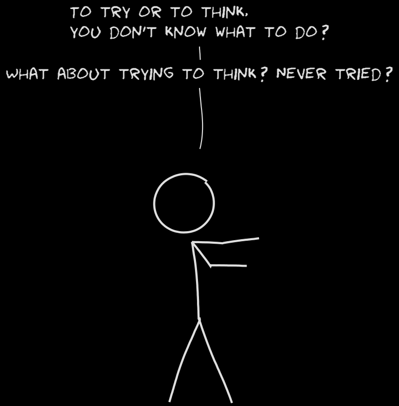
Intelligence Artificielle
Logique et SAT



LAURENT SIMON
BORDEAUX-INP / LABRI
@lorensipro
lsimon@labri.fr

L'I.A. c'est raisonner « logiquement »

L'intelligence Artificielle c'est savoir raisonner

Le raisonnement logique le plus simple : la logique propositionnelle

Raisonner c'est chercher la solution à un problème difficile

Matière par définition surhumaine : l'ordinateur va résoudre des problèmes hors d'atteinte des humains seuls

Dans ce cours

On va se focaliser sur une logique très simple mais :

- **SAT est la base de la programmation par contraintes**
- **On décrit le problème, l'ordinateur le résoud**
- **La *programmation par contraintes* est plus expressive**
- **L'idée est de comprendre l'aspect déclaratif de la méthode**

Laisser un algorithme spécialisé le résoudre

Toute l'intelligence est dans le choix des contraintes utilisées dans le problème

Why studying SAT ?

SAT, the canonical NP-Complete problem.

Related to a one-million dollar question (is NP=P ?)

- The **main open problem** of Theoretical Computer Science
- **Must be faced** in most real-world problems
- But, the *easiest of the hard* problems ⇐ **Hope here !**

S. Aaronson, MIT : « **If P = NP**, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in 'creative leaps,' no fundamental gap between solving a problem and recognizing the solution once it's found. **Everyone who could appreciate a symphony would be Mozart ; everyone who could follow a step-by-step argument would be Gauss.** » [1]

1. citation taken from [Vardi, 2015]

Why studying SAT ? – 2

Reduction to SAT : **prove the hardness of new problems**

- **many reductions exist from many problems to SAT**
- **SAT "captures" the difficulty** of many other problems

SAT Solvers are (incredibly) efficients and "user-friendly"

- Can be used as a **black box**
- Can be used as an **open box** (special heuristics, ...)
- Can be **extended** in many ways
  - SAT solvers as Oracles on many close formulas
  - SAT solvers working at an abstract level
  - SAT solvers working on "on the fly" generated constraints
  - ...

Why studying SAT ? – 3

D. Knuth (volume 4, Fascicle 6) :
« The story of satisfiability is the tale of a **triumph of software engineering**, blended with rich doses of **beautiful mathematics**. Thanks to **elegant new data structures and other techniques**, modern SAT solvers are able to deal routinely with practical problems that involve many thousands of variables, although such problems were regarded as hopeless just a few years ago. »

E. Clarke :
« The practical solving of **SAT is a key technology** for computer science in the 21st century. »

Why studying SAT ? – 3

D. Knuth (volume 4, Fascicle 6) :
« The story of satisfiability is the tale of a **triumph of software engineering**, blended with rich doses of **beautiful mathematics**. Thanks to **elegant new data structures and other techniques**, modern SAT solvers are able to deal routinely with practical problems that involve many thousands of variables, although such problems were regarded as hopeless just a few years ago. »

E. Clarke :
« The practical solving of **SAT is a key technology** for computer science in the 21st century. »

Simple is beautiful... But speed is essential

Is it really interesting to study how to implement a CDCL ?

The paradigm shift is essentially due to

- **Tight data structures**
- **Algorithms built upon this data structure**

➔ [The way "Modern" SAT solvers are solving problems has nothing common with a human strategy]

Interesting problems are not toy problems

Beeing fast is the way computers are not so dumb

Logique propositionnelle

Représenter la connaissance

Les **faits** :
soit vrais, soit faux.

Les **variables** :
soit $\top$, soit $\perp$.

Calculer pour raisonner

Si on sait que :

- A implique B
- B implique C

Alors on peut **déduire** :

- A implique C

Si on a :

- $\neg A \vee B$
- $\neg B \vee C$

Alors, par **résolution** :

- $\neg A \vee C$

Cette logique remonte à 2350 ans
« Simple is beautiful »

D'incroyables progrès pratiques et théoriques

Du point de vue pratique

Dans les années 90, les formules traitables étaient de quelques centaines de clauses sur une station de recherche...

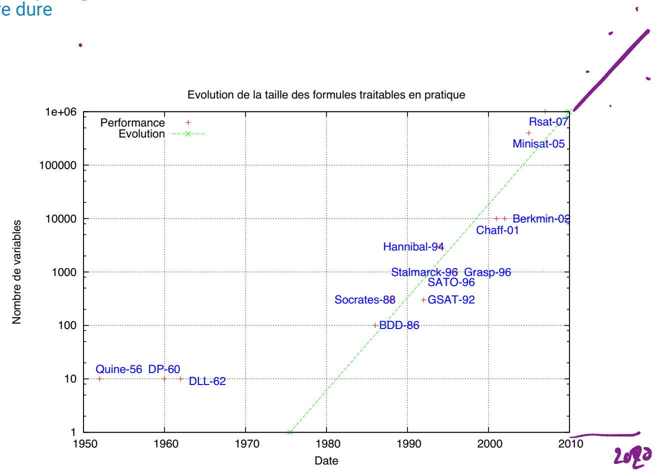
Aujourd'hui, sur un PC *famillial*, on peut résoudre des problèmes de taille industrielle en :

- Electronic Design Automation (EDA)
  - Vérification de Microprocesseurs
  - Génération automatique de tests
- (Bounded) Model Checking (des solveurs incorporés dans des outils de model checking *industriels*)
- Planification
- Utilisé pour prouver des théorèmes de logiques plus expressives (au dessus de NP)

(C'est tout de même un peu plus compliqué : beaucoup de problèmes résistent)

Illustration des progrès

DLL62 : La procédure dure



Attention, la taille des formules n'est pas une indication directe de leur difficulté

Que peut-on faire avec une logique si simple ?

Les faits sont des variables propositionnelles
La connaissance est une formule logique

clauses

clause unaire

CNF

$$
\begin{array}{lcccc}
 & (\neg x_1 & \vee & \neg x_2 & \vee & x_3) \\
\wedge & & & & & \neg x_3 \\
\wedge & (x_1 & \vee & x_2) & & \\
\wedge & & & (x_2 & \vee & x_3)
\end{array}
$$

- Variables : $x_1 \ldots x_3$;
- Littéraux : $x_1, \neg x_1$;
- Clauses : $\neg x_1 \vee \neg x_2 \vee x_3$;
- Formule Σ sous CNF (conjonction de clauses) ;

x_1	x_2	x_3
$\perp$	$\top$	$\perp$

interprétation

Que peut-on demander ?

difficulté

- **SAT** : existe-il une interprétation des variables qui satisfait la formule ?
- **UNSAT** : la théorie est-elle contradictoire ? → Preuve ?
- **PI** : déduire tout ce que l'on peut déduire de Σ. → très I.P.

Le problème SAT

Définition

Entrée : Un ensemble de clauses construites sur un langage propositionnel ayant n variables **Sortie** : Existe-t-il Une affectation des variables évaluant à vrai toutes les clauses (et donc la formule aussi) ?

