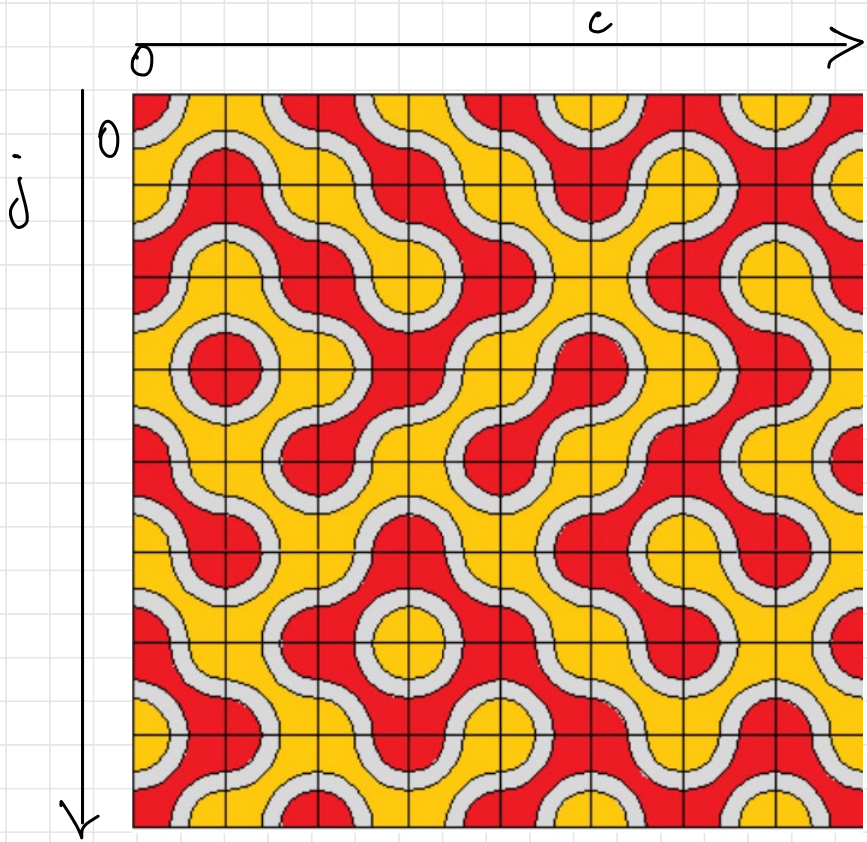
PROBLEME DE PAVAGE

EXEMPLE Etant donnés dim $\in \mathbb{N}$ et

Peut-on paver le carré de dimension dim x dim ?

PROBLEME DE PAVAGE

dim = 8



FORMALISATION DU PROBLÈME

ENTRÉE : Un ensemble fini T de "types" d'arêtes

Un ensemble fini $C \subseteq T^4$ de carreaux

Une dimension $dim \in \mathbb{N}$

Pour $c = (c_1, c_2, c_3, c_4)$ dans C, on pose
$c.nord = c_1$ $c.sud = c_2$
$c.ouest = c_3$ $c.est = c_4$

SORTIE : Une fonction totale $f : \{1, \dots, dim\}^2 \longrightarrow C$ telle que

$i \rightarrow$

$j \downarrow$

① pour tout $1 \leq i \leq dim$ et $1 < j \leq dim$,

$$f(i,j).nord = f(i, j-1).sud$$

② pour tout $1 \leq i < dim$ et $1 \leq j \leq dim$,

$$f(i,j).est = f(i+1, j).ouest$$

CHOIX DES VARIABLES

$X_{i,j,c}$ pour tout $i,j \in \{1,\dots,\dim\}^2$
pour tout c carreau de C

« $X_{i,j,c}$ vraie ssi (i,j) contient c »

CONTRAINTES

Pour tout $(i,j) \in \{1,\dots,\dim\}^2$ avec $j>1$, pour tout carreau c,
si (i,j) contient c alors il existe un carreau c'
tel que $(i,j-1)$ contient c' et $c.\text{nord} = c'.\text{sud}$.

$$\bigwedge_{\substack{(i,j)\in\{1,\dots,\dim\}^2 \\ j>1}} \bigwedge_{c\in C} \left(X_{i,j,c} \rightarrow \bigvee_{\substack{c'\in C \\ c.\text{nord}=c'.\text{sud}}} X_{i,j-1,c'} \right)$$

POUR LES CONTRAINTES EST / OUEST :

$$\bigwedge_{\substack{(i,j) \in \{1,\dots,dim\}^2 \\ i < dim}} \bigwedge_{c \in C} \left(X_{i,j,c} \rightarrow \bigvee_{\substack{c' \in C \\ c.est = c'.ouest}} X_{i+1,j,c'} \right)$$

IL FAUT AU MOINS UN CARREAU PAR CASE :

$$\bigwedge_{(i,j) \in \{1,\dots,dim\}^2} \left(\bigvee_{c \in C} X_{i,j,c} \right)$$

IL FAUT AU PLUS UN CARREAU PAR CASE

$$\bigwedge_{\substack{(i,j) \in \{1,\dots,dim\}^2 \\ c_1, c_2 \in C \\ c_1 \neq c_2}} \left(\neg X_{i,j,c_1} \vee \neg X_{i,j,c_2} \right)$$

AUTRE SOLUTION POSSIBLE POUR LES CONTRAINTES NORD/SUD

$$\bigwedge_{\substack{1 \leq i \leq dim \\ 1 < j \leq dim \\ c_1, c_2 \in C \\ c_1.nord \neq c_2.sud}} \left(\neg X_{i,j,c_1} \vee \neg X_{i,j-1,c_2} \right) \leftarrow$$

on ne peut pas à la fois avoir c_1 dans (i,j) et c_2 dans $(i,j-1)$

C'EST CETTE SOLUTION QUI EST IMPLÉMENTÉE EN PYTHON.