Exemple (avec des notations différentes) EDA I.A

$$\Sigma_1 = (\neg a \vee b) \wedge (\neg b \vee c) = (a' + b).(b' + c) = \{\neg a \vee b, \neg b \vee c\}$$

CNF

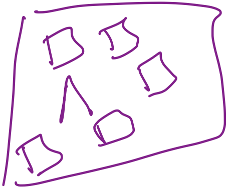
$$\Sigma_2 = \Sigma_1 \wedge a \wedge \neg c = \Sigma_1 \cup \{a, \neg c\}$$

Pour Σ_1, la réponse est oui, pour Σ_2 la réponse est non

$$\Sigma_1 \models \neg a \vee c = \neg(a \wedge \neg c)$$

Why working on CNF? Bec(l)ause!

$(a \wedge \neg b \wedge f)$
$\vee (e \wedge \neg g \wedge h)$ DNF
$\vee$ ... ---



Why considering SAT only on CNF?

(SAT is trivial on DNF!)

- Human-designed systems are conjunctions of properties (in general)
- We want to do symbolic reasoning (not trying all possible solutions)
- There are (too) many ways of applying rules at each step
- We need to restrict the possibilities at each step

But, what about rewritting any formula into CNF?

- Very hard in general
- But, very easy *if we just want to check SAT*

Working on CNF [Tseitin 1968]

Idea : introduce new variables encoding the satifiability of subformulas

Let's say we need to check $SAT(f)$ with

$$f \equiv g \vee h$$

We introduce x_f, x_g and x_h representing the satifiability of f, g and h, respectively

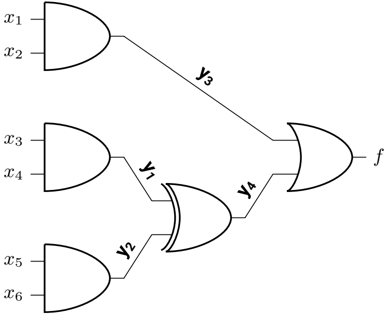
$$(\neg x_f \vee x_g \vee x_h) \wedge (x_f \vee \neg x_g) \wedge (x_f \vee \neg x_h)$$

x_f **encodes the satisfiability of** f**. Easy ! (linear, no blow-up !)**

(Introducing new variables is so powerful, isn't it ?)

It is like naming all the wires in a circuit

$$f = (x_1 \wedge x_2) \vee ((x_3 \wedge x_4) \oplus (x_5 \wedge x_6))$$



Adds y_1, y_2, y_3, y_4, y_f and :

$$\Sigma_f \equiv \begin{pmatrix} (y_f \leftrightarrow y_3 \wedge y_4) \\ \wedge\, (y_4 \leftrightarrow y_1 \oplus y_2) \\ \wedge\, (y_1 \leftrightarrow x_3 \wedge x_4) \\ \wedge\, (y_2 \leftrightarrow x_5 \wedge x_6) \\ \wedge\, (y_3 \leftrightarrow x_1 \wedge x_2) \\ \wedge\, (y_f) \end{pmatrix}$$

f is satisfiable iff Σ_f is

At the heart of most procedures : resolution

$(x \vee A) \wedge (\neg x \vee B)$

The Resolution Rule (Cut) [Gentzen 1934, Robinson 1965]

> Let $c_1 = (x \vee \underbrace{a_1 \vee \ldots a_n}_{A})$ and $c_2 = (\neg x \vee \overbrace{b_1 \vee \ldots b_m}^{B})$
>
> $$c = (a_1 \vee \ldots a_n \vee b_1 \vee \ldots b_m)$$
>
> is obtained by resolution on x between c_1 and c_2

It is a particular case of the following **deduction rule** :

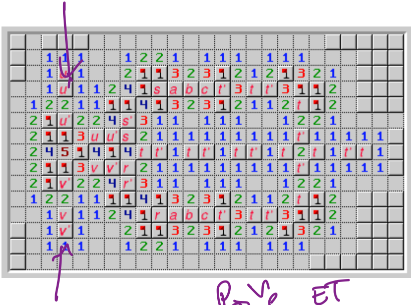
if $a \to b$ **and** $b \to c$ **then** $a \to c$

$\neg a \vee b \quad \neg b \vee c \quad \neg a \vee c$

In general, SAT solvers are only using this rule
(but many, many times per second)

Finding which ones to trigger is the secret of efficient SAT solvers

D'où vient la difficulté de SAT ?



Porte ET

- **Facile** si trouver chaque mine reste relativement local
- **Difficile** si une mine posée à un endroit a un impact éloigné

Déminer est un problème aussi difficile que SAT

La brique SAT

Le théorème de Cook [71] : SAT est NP-Complet

C'est la brique de base de la théorie de la complexité.

- Très étudié pour ses limitations théoriques
- Très étudié pour sa résolution pratique

Un problème semble difficile à résoudre ?

Dans les années 80

« Réduisez SAT à ce problème :
cela montrera que personne ne peut le résoudre ! »

Au 21ième siècle...

« Réduisez SAT à ce problème : résolvez-le en pratique »

Taille de l'espace de recherche

Système solaire

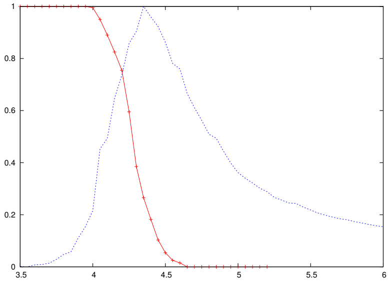
189 vars

Pèse le nombre de particules en nombre d'interprétations possibles

Évolution des performances sur les aléatoires

Un exemple pour les autres domaines de recherche

- Années 1980 :
  Peu de vrais problèmes ;
- Années 1990 :
  Se concentrer autour du seuil ;
- Années 2000 :
  Utilisation de méthodes de physique statistique ;

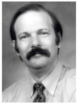


Bilan, aujourd'hui

Facile (SAT) jusque $r = 4.25$, Difficile (SAT) entre $r = 4.25$ et $r = 4.267$
Extrêmement Difficile (UNSAT) après $r = 4.267$

Des progrès pratiques et théoriques

- Vérification de Microprocesseurs, Planifications, Contraintes, Biologie, ...
- Théorèmes de logiques plus expressives (au dessus de NP)



Moshe Vardi

« Les progrès les plus importants à attendre en logique pourraient bien venir de SAT »

Edmund Clarke

« La résolution pratique du problème SAT est une technologie clé pour l'informatique du 21ième siècle »

Quelques questions pour l'intuition

Pas si difficile

- (Mots Croisés) Débarrasser le pétrole de ses impuretés
- Pouvez-vous donner deux nombres qui divisent 49 ? 143 ? 8633 ?
- Montrez la *conjecture de Fermat* suivante :
  Tous les nombres $F_n = 2^{2^n} + 1$ sont premiers (n entier naturel).

- (Mots Croisés) Évite les suites mais se lève à l'Est.
- Pouvez-vous trouver deux nombres premiers qui divisent 57 ? 143 ? 8637 ?
- Est-ce que ce *théorème*, toujours de Fermat, est vrai : Peut-on trouver des entiers tels que $x^n + y^n = z^n$ pour n fixé et $n > 2$?

P comme « Facile » ?

Classe de complexité P

La classe P est l'ensemble des problèmes de décision pouvant être résolus en temps polynômial.

En terme de fonction calculable, ces problèmes sont caractérisés par la réponse à la question « Que vaut $f(x)$? », avec f calculable en temps polynômial par rapport à la taille de la donnée x.

$NP \neq$ Non Polynômial !

Classe de complexité NP

La classe NP est l'ensemble des problèmes de décision pouvant être résolus en temps polynômial de manière non déterministe.

La classe de complexité NP correspond à l'ensemble des problèmes de décision s'exprimant logiquement comme

$$\exists x . f(x)$$

« existe-t-il un x tel que l'on ait $f(x)$? », avec f une fonction calculable en temps polynômial.

C'est la classe

$\forall x . f(x)$

Classe de complexité $CoNP$

La classe $CoNP$ est l'ensemble des problèmes dont le problème complémentaire appartient à NP.

Réduction α_p

Un problème $\mathcal{P}_1$ est polynômialement réductible à un autre problème $\mathcal{P}_2$ si et seulement si il existe une fonction f calculable en temps polynômial telle que pour tout problème de décision x de $\mathcal{P}_1$, $\mathcal{P}_2$ répond identiquement sur l'entrée $f(x)$. On le note $\mathcal{P}_1 \; \alpha_p \; \mathcal{P}_2$.

La réduction α_p pour la NP-Complétude

Classe NP-complet

Soit $\mathcal{P}$ une classe de problèmes. On dit que $\mathcal{P}$ est NP-Complet si et seulement si (1) $\mathcal{P} \in NP$; (2) $\forall \mathcal{P}' \in NP,\ \mathcal{P}' \alpha_p \mathcal{P}$.

Classe CoNP-complet

Soit $\mathcal{P}$ une classe de problèmes. On dit que $\mathcal{P}$ est $CoNP$-Complet si et seulement si (1) $\mathcal{P} \in CoNP$; (2) $\forall \mathcal{P}' \in CoNP,\ \mathcal{P}' \alpha_p \mathcal{P}$.

Que retenir de tout ça ?

La notion NP est très fortement reliée à la garantie de l'existence d'un certificat court, et facilement vérifiable. En pratique, on peut observer d'énormes différences de performances.

À noter : les algorithmes « modernes » pour SAT ne montrent aucune différence de performance pour les problèmes SAT ou les problèmes UNSAT.

Utilisation de systèmes de preuves

À l'attaque de P$\neq$NP

Qu'est-ce qu'un système de preuve ?

Un moyen (automatique) permettant de vérifier une preuve donnée.

Augmenter leur pouvoir petit à petit

À l'attaque de P$\neq$NP

Si le plus puissant des systèmes de preuve (à l'équivalence près) butte sur un problème donnée, alors P$\neq$NP.
S'il existe un problème pour lequel la preuve minimale UNSAT est exponentiellement longue, alors on a gagné (pour le plus puissant des systèmes de preuve).

Feuille de route proposée pour montrer P$\neq$NP

Montrer NP $\neq$ Co-NP comme suit :

1. Prendre le moins puissant des systèmes de preuve
2. Montrer qu'il existe des problèmes difficiles pour lui
3. Augmenter la puissance du système de preuve
4. Boucler

D'autres problèmes, d'autres encodages

- Sudoku (pas encore un problème industriel)
- Recherche d'haplotype dans une population
- (Bounded) Model Checking
- Logiques modales / Logiques temporelles
- Raisonnement sur les ontologies
- Organisation de tournois de golf
- ...

Pourquoi beaucoup s'y mettent ?

- Soit ça marche, et leur problème est résolu
- Soit ça ne marche pas et leur problème devient un benchmark de valeur

Si en plus des industriels sont intéressés...

Exemple de raisonnement via SAT

Le petit Monde du monstre

$$M_{i,j} \to (O_{i-1,j} \wedge O_{i+1,j} \wedge O_{i,j+1} \wedge O_{i,j-1})$$

Odeur		*Vent*	**Trou**
Monstre *Odeur*	*Vent* *Odeur* **Or**	**Trou**	*Vent*
$O_{1,2} = T$ *Odeur* $\neg M$ $\neg T$		*Vent*	
$\neg M$ x **Départ** $\neg T$	$\neg M$ $\neg T$ *Vent*	**Trou**	*Vent*

$$T_{i,j} \to (\; \vdots \;)$$

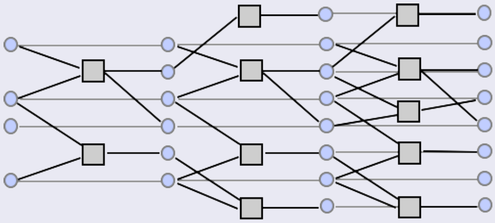
Une première révolution

37 ans de STRIPS, une révolution : GraphPlan

Formulation STRIPS

```
MOVE(x,y,z)
PRE: CLEAR(x), ON(x,y), CLEAR(z)
ADD: CLEAR(y), ON(x,z)
DEL: CLEAR(z), ON(x,y)
```

Principe (graphiques) de GraphPlan [95]



Principes récursifs de STRIPS

1. Empiler le but à atteindre
2. Effacer le haut de la pile si c'est un fait valide
3. Si c'est une conjonction, empiler tous ses faits
4. Sinon chercher à unifier une action dont l'un des effet est justement le fait recherché, l'ajouter à la pile avec toutes ses préconditions
5. Si c'est une action, l'executer
6. Boucler tant que la pile n'est pas vide

Une première révolution

37 ans de STRIPS, une révolution : GraphPlan

1. Génère un GraphPlan de longueur k
2. Transforme les contraintes du graphe en clauses. Chaque instance d'une action ou d'un fait à chaque moment est une proposition.
3. Utilise un solveur SAT
4. Si on ne trouve pas de solution (UNSAT ou TimeOut), incrémente k et recommence
5. Si on trouve une solution, transforme la solution en solution de planificiation
6. Simplifie la solution.

Recette de planification

Un peu d'imagination

Supposons qu'un individu veuille préparer un petit-déjeuner surprise à sa fiançée, accompagné d'un cadeau à emballer. Le tout alors qu'elle dort encore.

Etat initial : $poubelles, mainPropres, silence$
Etat Final : $dej, cadeau, nonpoubelles$

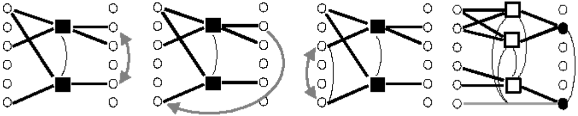
Action	Préconditions	Effets
Cuisiner()	mainPropres	dej
Envelopper()	silence	cadeau
Sortir()	aucunes	non poubelles, non mainPropres
Chariot()	aucunes	non poubelles, non silence

Encodage surprise

Idée	Encodage
Conditions initiales	$poubelles^0 \wedge mainPropres^0 \wedge silence^0$
Conditions finales, après k étapes	$dej^2 \wedge cadeau^2 \wedge \neg poubelles^2$
Exemple d'encodage (les faits qui changent)	$Cuisiner^0 \rightarrow mainPropres^0 \wedge dej^1$ $Cuisiner^1 \rightarrow mainPropres^1 \wedge dej^2$
Ne pas oublier ceux qui ne changent pas	$Cuisiner^0 \wedge poubelles^0 \rightarrow poubelles^1$
Tout changement d'état s'explique	$poubelles^0 \wedge \neg poubelles^1 \rightarrow Chariot^0 \vee Sortir^0$
Faire au moins une action à chaque étape	$Cuisiner^0 \vee Envelopper^0 \vee Sortir^0 \vee Chariot^0$
Empêcher les exclusions mutuelles implicites	$\neg Cuisiner^0 \vee \neg Sortir^0$