COMMENT TROUVER LES VARIABLES À UTILISER ?

SI LE PROBLEME CONSISTE À TROUVER UNE FONCTION

$$f : A \longrightarrow B$$

POUR DEUX ENSEMBLES QUELCONQUES FINIS A, B,

ON POSE UNE VARIABLE PAR PAIR $(a,b) \in A \times B$.

$$X = \{ x_{a,b} \mid a \in A, \ b \in B \}$$

AVEC L'INTERPRÉTATION " $x_{a,b}$ est vraie <u>ssi</u> a est associée à b par la fonction".

EXEMPLES :

coloriage : $A =$ Régions $B =$ couleurs

8-reines : $A = \{1, \dots, 8\}^2$ $B = \{0, 1\}$ (VOIR AUSSI SLIDES SUIVANT)

pavage : $A = \{1, \dots, \dim\}^2$ $B =$ ensemble des carreaux

Sudoku : $A = \{1, \dots, 9\}^2$ $B = \{1, \dots, 9\}$

IL SE POURRAIT QUE LE SOLVEUR TROUVE UNE SOLUTION AVEC $X_{a,b}$ vrai et $X_{a,b'}$ vrai pour $b \neq b'$

$\longrightarrow$ ON DOIT IMPOSER L'EXISTENCE D'UNE UNIQUE VALEUR !

AU MOINS UNE : $\bigwedge_{a \in A} \left(\bigvee_{b \in B} X_{a,b} \right)$

AU PLUS UNE : $\bigwedge_{a \in A} \bigwedge_{\substack{b_1, b_2 \in B \\ b_1 \neq b_2}} \left(\neg X_{a,b_1} \vee \neg X_{a,b_2} \right)$

Dans le cas* où $B = \{0, 1\}$, on peut être plus efficace (avoir moins de variables), car les variables ont une valeur vraie ou fausse, qu'on peut utiliser pour "coder" les valeurs de B. Par exemple, dans les 8 reines, il suffit d'avoir une variable par case : si elle est vraie, il y a une reine, sinon il n'y a pas de reine.

* Et de manière générale quand B a deux éléments uniquement

39 Problème du Sudoku (Au tableau)

i

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

j

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

CHOIX DES VARIABLES

On cherche une fonction $f : \underbrace{\{1,\dots,9\}}_{\text{coordonnées des cases}}{}^2 \longrightarrow \underbrace{\{1,\dots,9\}}_{\text{valeurs}}$

On peut voir cette fonction comme une fonction

$$g : \{1,\dots,9\}^3 \longrightarrow \{0,1\}$$

définie par $g(i,j,v) = 1$ <u>ssi</u> $f(i,j) = v$

Donc on peut prendre une variable par triplet $(i,j,v) \in \{1,\dots,9\}$

$$X = \{ x_{i,j,v} \mid i,j,v \in \{1,\dots,9\} \}$$

Jamais deux valeurs différentes dans une même case

- Si la case $(5,7)$ contient 3 alors elle ne contient pas 6

$$\neg\left(x_{5,7,3} \wedge x_{5,7,6}\right) \equiv \neg x_{5,7,3} \vee \neg x_{5,7,6}$$
$$\equiv x_{5,7,3} \rightarrow \neg x_{5,7,6}$$

- La case $(5,7)$ ne contient pas deux valeurs différentes

$$\bigwedge_{v=1}^{g} \bigwedge_{v'=v+1}^{g} \left(\neg x_{5,7,v} \vee \neg x_{5,7,v'}\right)$$

- Jamais deux $\neq$ dans une même case

$$\bigwedge_{(i,j)\in\{1,\dots,g\}^2} \bigwedge_{v=1}^{g} \bigwedge_{v'=v+1}^{g} \left(\neg x_{i,j,v} \vee \neg x_{i,j,v'}\right)$$

Au moins une valeur par case

Pour toute case (i,j), il existe une valeur v dans (i,j)

$$\bigwedge_{(i,j)\in\{1,\dots,9\}^2} \left(\bigvee_{v\in\{1,\dots,9\}} x_{i,j,v} \right)$$

Jamais deux fois la même valeur sur une même ligne

Pour toute ligne i, pour toute colonne j, pour toute valeur v, si (i,j) contient v, alors pour toute colonne $k \neq j$, (i,k) <u>ne contient pas</u> v

$X_{i,j,v}$ $\longrightarrow$

$\neg X_{i,k,v}$

$v \in \{1,\dots,9\}$
$i \in \{1,\dots,9\}$
$j \in \{1,\dots,9\}$

$k \in \{1,\dots,9\}$
$k \neq j$

$$\equiv \bigwedge_{\substack{v \in \{1,\dots,9\} \\ i \in \{1,\dots,9\} \\ j \in \{1,\dots,9\}}} \left(\neg X_{i,j,v} \vee \left(\bigwedge_{\substack{k \in \{1,\dots,9\} \\ k \neq j}} \neg X_{i,k,v} \right) \right)$$

$$\equiv \bigwedge_{\substack{v \in \{1,\dots,9\} \\ i \in \{1,\dots,9\} \\ j \in \{1,\dots,9\}}} \bigwedge_{\substack{k \in \{1,\dots,9\} \\ k \neq j}} \left(\neg X_{i,j,v} \vee \neg X_{i,k,v} \right)$$

$$\equiv \bigwedge_{\substack{v \in \{1,\dots,9\} \\ i \in \{1,\dots,9\} \\ j \in \{1,\dots,9\}}} \left(\neg X_{i,j,v} \vee \neg X_{i,k,v} \right) \quad k \in \{j+1,\dots,9\}$$

<u>Remarque</u> : contrainte générale pour un sous-ensemble de cases.