D'autres idées

Des exclusions mutuelles



Quelques autres idées

- Découper les actions avec beaucoup d'arguments
  $embarquer(voilier, 8h15, Runion, capitaine)$
- Encoder le temps et/ou diverses mesures diverses en log

1992 la recherche locale donne de bons résultats
2001 bénéficie de zchaff, berkmin et siege
2004 premier prix de la compétition (domaines prop.)
2006 idem, ex aequo avec un autre SAT-planificateur

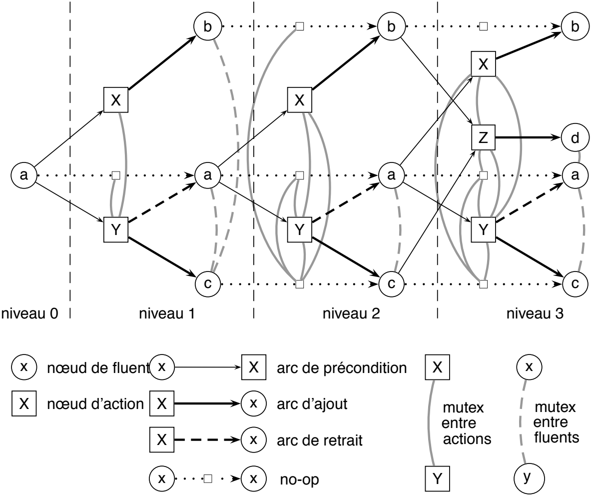
A ouvert la porte aux encodages SAT de problèmes *industriels*.

Un exemple de GraphPlan

C’est ce schéma qui sera encodé en SAT

Problème :
$\Pi = \langle F, A, I, B\rangle$
- F = {a, b, c, d}
- A = {X, Y, Z}
- I = {a}
- B = {d}

Actions :
- X = ⟨{a}, {b}, {}⟩
- Y = ⟨{a}, {c}, {a}⟩
- Z = ⟨{b, c}, {d}, {}⟩



Principes du Model Checking

Étant donné un automate décrivant les états possibles d'un système

Vérifier qu'un état n'est jamais atteint

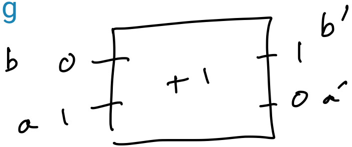
C'est généralement la recherche de bug. Un état spécial « erreur » est utilisé dans la modélisation.

Vérifier que le système ne bloque jamais

Tout état doit être accessible depuis tout état, à n'importe quel moment du futur (impossible de construire une boucle infinie).

A des liens étroits avec les logiques temporelles.
Avant SAT, les BDD étaient utilisés pour résoudre ces problèmes.

Principes du (Bounded) Model Checking



Comme pour la planification, on se borne à un k fixé, qu'on incrémente à loisir.

- L'automate est représenté par la fonction caractéristique T de ses transitions entre états.
  Exemple (2-bit additionneur) : $(a' \leftrightarrow \neg a) \wedge (b' \leftrightarrow a \oplus b)$
- La propriété à vérifier est une formule logique sur l'état du système
  Exemple : $a \wedge b$ (l'état (11) peut-il être atteint ?)
- L'état initial sera une affectation le codant au temps 0

Déroulement de boucles

Vérifions si l'état (11) est atteignable en deux itérations dans notre 2-bit additionneur, depuis (00)

$I(s_0) = \neg a_0 \wedge \neg b_0$
$T(s_0, s_1) = (a_1 \leftrightarrow \neg a_0) \wedge (b_1 \leftrightarrow a_0 \oplus b_0)$
$T(s_1, s_2) = (a_2 \leftrightarrow \neg a_1) \wedge (b_2 \leftrightarrow a_1 \oplus b_1)$
$p(s_2) = a_2 \wedge b_2$

$p(s_0) = a_0 \wedge b_0$
$p(s_1) = a_1 \wedge b_1$

Au final

$(\neg a_0 \wedge \neg b_0) \wedge ((a_1 \leftrightarrow \neg a_0) \wedge (b_1 \leftrightarrow a_0 \oplus b_0)) \wedge ((a_2 \leftrightarrow \neg a_1)$
$\wedge (b_2 \leftrightarrow a_1 \oplus b_1)) \wedge (a_2 \wedge b_2)$ est-elle satisfiable ?

Quelques limites
Quand même...

- Perte de la structure
- Difficulté d'encoder des relations entre nombres (ensembles non discrets ou grands)
- Difficulté d'encoder certaines contraintes implicites
- Difficulté d'encoder certains problèmes de math

- Comportement des démonstrateurs... non encore compris

Unbounded Model Checking

$$I \wedge T_1 \wedge T_2 \wedge \ldots \wedge T_k \wedge BUG_k$$

How to ensure that BUG is unreachable ?

Idea : find an invariant Inv s.t. BUG is not reachable in $k > 0$ steps

- Inv characterizes an over approximation of the reachable states in j steps :

$$I \wedge T_1 \wedge \cdots \wedge T_j \rightarrow Inv$$

- Inv is an inductive property :

$$Inv \wedge T_1 \rightarrow Inv_1$$

- BUG is not reachable from Inv in k steps :

$$Inv \wedge T_1 \wedge T_2 \wedge \ldots \wedge T_k \wedge BUG_k \equiv \perp$$

- Incremental SAT Solving / Proof Analysis

The Erdös Discrepancy Problem (1932)

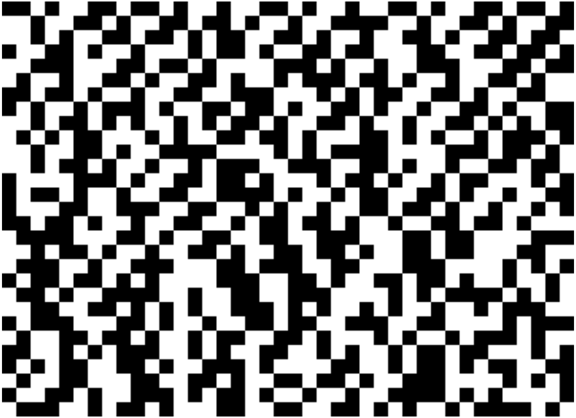
- Infinite series of +1 and -1 : $< -1, 1, 1, -1 - 1, 1, 1, \ldots >$
- $\forall C\ \exists k, d\ t.q.\ \ |\sum_{i=1}^{k} x_{i.d}| \geq C$

$$
\begin{array}{ccccccccccc|c}
+ & - & - & + & + & - & - & - & + & + & - & -1 \\
 & - & & + & & - & & - & & + & & -1 \\
 & & - & & & - & & & + & & & -1 \\
 & & & + & & & & - & & & & 0 \\
 & & & & + & & & & & + & & +2
\end{array}
$$

- Proven in 2014 for $C = 2$ (k=1161)
- The *proof* : UNSAT certificate (trace) from Glucose (13 Gb) [2]
- General case proven two years later by Terence Tao (previous proof considered as the biggest mathematical proof ever by T. Tao).

2. `cgi.csc.liv.ac.uk/~konev/SAT14/`

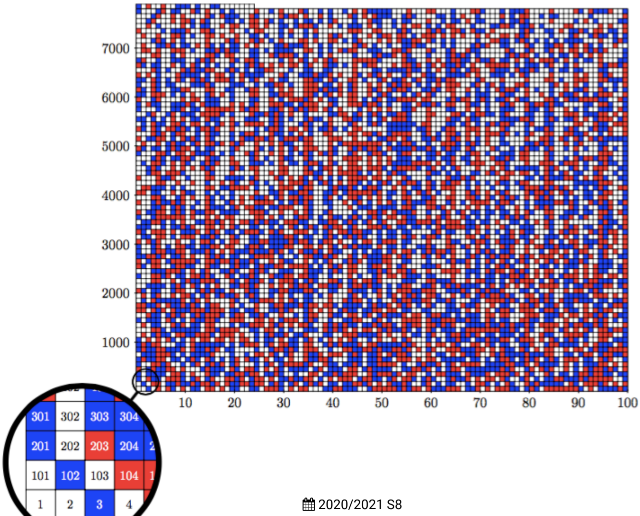
Solution for C=2, 1160 steps



(For C=3, maximum solution is not yet known)

The "biggest proof" in the world

Boolean Pythagorean triples problem
Is it possible to colorize the n integers $\leq n$ in two colors s.t. no triplet (a, b, c) is $a^2 + b^2 = c^2$ monochromatic ?



The "biggest proof" in the world

No solution for n=7825

- Open question since 20 years
- 10^{2300} possible canidates
- SAT encoding
- Original problem splitted in 1,000,000 subproblems
- 800 CPUs

Proof is 200Tb long (Glucose's output)

- In practice the proof is not really kept

Davis & Putnam : the firsts SAT steps

1958 : Hilary Putnam and Martin Davis look for funding their research around propositional logic

« What we're interested in is good algorithms for propositional calculus » (NSA)

Before that, only inefficient methods (truth tables, …)

First papers

- *Computational Methods in The Propositional calculus* [Davis Putnam 1958] [3]
- *A Computing Procedure for Quantification Theory* [Davis Putnam 1960]

3. Rapport interne NSA

1960, already a first (kind of) competition !

« *The superiority of the present procedure (i.e. DP) over those previously available is indicated in part by the fact that a formula on which Gilmore's routine for the IBM 704 causes* **the machine to compute for 21 minutes** *without obtaining a result was worked successfully by* **hand computation using the present method in 30 minutes** »

[Davis et Putnam 1960], page 202.

Principles of DP-60

DP-60 : forgets variables one after the other

Example : forgets x_1.

$$
\begin{array}{l}
x_1 \vee x_4 \\
\overline{x_1} \vee x_4 \vee x_{14} \\
\overline{x_1} \vee \overline{x_3} \vee \overline{x_8} \\
x_1 \vee x_8 \vee x_{12} \\
x_1 \vee x_5 \vee \overline{x_9} \\
x_2 \vee x_{11} \\
\overline{x_3} \vee \overline{x_7} \vee x_{13} \\
\overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
x_8 \vee \overline{x_7} \vee \overline{x_9}
\end{array}
$$

Principles of DP-60

DP-60 : forgets variables one after the other

Example : forgets x_1.

$$x_1 \lor x_4$$
$$x_1 \lor x_8 \lor x_{12}$$
$$x_1 \lor x_5 \lor \overline{x_9}$$

$$\overline{x_1} \lor x_4 \lor x_{14}$$
$$\overline{x_1} \lor \overline{x_3} \lor \overline{x_8}$$

$$x_2 \lor x_{11}$$
$$\overline{x_3} \lor \overline{x_7} \lor x_{13}$$
$$\overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9$$
$$x_8 \lor \overline{x_7} \lor \overline{x_9}$$

Principles of DP-60

DP-60 : forgets variables one after the other

Example : forgets x_1.

$x_1 = \top$?

$x_1 = \bot$?

$$x_1 \vee \begin{pmatrix} x_4 \\ x_8 \vee x_{12} \\ x_5 \vee \overline{x_9} \end{pmatrix}$$

$$\overline{x_1} \vee \begin{pmatrix} x_4 \vee x_{14} \\ \overline{x_3} \vee \overline{x_8} \end{pmatrix}$$

$$\begin{bmatrix} x_2 \vee x_{11} \\ \overline{x_3} \vee \overline{x_7} \vee x_{13} \\ \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\ x_8 \vee \overline{x_7} \vee \overline{x_9} \end{bmatrix}$$

Principles of DP-60

DP-60 : forgets variables one after the other

Example : forgets x_1.

$\to$ CNF !

distribution

$$\begin{pmatrix} x_4 \\ x_8 \vee x_{12} \\ x_5 \vee \overline{x_9} \end{pmatrix} \vee \begin{pmatrix} x_4 \vee x_{14} \\ \overline{x_3} \vee \overline{x_8} \end{pmatrix}$$

$$\begin{array}{l} x_2 \vee x_{11} \\ \overline{x_3} \vee \overline{x_7} \vee x_{13} \\ \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\ x_8 \vee \overline{x_7} \vee \overline{x_9} \end{array}$$

Principles of DP-60

DP-60 : forgets variables one after the other

Example : forgets x_1.

$x_4 \vee x_{14}$

$x_4 \vee \overline{x_3} \vee \overline{x_8}$

$x_8 \vee x_{12} \vee x_4 \vee x_{14}$

$x_5 \vee \overline{x_9} \vee x_4 \vee x_{14}$

$x_5 \vee \overline{x_9} \vee \overline{x_3} \vee \overline{x_8}$

$x_2 \vee x_{11}$

$\overline{x_3} \vee \overline{x_7} \vee x_{13}$

$\overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$

$x_8 \vee \overline{x_7} \vee \overline{x_9}$

Principles of DP-60

DP-60 : forgets variables one after the other

Example : forgets x_1.

x_1 n'est plus là.

$$
\begin{array}{l}
x_4 \lor x_{14} \\
x_4 \lor \overline{x_3} \lor \overline{x_8} \\
x_8 \lor x_{12} \lor x_4 \lor x_{14} \\
x_5 \lor \overline{x_9} \lor x_4 \lor x_{14} \\
x_5 \lor \overline{x_9} \lor \overline{x_3} \lor \overline{x_8} \\
x_2 \lor x_{11} \\
\overline{x_3} \lor \overline{x_7} \lor x_{13} \\
\overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \\
x_8 \lor \overline{x_7} \lor \overline{x_9}
\end{array}
$$

Variable elimination – on a trivial example

$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(x_3 \vee \neg x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$x_3 \quad \models$

$\top$

- Formula is SAT (and x_3 is True in all models)

Variable elimination – on a trivial example

$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(x_3 \vee \neg x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$x_3 \quad \models$

$\top$

- Formula is SAT (and x_3 is True in all models)

Variable elimination – on a trivial example

$(x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models$

$(\neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models$

$(x_3 \lor \neg x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models$

$(x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models$

$x_3 \models$

$\top$

- Formula is SAT (and x_3 is True in all models)

Variable elimination – on a trivial example

$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(x_3 \vee \neg x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$x_3 \quad \models$

$\top$

- Formula is SAT (and x_3 is True in all models)

Variable elimination – on a trivial example

$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(x_3 \vee \neg x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$(x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \quad \models$

$x_3 \quad \models$

$\top$

- Formula is SAT (and x_3 is True in all models)

Variable elimination – on a trivial example

$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \models$

$(\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \models$

$(x_3 \vee \neg x_3) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \models$

$(x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \models$

$x_3 \models$

$\top$

- Formula is SAT (and x_3 is True in all models)

DP-60 → utilisé en I.A
→ raisonnement.

trop gourmand en mémoire.