Soit $C \subseteq \{1, .., 9\}^2$

Dans C, il n'existe pas deux fois la même valeur

$$\bigwedge_{\substack{(i,j) \in C \\ (i',j') \in C \\ (i,j) \neq (i',j') \\ v \in \{1,..,9\}}} \left(\neg X_{i,j,v} \vee \neg X_{i',j',v} \right)$$

Jamais deux fois la même valeur sur une même colonne

$$\bigwedge_{\substack{v \in \{1,\dots,9\} \\ j \in \{1,\dots,9\} \\ i \in \{1,\dots,9\} \\ i' \in \{i+1,\dots,9\}}} \left(\neg X_{i,j,v} \vee \neg X_{i',j,v} \right)$$

Jamais deux fois la même valeur dans un même sous-carré 3x3

$\bigwedge_{C \text{ sous-carré de la grille}} \bigwedge_{\substack{(i,j) \in C \\ (i',j') \in C \\ (i,j) \neq (i',j') \\ v \in \{1,\dots,9\}}} \left(\neg X_{i,j,v} \vee \neg X_{i',j',v} \right)$

La grille est pré-remplie

En entrée on se donne une fonction $G : \{1,\dots,9\}^2 \to \{1,\dots,9\} \cup \{NULL\}$

$\to$ on doit exprimer l'existence des valeurs prédéfinies

$\wedge$

$x_{i,j,G(i,j)}$

$(i,j) \in \{1,\dots,9\}^2$

$G(i,j) \neq NULL$

Remarque : La contrainte du début "jamais deux valeurs différentes sur une même case" est inutile car impliquée par toutes les autres.

Néanmoins, lors des tests, il apparaît qu'elle permet de mieux guider le solveur et donc d'être plus efficace.

40 Algorithme DPLL

(NON EVALUÉE À L'EXAMEN)
↳ JUSQUE SLIDE 50

- proposé par M. Davis, H. Putman, G. Logemann et D. Loveland en 1962.
- *principe de base* : engendrer et tester des solutions partielles
- Par exemple, pour tester la satisfiabilité de $(x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z)$, au lieu d'essayer toutes les valuations possibles de x, y et z, on peut d'abord essayer les interprétations possibles de x.
- Ici, on se rend compte tout de suite que la valuation (partielle) $V(x) \mapsto 1$ satisfait la formule, peu importe l'interprétation des autres propositions.
- l'algorithme DPLL propose des critères pour choisir quelles variables tester en premier.

41 Interprétation Partielle

Définition

Une *interprétation partielle* est un assignement noté $x/1$ ou $x/0$, qui signifie que assigne la valeur 1 à x ou la valeur 0.

41 Interprétation Partielle

Définition

Une *interprétation partielle* est un assignement noté $x/1$ ou $x/0$, qui signifie que assigne la valeur 1 à x ou la valeur 0.

Lorsqu'on a choisi la valeur d'une proposition, on peut simplifier la formule. Par exemple, la formule $(x \vee y) \wedge (\neg x \vee z \vee \neg y)$ se simplifie en $z \vee \neg y$ sous l'interprétation partielle $x/1$.
Définissons de manière formelle cette notion.

42 Simplification sous une interprétation partielle

Etant donnée une clause C et une interprétation partielle x/b où $b \in \{0, 1\}$, la formule $C[x/b]$ est obtenue selon la règle suivante :

- Si C ne contient pas x ou $\neg x$, alors $C[x/b] = C$
- Sinon si C ne contient que x ou que $\neg x$, alors on considère les cas suivants :
  1. $C = x$ et $b = 1$, alors $C[x/b] = \top$
  2. $C = x$ et $b = 0$, alors $C[x/b] = \bot$
  3. $C = \neg x$ et $b = 0$, alors $C[x/b] = \top$
  4. $C = \neg x$ et $b = 1$, alors $C[x/b] = \bot$
- sinon si C contient x et $b = 1$ ou C contient $\neg x$ et $b = 0$, alors $C[x/b] = \top$
- sinon on retire de C les occurences de x ou de $\neg x$

Si $\phi = \bigwedge_{i=1}^{n} C_i$ est une conjonction de clauses, alors $\phi[x/b] = \bot$ si il existe une clause C_i telle que $C_i[x/b] = \bot$, sinon $\phi[x/b] = \bigwedge_{i:C_i[x/b] \neq \top} C_i[x/b]$ (on simplifie toutes les clauses et on retire celle qui deviennent égales à $\top$). Si toutes les clauses se simplifient en $\top$, alors $\phi[x/b] = \top$.