→ DPLL-62

1962-2001 : DPLL rules the world

Systematically explore the space of partial models (backtrack)

- Choose a literal
- Try to find a solution with this literal set to True
- If it is not possible :
  Finds a solution with this literal set to False

Backtrack search on partial models
Systematic (ordered) exploration ensures completeness

1962-2001 : DPLL rules the world

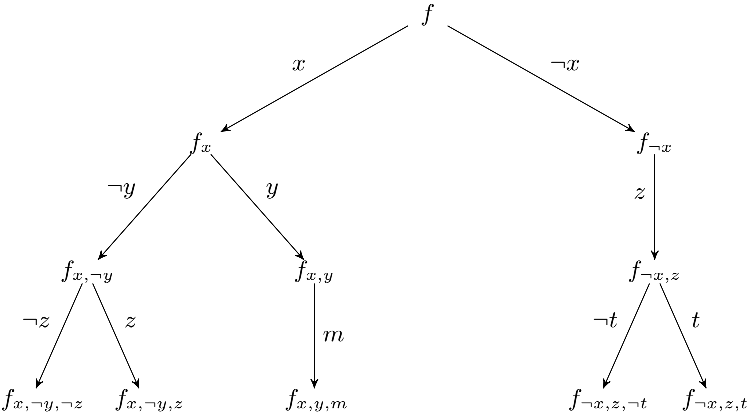
Systematically explore the space of partial models (backtrack)

1

- Choose a literal
- Try to find a solution with this literal set to True
- If it is not possible :
  Finds a solution with this literal set to False

Backtrack search on partial models
Systematic (ordered) exploration ensures completeness

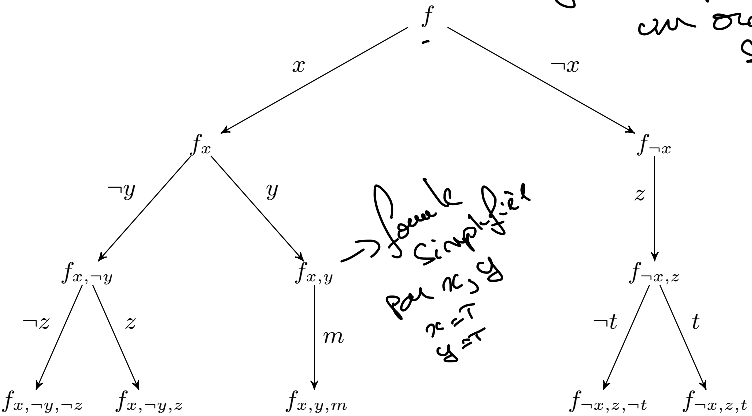
Backtrack search



- How to choose the right literal to branch on ?
- First search for a model or a contradiction ?

Backtrack search

La complétude de la recherche est garantie par un ordre systématique



→ formule simplifiée par x, y
$x = \top$
$y = \top$

- How to choose the right literal to branch on ?
- First search for a model or a contradiction ?

An example of DPLL

Formula	Simplified Formula
$x_1 \vee x_4$	$x_1 \vee x_4$
$\overline{x_1} \vee x_4 \vee x_{14}$	$\overline{x_1} \vee x_4 \vee x_{14}$
$x_1 \vee \overline{x_3} \vee \overline{x_8}$	$x_1 \vee \overline{x_3} \vee \overline{x_8}$
$x_1 \vee x_8 \vee x_{12}$	$x_1 \vee x_8 \vee x_{12}$
$x_2 \vee x_{12}$	$x_2 \vee x_{12}$
$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$	$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$
$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$	$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$
$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$	$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Partial Model

Lev.	Lit.	Back ?

$x1$ appears in 4 clauses and 1 binary clause

An example of DPLL

$\overline{x_5}$

Formula

$x_1 \vee x_4$
$\overline{x_1} \vee x_4 \vee x_{14}$
$x_1 \vee \overline{x_3} \vee \overline{x_8}$
$x_1 \vee x_8 \vee x_{12}$
$x_2 \vee x_{12}$
$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$
$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$
$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Simplified Formula

$x_1 \vee x_4$
$\overline{x_1} \vee x_4 \vee x_{14}$
$x_1 \vee \overline{x_3} \vee \overline{x_8}$
$x_1 \vee x_8 \vee x_{12}$
$x_2 \vee x_{12}$
$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$
$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$
$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Partial Model

Lev.	Lit.	Back ?
1	$\overline{x_1}$	(d)

x_4 appears in 1 unary clause

An example of DPLL

Formula	Simplified Formula
$x_1 \vee x_4$	$x_1 \vee x_4$
$\overline{x_1} \vee x_4 \vee x_{14}$	$\overline{x_1} \vee x_4 \vee x_{14}$
$x_1 \vee \overline{x_3} \vee \overline{x_8}$	$x_1 \vee \overline{x_3} \vee \overline{x_8}$
$x_1 \vee x_8 \vee x_{12}$	$x_1 \vee x_8 \vee x_{12}$
$x_2 \vee x_{12}$	$x_2 \vee x_{12}$
$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$	$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$
$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$	$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$
$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$	$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Partial Model

Lev.	Lit.	Back ?
1	$\overline{x_1}$	(d)
+	x_4	

x_3 appears in 3 clauses incl. 1 (new) binary clause

An example of DPLL

Formula

$x_1 \vee x_4$
$\overline{x_1} \vee x_4 \vee x_{14}$
$x_1 \vee \overline{x_3} \vee \overline{x_8}$
$x_1 \vee x_8 \vee x_{12}$
$x_2 \vee x_{12}$
$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$
$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$
$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Simplified Formula

$x_1 \vee x_4$
$\overline{x_1} \vee x_4 \vee x_{14}$
$x_1 \vee \overline{x_3} \vee \overline{x_8}$
$x_1 \vee x_8 \vee x_{12}$
$x_2 \vee x_{12}$
$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$
$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$
$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Partial Model

Lev.	Lit.	Back ?
1	$\overline{x_1}$	(d)
+	x_4	
2	x_3	(d)

$\overline{x_8}$ appears in one unary clause

An example of DPLL

Formula	Simplified Formula
$x_1 \vee x_4$	$x_1 \vee x_4$
$\overline{x_1} \vee x_4 \vee x_{14}$	$\overline{x_1} \vee x_4 \vee x_{14}$
$x_1 \vee \overline{x_3} \vee \overline{x_8}$	$x_1 \vee \overline{x_3} \vee \overline{x_8}$
$x_1 \vee x_8 \vee x_{12}$	$x_1 \vee x_8 \vee x_{12}$
$x_2 \vee x_{12}$	$x_2 \vee x_{12}$
$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$	$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$
$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$	$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$
$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$	$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Partial Model

Lev.	Lit.	Back ?
1	$\overline{x_1}$	(d)
+	x_4	
2	x_3	(d)
+	$\overline{x_8}$	

x_{12} appears in 1 unary clause

An example of DPLL

Formula

$x_1 \vee x_4$

$\overline{x_1} \vee x_4 \vee x_{14}$

$x_1 \vee \overline{x_3} \vee \overline{x_8}$

$x_1 \vee x_8 \vee x_{12}$

$x_2 \vee x_{12}$

$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$

$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$

$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Simplified Formula

x_{13}

$x_7 \vee \overline{x_{13}}$

$\overline{x_7}$

Partial Model

Lev.	Lit.	Back ?
1	$\overline{x_1}$	(d)
+	x_4	
2	x_3	(d)
+	$\overline{x_8}$	
+	x_{12}	

x_{13}, $\overline{x_7}$ appear in unary clauses

An example of DPLL

Formula	Simplified Formula
$x_1 \vee x_4$	$x_1 \vee x_4$
$\overline{x_1} \vee x_4 \vee x_{14}$	$\overline{x_1} \vee x_4 \vee x_{14}$
$x_1 \vee \overline{x_3} \vee \overline{x_8}$	$x_1 \vee \overline{x_3} \vee \overline{x_8}$
$x_1 \vee x_8 \vee x_{12}$	$x_1 \vee x_8 \vee x_{12}$
$x_2 \vee x_{12}$	$x_2 \vee x_{12}$
$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$	$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$
$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$	$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$
$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$	$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Partial Model

Lev.	Lit.	Back ?
1	$\overline{x_1}$	(d)
+	x_4	
2	x_3	(d)
+	$\overline{x_8}$	
+	x_{12}	
+	x_{13}	

x_7, $\overline{x_7}$ appear in unary clauses

An example of DPLL

Formula

$x_1 \vee x_4$
$\overline{x_1} \vee x_4 \vee x_{14}$
$x_1 \vee \overline{x_3} \vee \overline{x_8}$
$x_1 \vee x_8 \vee x_{12}$
$x_2 \vee x_{12}$
$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$
$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$
$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

→ clause vide

Simplified Formula

$x_1 \vee x_4$
$\overline{x_1} \vee x_4 \vee x_{14}$
$x_1 \vee \overline{x_3} \vee \overline{x_8}$
$x_1 \vee x_8 \vee x_{12}$
$x_2 \vee x_{12}$
$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$
$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$
$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Partial Model

Lev.	Lit.	Back ?
1	$\overline{x_1}$	(d)
+	x_4	
2	x_3	(d)
+	$\overline{x_8}$	
+	x_{12}	
+	x_{13}	
+	$\overline{x_7}$	

Conflict ! Undo everything until last decision

An example of DPLL

Formula

$x_1 \vee x_4$

$\overline{x_1} \vee x_4 \vee x_{14}$

$x_1 \vee \overline{x_3} \vee \overline{x_8}$

$x_1 \vee x_8 \vee x_{12}$

$x_2 \vee x_{12}$

$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$

$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$

$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

Simplified Formula

$x_1 \vee x_4$

$\overline{x_1} \vee x_4 \vee x_{14}$

$x_1 \vee \overline{x_3} \vee \overline{x_8}$

$x_1 \vee x_8 \vee x_{12}$

$x_2 \vee x_{12}$

$\overline{x_3} \vee \overline{x_{12}} \vee x_{13}$

$\overline{x_3} \vee x_7 \vee \overline{x_{13}}$

$x_8 \vee \overline{x_7} \vee \overline{x_{12}}$

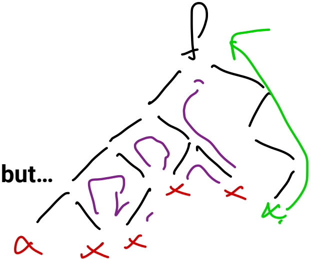
Partial Model

Lev.	Lit.	Back ?
1	$\overline{x_1}$	(d)
+	x_4	
*	$\overline{x_3}$	

Now, $\overline{x_3}$ is not a decision

A very simple procedure ?

Very simple to write a backtrack search but…



Where to branch ?

- **Mistakes at the top of the tree are dramatic !**
- **(almost) As many nodes where to decide than where to explore**
- **A perfect branching scheme is NP-Hard**

To try or to think ?

En 2001, arrivée de la recherche avec apprentissage

(On ne parle pas d'apprentissage comme dans "Machine Learning" ici)

L'idée est d'apprendre de chaque échec, au bout de chaque branche de DPLL.

Le solveur SAT va apprendre une clause à chaque retour arrière

Mieux : apprendre la clause va le forcer à revenir en arrière

Mieux, mieux : le retour arrière remonte plus haut dans l'arbre de recherche

Mieux ++++ : encore d'autres mécanismes maintenant possible !

Années 2000, entrée de SAT dans l'ère industrielle

- Importance des propagations unitaires,
- Coûteux de maintenir des heuristiques bien informées,
- Le compromis refléchir/essayer n'est pas clair
  - Heuristiques de plus en plus coûteuses
  - Les heuristiques se confondent avec de la recherche
  - A contrario, sur certains problèmes, des heuristiques simplistes peuvent donner de bons résultats...
- Grasp a porté à SAT les mécanismes d'apprentissage

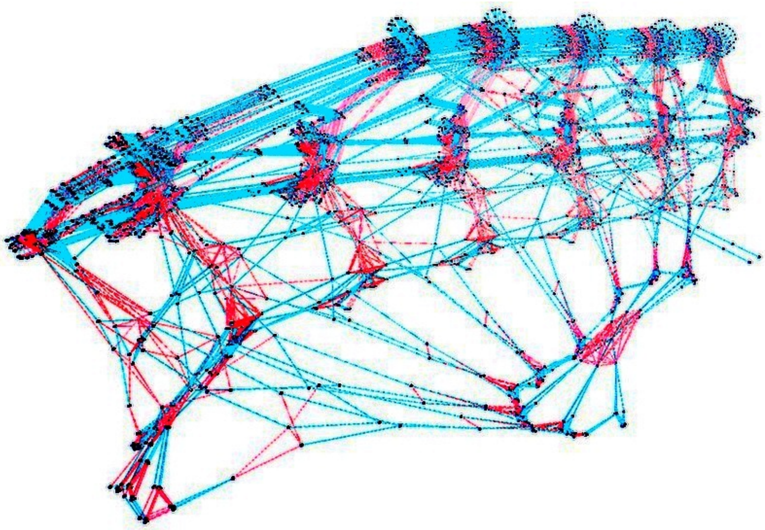
Il ne manque plus que des problèmes... industriels

Années 2000, entrée de SAT dans l'ère industrielle

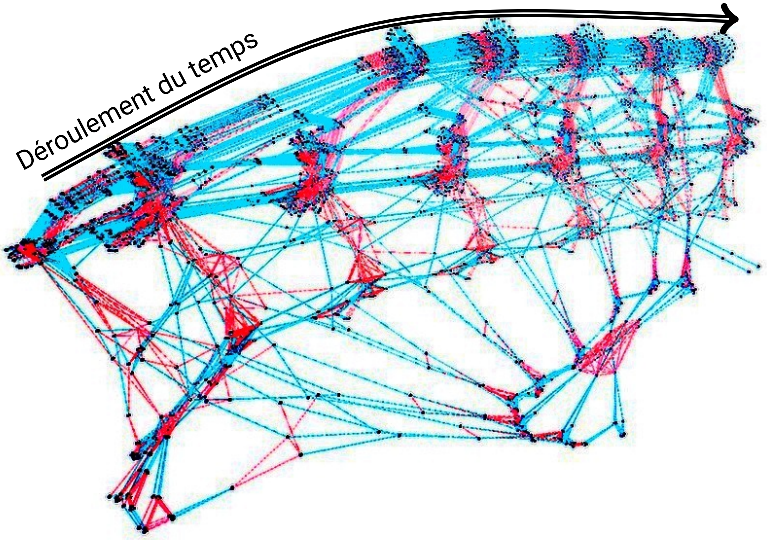
- Importance des propagations unitaires,
- Coûteux de maintenir des heuristiques bien informées,
- Le compromis refléchir/essayer n'est pas clair
  - Heuristiques de plus en plus coûteuses
  - Les heuristiques se confondent avec de la recherche
  - A contrario, sur certains problèmes, des heuristiques simplistes peuvent donner de bons résultats...
- Grasp a porté à SAT les mécanismes d'apprentissage