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] =$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$,

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] =$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$,

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] =$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] =$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] = x \vee \neg y$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] = x \vee \neg y$
- $a[x/0] =$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] = x \vee \neg y$
- $a[x/0] = y \vee z$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] = x \vee \neg y$
- $a[x/0] = y \vee z$
- $b[x/0] =$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] = x \vee \neg y$
- $a[x/0] = y \vee z$
- $b[x/0] = \neg y \vee \neg z$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] = x \vee \neg y$
- $a[x/0] = y \vee z$
- $b[x/0] = \neg y \vee \neg z$
- $(a \wedge b)[x/0] =$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] = x \vee \neg y$
- $a[x/0] = y \vee z$
- $b[x/0] = \neg y \vee \neg z$
- $(a \wedge b)[x/0] = (y \vee z) \wedge (\neg y \vee \neg z)$
- $(a \wedge b)[x/0][y/0] =$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] = x \vee \neg y$
- $a[x/0] = y \vee z$
- $b[x/0] = \neg y \vee \neg z$
- $(a \wedge b)[x/0] = (y \vee z) \wedge (\neg y \vee \neg z)$
- $(a \wedge b)[x/0][y/0] = z$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] = x \vee \neg y$
- $a[x/0] = y \vee z$
- $b[x/0] = \neg y \vee \neg z$
- $(a \wedge b)[x/0] = (y \vee z) \wedge (\neg y \vee \neg z)$
- $(a \wedge b)[x/0][y/0] = z$
- $(a \wedge b)[x/0][y/0][z/1] =$

43 Remarque et exemples

Remarque

$\phi[x/b]$ est une formule, on peut donc lui réappliquer une autre interprétation partielle $[y/b']$ et obtenir une nouvelle formule $(\phi[x/b])[y/b']$ qu'on notera simplement $\phi[x/b][y/b']$.

Exemples

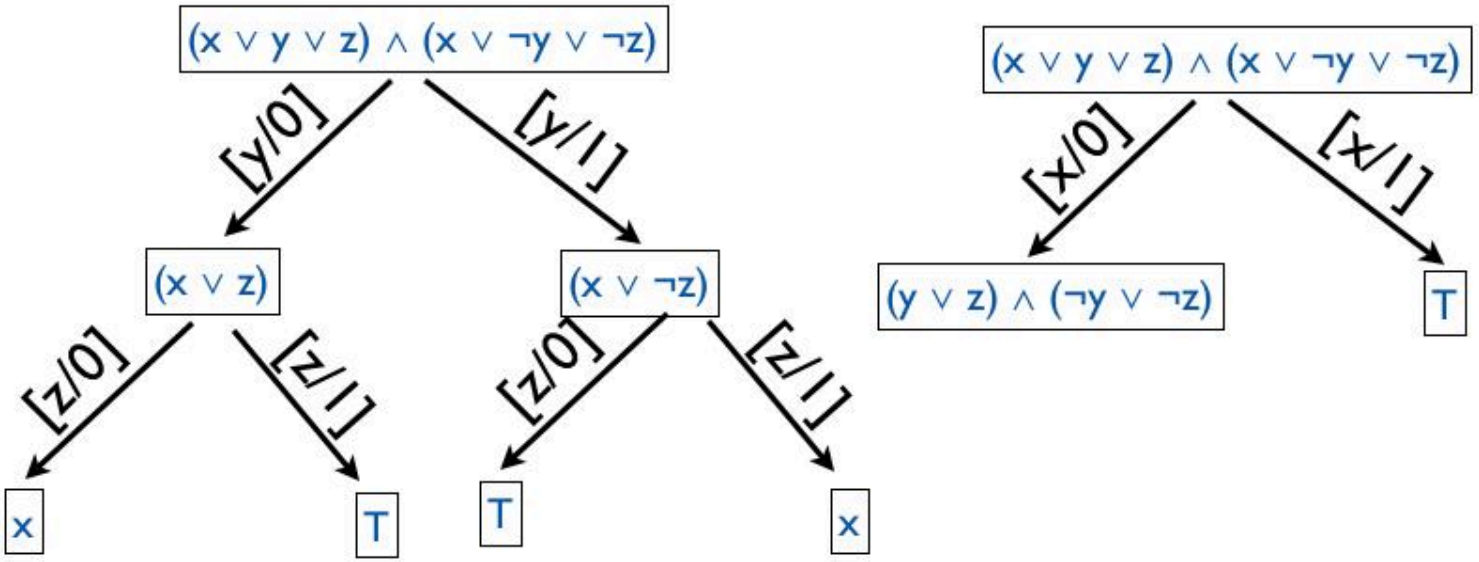
Soit $a = x \vee y \vee z$ et $b = x \vee \neg y \vee \neg z$. Alors :

- $a[x/1] = \top$, $b[x/1] = \top$, $(a \wedge b)[x/1] = \top$
- $b[y/1] = x \vee \neg z$
- $b[z/1] = x \vee \neg y$
- $a[x/0] = y \vee z$
- $b[x/0] = \neg y \vee \neg z$
- $(a \wedge b)[x/0] = (y \vee z) \wedge (\neg y \vee \neg z)$
- $(a \wedge b)[x/0][y/0] = z$
- $(a \wedge b)[x/0][y/0][z/1] = \top$

44 Proposition Pivot

- l'algorithme DPLL va essayer des interprétations partielles, la proposition choisie est la *proposition pivot*. Il va successivement essayer de mettre la proposition à vrai ou à faux, et tester récursivement la satisfaisabilité de formules simplifiées obtenues.
- le choix de la proposition choisie peut fortement influencer le résultat comme on peut le voir sur cet exemple :

$$\phi = (x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z)$$



45 Premier critère de choix : clauses unitaires

- Une clause unitaire est une clause qui ne contient qu'un seul littéral x ou $\neg x$
- une clause unitaire force donc le choix de la valeur de la proposition. Par exemple dans $x \wedge (y \vee \neg z)$, il y a une clause unitaire x et on sait que pour satisfaire cette formule, il faut nécessairement interpréter x par 1
- l'algorithme DPLL choisi en priorité la proposition d'une clause unitaire comme proposition pivot.
- la formule simplifiée obtenue peut elle même contenir des clauses unitaires : *propogation de clauses unitaires*
- c'est une optimisation essentielle des solveurs SAT, qui passent l'essentiel de leur temps à faire de la propogation de clauses unitaires.

46 Propagation de clauses unitaires : exemple

Prenons

$$\phi = (x \vee y) \wedge \neg y \wedge (\neg x \vee y \vee \neg z)$$

Alors

$$\phi[y/0] = x \wedge (\neg x \vee \neg z)$$

ce qui donne la nouvelle clause unitaire x qui impose donc d'assigner la valeur 1 à x :

$$\phi[y/0][x/1] = \neg z$$

qui ne contient qu'une seule clause, unitaire, et on obtient finalement :

$$\phi[y/0][x/1][z/0] = \top$$

Donc ϕ est satisfaisable avec l'interprétation $V(x) = 1$ et $V(y) = V(z) = 0$.

47 Deuxième critère de choix : proposition à polarité unique

Prenons la formule

$$\phi = (x \vee \neg y \vee z) \wedge (x \vee \neg z) \wedge (y \vee z) \wedge (x \vee \neg y)$$

- dans cette formule, x apparaît toujours positivement, on peut donc directement lui assigner la valeur 1 sans être obligé de tester la valeur 0.
- on peut de même avoir un effet cascade : par exemple avec

$$(x \vee \neg y) \wedge (y \vee \neg z_1) \wedge (\neg z_1 \vee \neg z_2)$$

(Exercice)

48 Algorithme DPLL

Principe

- génération d'interprétation partielle
- la proposition pivot est choisie avec le critère de clause unitaire d'abord, et de polarité ensuite
- sinon, lorsque les deux critères ne s'appliquent pas, une proposition est choisie au hasard

49 Algorithme DPLL

Fonction DPLL(ϕ) = `retourne VRAI ssi` ϕ `est satisfaisable`

1. si $\phi = \top$ alors retourner VRAI (la formule est satisfaisable)
2. sinon si $\phi = \bot$ alors retourner FAUX (la formule n'est pas satisfaisable)
3. sinon si ϕ contient une clause unitaire x, alors retourner DPLL($\phi[x/1]$)
4. sinon si ϕ contient une clause unitaire $\neg x$, alors retourner DPLL($\phi[x/0]$)
5. sinon si ϕ contient une proposition x de polarité toujours positive, alors retourner DPLL($\phi[x/1]$)
6. sinon si ϕ contient une proposition x de polarité toujours négative, alors retourner DPLL($\phi[x/0]$)
7. sinon dans tous les autres cas, choisir une proposition x au hasard et retourner (DPLL($\phi[x/0]$) `ou` DPLL($\phi[x/1]$))

51 Transformation de Tseitin

- parfois, le problème qu'on cherche à résoudre ne s'exprime pas facilement par une formule en FNC
- Tranformation de Tseitin : ajouter des nouvelles variables et des équivalences

Exemple

Considérons $\phi = (x \wedge q) \vee \neg(y \vee r)$. Dans la transformation de Tseitin, on va remplacer $x \wedge q$ par une nouvelle variable x_1, et $\neg(y \vee r)$ par une nouvelle variable x_2. Nous considérons alors la formule

$$(x_1 \vee x_2) \wedge (x_1 \leftrightarrow x \wedge q) \wedge (x_2 \leftrightarrow \neg(y \vee r))$$

Il reste encore à mettre les deux formules $(x_1 \leftrightarrow x \wedge q)$ et $(x_2 \leftrightarrow \neg(y \vee r))$ sous FNC.

52 Transformation de Tseitin

Exemple (Suite)

On a la série d'équivalences suivantes :

$$\begin{aligned} & x_1 \leftrightarrow x \wedge q \\ \equiv & \end{aligned}$$

52 Transformation de Tseitin

Exemple (Suite)

On a la série d'équivalences suivantes :

$$\begin{aligned} & x_1 \leftrightarrow x \wedge q \\ \equiv \quad & (x_1 \rightarrow (x \wedge q)) \wedge ((x \wedge q) \rightarrow x_1) \\ \equiv \quad & \end{aligned}$$

52 Transformation de Tseitin

Exemple (Suite)

On a la série d'équivalences suivantes :

$$
\begin{aligned}
& x_1 \leftrightarrow x \wedge q \\
\equiv\ & (x_1 \rightarrow (x \wedge q)) \wedge ((x \wedge q) \rightarrow x_1) \\
\equiv\ & (\neg x_1 \vee (x \wedge q)) \wedge (\neg(x \wedge q) \vee x_1) \\
\equiv\ &
\end{aligned}
$$

52 Transformation de Tseitin

Exemple (Suite)

On a la série d'équivalences suivantes :

$$\begin{array}{rl} & x_1 \leftrightarrow x \wedge q \\ \equiv & (x_1 \rightarrow (x \wedge q)) \wedge ((x \wedge q) \rightarrow x_1) \\ \equiv & (\neg x_1 \vee (x \wedge q)) \wedge (\neg(x \wedge q) \vee x_1) \\ \equiv & (\neg x_1 \vee x) \wedge (\neg x_1 \vee q) \wedge (\neg x \vee \neg q \vee x_1) \end{array}$$

et la série suivante :

$$\begin{array}{rl} & x_2 \leftrightarrow \neg(y \vee r) \\ \equiv & \end{array}$$

52 Transformation de Tseitin

Exemple (Suite)

On a la série d'équivalences suivantes :

$$\begin{array}{rl} & x_1 \leftrightarrow x \wedge q \\ \equiv & (x_1 \rightarrow (x \wedge q)) \wedge ((x \wedge q) \rightarrow x_1) \\ \equiv & (\neg x_1 \vee (x \wedge q)) \wedge (\neg(x \wedge q) \vee x_1) \\ \equiv & (\neg x_1 \vee x) \wedge (\neg x_1 \vee q) \wedge (\neg x \vee \neg q \vee x_1) \end{array}$$

et la série suivante :

$$\begin{array}{rl} & x_2 \leftrightarrow \neg(y \vee r) \\ \equiv & (x_2 \rightarrow \neg(y \vee r)) \wedge (\neg(y \vee r) \rightarrow x_2) \\ \equiv & \end{array}$$

52 Transformation de Tseitin

Exemple (Suite)

On a la série d'équivalences suivantes :

$$\begin{array}{rl} & x_1 \leftrightarrow x \wedge q \\ \equiv & (x_1 \rightarrow (x \wedge q)) \wedge ((x \wedge q) \rightarrow x_1) \\ \equiv & (\neg x_1 \vee (x \wedge q)) \wedge (\neg(x \wedge q) \vee x_1) \\ \equiv & (\neg x_1 \vee x) \wedge (\neg x_1 \vee q) \wedge (\neg x \vee \neg q \vee x_1) \end{array}$$

et la série suivante :

$$\begin{array}{rl} & x_2 \leftrightarrow \neg(y \vee r) \\ \equiv & (x_2 \rightarrow \neg(y \vee r)) \wedge (\neg(y \vee r) \rightarrow x_2) \\ \equiv & (\neg x_2 \vee \neg y \wedge \neg r) \wedge (y \vee r \vee x_2) \\ \equiv & \end{array}$$

52 Transformation de Tseitin

Exemple (Suite)

On a la série d'équivalences suivantes :

$$\begin{array}{rl} & x_1 \leftrightarrow x \wedge q \\ \equiv & (x_1 \rightarrow (x \wedge q)) \wedge ((x \wedge q) \rightarrow x_1) \\ \equiv & (\neg x_1 \vee (x \wedge q)) \wedge (\neg(x \wedge q) \vee x_1) \\ \equiv & (\neg x_1 \vee x) \wedge (\neg x_1 \vee q) \wedge (\neg x \vee \neg q \vee x_1) \end{array}$$

et la série suivante :

$$\begin{array}{rl} & x_2 \leftrightarrow \neg(y \vee r) \\ \equiv & (x_2 \rightarrow \neg(y \vee r)) \wedge (\neg(y \vee r) \rightarrow x_2) \\ \equiv & (\neg x_2 \vee \neg y \wedge \neg r) \wedge (y \vee r \vee x_2) \\ \equiv & (\neg x_2 \vee \neg y) \wedge (\neg x_2 \vee \neg r) \wedge (y \vee r \vee x_2) \end{array}$$

53 Transformation de Tseitin

Exemple (Suite)

La formule ϕ de départ était $(x \wedge q) \vee \neg(y \vee r)$ qu'on a transformé en :

$$(x_1 \vee x_2) \wedge (x_1 \leftrightarrow x \wedge q) \wedge (x_2 \leftrightarrow (y \vee r))$$

On a montré les équivalences suivantes :

$$x_1 \leftrightarrow x \wedge y \equiv (\neg x_1 \vee p) \wedge (\neg x_1 \vee q) \wedge (\neg x \vee \neg y \vee x_1)$$
$$x_2 \leftrightarrow (\neg x_2 \vee \neg q) \wedge (\neg x_2 \vee \neg r) \wedge (y \vee r \vee x_1)$$

Donc finalement on obtient la formule sous FNC suivante $\psi =$

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee p) \wedge (\neg x_1 \vee q) \wedge (\neg x \vee \neg y \vee x_1) \wedge (\neg x_2 \vee \neg q) \wedge (\neg x_2 \vee \neg r) \wedge (y \vee r \vee x_1)$$

Note : la satisfaisabilité est préservée, i.e. ϕ est satisfaisable ssi ψ est satisfaisable.

54 Transformation de Tseitin

- la technique de Tseitin sera particulièrement intéressante lorsqu'on devra mettre sous FNC des formules qui sont sous forme normale *disjonctive*

$$C_1 \vee C_2 \vee \cdots \vee C_n$$

  où les C_i sont des conjonctions de littéraux.
- on va introduire une variable x_i pour chaque C_i, et on obtient la formule

$$(x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (x_1 \leftrightarrow C_1) \wedge \cdots \wedge (x_n \leftrightarrow C_n)$$

- supposons que $C_i = l_1 \wedge \ldots l_k$ où les l_i sont des littéraux, alors mettre $x_i \leftrightarrow C_i$ sous FNC est assez simple :

$$x_i \leftrightarrow C_i \equiv (\bigwedge_{j=1}^{k} (\neg x_i \vee l_j)) \wedge (x_i \vee \bigvee_{j=1}^{k} \neg l_j)$$

  qui est bien sous FNC.

MISE EN FNC de $x \leftrightarrow (l_1 \wedge l_2 \wedge \dots \wedge l_k)$

$$\left[x \longrightarrow (l_1 \wedge \dots \wedge l_k) \right] \wedge \left[\underbrace{(l_1 \wedge \dots \wedge l_k) \rightarrow x} \right]$$

$$\left[\neg x \vee (l_1 \wedge \dots \wedge l_k) \right] \wedge \left[\neg l_1 \vee \dots \vee \neg l_k \vee x \right]$$

$$(\neg x \vee l_1) \wedge (\neg x \vee l_2) \wedge \dots \wedge (\neg x \vee l_k) \wedge (\neg l_1 \vee \dots \vee \neg l_k \vee x)$$

$$\left(\bigwedge_{i=1}^{k} (\neg x \vee l_i) \right) \wedge \left(x \vee \bigvee_{i=1}^{k} \neg l_i \right)$$

55 A retenir

Principe (souvent) valable pour les SAT solveurs

Ne pas hésiter à introduire des nouvelles variables pour minimiser le nombre de clauses !

56 Variantes de SAT

- 2-SAT : les clauses ne contiennent qu'au plus deux littéraux.
  - Par exemple $(x_1 \vee \neg x_2) \wedge x_3 \wedge (x_4 \vee \neg x_1) \wedge (\neg x_3 \vee \neg x_2)$
  - 2-SAT est solvable en temps polynomial (alors que 3-SAT est NP-complet).

56 Variantes de SAT

- 2-SAT : les clauses ne contiennent qu'au plus deux littéraux.
  - Par exemple $(x_1 \vee \neg x_2) \wedge x_3 \wedge (x_4 \vee \neg x_1) \wedge (\neg x_3 \vee \neg x_2)$
  - 2-SAT est solvable en temps polynomial (alors que 3-SAT est NP-complet).
- QSAT : décider la satisfaisabilité de formules de la forme

$$\forall x_1 \exists q_1 \forall x_2 \exists q_2 \ldots \forall q_n \exists q_n . \phi$$

où ϕ est une formule en FNC construite sur les propositions $x_1, q_1, \ldots, x_n, q_n$.

56 Variantes de SAT

- 2-SAT : les clauses ne contiennent qu'au plus deux littéraux.
  - Par exemple $(x_1 \vee \neg x_2) \wedge x_3 \wedge (x_4 \vee \neg x_1) \wedge (\neg x_3 \vee \neg x_2)$
  - 2-SAT est solvable en temps polynomial (alors que 3-SAT est NP-complet).
- QSAT : décider la satisfaisabilité de formules de la forme

$$\forall x_1 \exists q_1 \forall x_2 \exists q_2 \ldots \forall q_n \exists q_n . \phi$$

où ϕ est une formule en FNC construite sur les propositions $x_1, q_1, \ldots, x_n, q_n$.

  - par exemple : $\forall x \exists q . (x \vee q) \wedge (\neg x \vee \neg q)$.

CAS $x = 1$ $\longrightarrow$ on doit prendre $q = 0$

CAS $x = 0$ $\longrightarrow$ on doit prendre $q = 1$

56 Variantes de SAT

- 2-SAT : les clauses ne contiennent qu'au plus deux littéraux.
  - Par exemple $(x_1 \vee \neg x_2) \wedge x_3 \wedge (x_4 \vee \neg x_1) \wedge (\neg x_3 \vee \neg x_2)$
  - 2-SAT est solvable en temps polynomial (alors que 3-SAT est NP-complet).
- QSAT : décider la satisfaisabilité de formules de la forme

$$\forall x_1 \exists q_1 \forall x_2 \exists q_2 \ldots \forall q_n \exists q_n . \phi$$

  où ϕ est une formule en FNC construite sur les propositions $x_1, q_1, \ldots, x_n, q_n$.
  - par exemple : $\forall x \exists q . (x \vee q) \wedge (\neg x \vee \neg q)$.
  - Si p est vraie, alors q doit être fausse, et si p est fausse, alors q doit être vraie.

56 Variantes de SAT

- 2-SAT : les clauses ne contiennent qu'au plus deux littéraux.
  - Par exemple $(x_1 \vee \neg x_2) \wedge x_3 \wedge (x_4 \vee \neg x_1) \wedge (\neg x_3 \vee \neg x_2)$
  - 2-SAT est solvable en temps polynomial (alors que 3-SAT est NP-complet).
- QSAT : décider la satisfaisabilité de formules de la forme

$$\forall x_1 \exists q_1 \forall x_2 \exists q_2 \ldots \forall q_n \exists q_n . \phi$$

  où ϕ est une formule en FNC construite sur les propositions $x_1, q_1, \ldots, x_n, q_n$.
  - par exemple : $\forall x \exists q . (x \vee q) \wedge (\neg x \vee \neg q)$.
  - Si p est vraie, alors q doit être fausse, et si p est fausse, alors q doit être vraie.
  - QSAT est PSPACE-complet (QSAT peut être résolu en espace polynomial et tous les problèmes pouvant être résolus en espace polynomial peuvent se réduire au problème QSAT en temps polynomial, autrement dit, QSAT est aussi difficile que tous les problèmes pouvant être résolu en espace polynomial)

57 Variantes de SAT

- MAX-SAT : étant donnée une formule ϕ en FNC et un entier $k \in \mathbb{N}$, peut-on satisfaire au moins k clauses ?
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$

57 Variantes de SAT

- MAX-SAT : étant donnée une formule ϕ en FNC et un entier $k \in \mathbb{N}$, peut-on satisfaire au moins k clauses ?
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$on peut satisfaire au plus deux clauses.

57 Variantes de SAT

- MAX-SAT : étant donnée une formule ϕ en FNC et un entier $k \in \mathbb{N}$, peut-on satisfaire au moins k clauses ?
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$on peut satisfaire au plus deux clauses.
  - Pourquoi MAX-SAT est aussi difficile que SAT ?

57 Variantes de SAT

- MAX-SAT : étant donnée une formule ϕ en FNC et un entier $k \in \mathbb{N}$, peut-on satisfaire au moins k clauses ?
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$on peut satisfaire au plus deux clauses.
  - Pourquoi MAX-SAT est aussi difficile que SAT ?
- WEIGHTED-MAX-SAT : on attribue des poids à chaque clause, on se donne un entier r, et on veut savoir s'il on peut satisfaire un ensemble de clauses dont la somme des poids est au moins r

57 Variantes de SAT

- MAX-SAT : étant donnée une formule ϕ en FNC et un entier $k \in \mathbb{N}$, peut-on satisfaire au moins k clauses ?
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$on peut satisfaire au plus deux clauses.
  - Pourquoi MAX-SAT est aussi difficile que SAT ?
- WEIGHTED-MAX-SAT : on attribue des poids à chaque clause, on se donne un entier r, et on veut savoir s'il on peut satisfaire un ensemble de clauses dont la somme des poids est au moins r
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$ avec $poids(x) = 1$, $poids(y) = 1$ et $poids(\neg x \vee \neg y) = 2$ et $r = 3$

57 Variantes de SAT

- MAX-SAT : étant donnée une formule ϕ en FNC et un entier $k \in \mathbb{N}$, peut-on satisfaire au moins k clauses ?
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$on peut satisfaire au plus deux clauses.
  - Pourquoi MAX-SAT est aussi difficile que SAT ?
- WEIGHTED-MAX-SAT : on attribue des poids à chaque clause, on se donne un entier r, et on veut savoir s'il on peut satisfaire un ensemble de clauses dont la somme des poids est au moins r
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$ avec $poids(x) = 1$, $poids(y) = 1$ et $poids(\neg x \vee \neg y) = 2$ et $r = 3$ En mettant x et y à vrai, on ne satisfait pas la dernière clause, et on obtient le poids total $1 + 1 = 2$. En mettant x à vrai et y à faux, on satisfait la première et la dernière clause, et on obtient le poids total $1 + 2 = 3$.

57 Variantes de SAT

- MAX-SAT : étant donnée une formule ϕ en FNC et un entier $k \in \mathbb{N}$, peut-on satisfaire au moins k clauses ?
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$on peut satisfaire au plus deux clauses.
  - Pourquoi MAX-SAT est aussi difficile que SAT ?
- WEIGHTED-MAX-SAT : on attribue des poids à chaque clause, on se donne un entier r, et on veut savoir s'il on peut satisfaire un ensemble de clauses dont la somme des poids est au moins r
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$ avec $poids(x) = 1$, $poids(y) = 1$ et $poids(\neg x \vee \neg y) = 2$ et $r = 3$ En mettant x et y à vrai, on ne satisfait pas la dernière clause, et on obtient le poids total $1 + 1 = 2$. En mettant x à vrai et y à faux, on satisfait la première et la dernière clause, et on obtient le poids total $1 + 2 = 3$.
  - Pourquoi WEIGHTED-MAX-SAT est aussi difficile que MAX-SAT ?

57 Variantes de SAT

- MAX-SAT : étant donnée une formule ϕ en FNC et un entier $k \in \mathbb{N}$, peut-on satisfaire au moins k clauses ?
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$on peut satisfaire au plus deux clauses.
  - Pourquoi MAX-SAT est aussi difficile que SAT ?
- WEIGHTED-MAX-SAT : on attribue des poids à chaque clause, on se donne un entier r, et on veut savoir s'il on peut satisfaire un ensemble de clauses dont la somme des poids est au moins r
  - Par exemple $x \wedge y \wedge (\neg x \vee \neg y)$ avec $poids(x) = 1$, $poids(y) = 1$ et $poids(\neg x \vee \neg y) = 2$ et $r = 3$ En mettant x et y à vrai, on ne satisfait pas la dernière clause, et on obtient le poids total $1 + 1 = 2$. En mettant x à vrai et y à faux, on satisfait la première et la dernière clause, et on obtient le poids total $1 + 2 = 3$.
  - Pourquoi WEIGHTED-MAX-SAT est aussi difficile que MAX-SAT ? MAX-SAT est le cas particulier de WEIGHTED-MAX-SAT où le poids de chaque clause vaut 1.

58 Références

- http://en.wikipedia.org/wiki/Boolean_satisfiability_problem
- cours/tutoriaux : http://www.cse.unsw.edu.au/~tw/sat/
- http://www.pps.jussieu.fr/~treinen/teaching/ (cours de Ralf Treinen)
- http://www.cs.wm.edu/~idillig/cs780-02/lecture3.pdf
- Handbook of satisfiability. A. Biere, M. Heule, H. van Maaren, T. Walsh. IOS Press. 2008. 981 pages.