Il ne manque plus que des problèmes... industriels

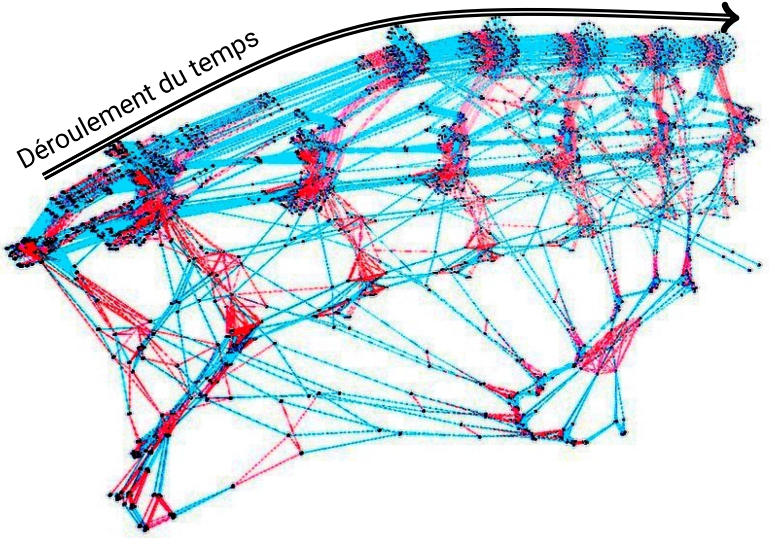
Représentation d'un problème issu de IBM



Représentation d'un problème issu de IBM

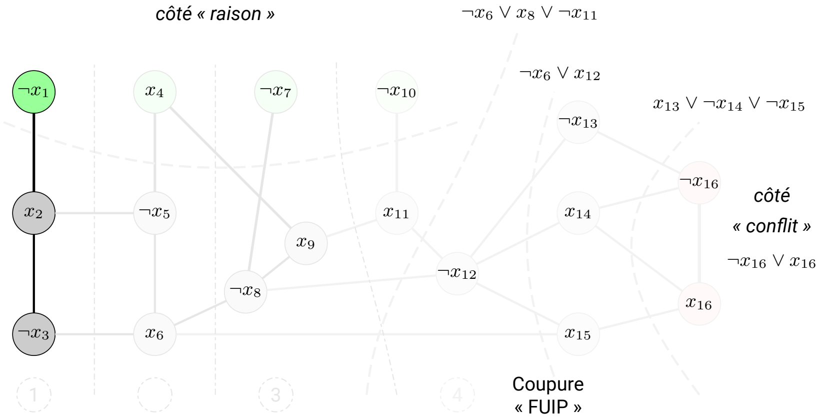


Représentation d'un problème issu de IBM



Présence de backdoors

Le graphe d'implication



$\neg x_2 \vee \neg x_3$

$\neg x_4 \vee \neg x_5$
$x_5 \vee x_6$

$x_7 \vee \neg x_6 \vee \neg x_8$
$\neg x_4 \vee x_8 \vee x_9$

$x_{10} \vee \neg x_9 \vee x_{11}$
$\neg x_{11} \vee x_8 \vee \neg x_{12}$
$x_{12} \vee \neg x_{13}$
$x_{12} \vee x_{14}$

$\neg x_6 \vee x_{12} \vee x_{15}$
$x_{13} \vee \neg x_{14} \vee \neg x_{16}$
$\neg x_{15} \vee \neg x_{14} \vee x_{16}$

Le graphe d'implication

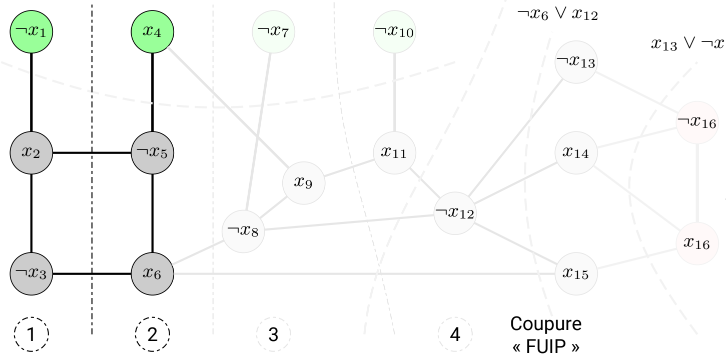
côté « raison »

$x_1 \vee \neg x_4 \vee x_7 \vee x_{10}$

$\neg x_6 \vee x_8 \vee \neg x_{11}$

$\neg x_6 \vee x_{12}$

$x_{13} \vee \neg x_{14} \vee \neg x_{15}$

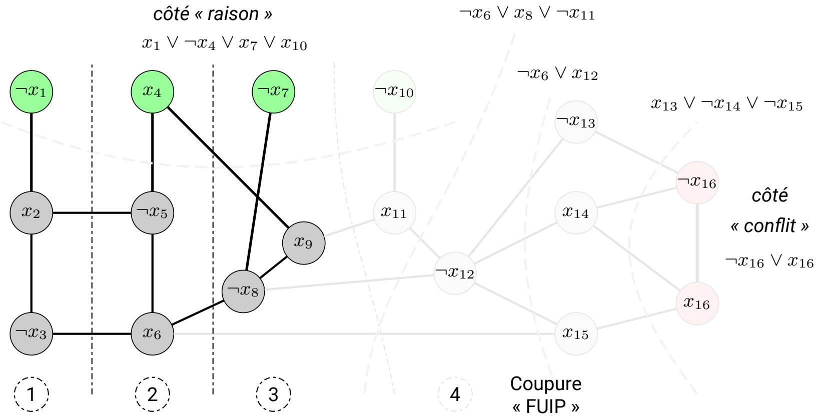


côté
« conflit »

$\neg x_{16} \vee x_{16}$

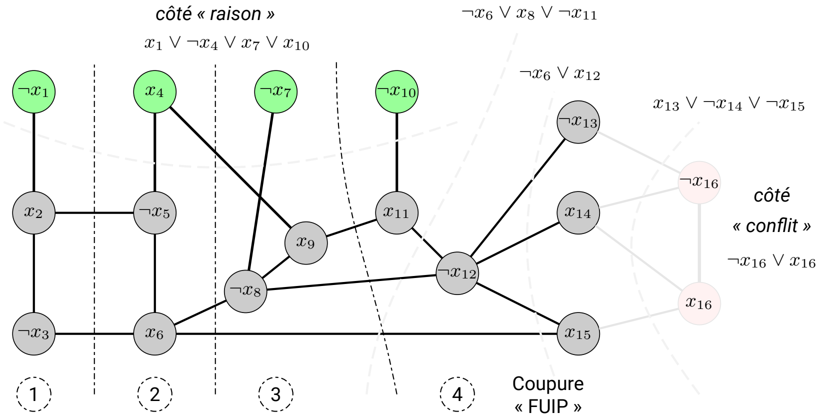
$x_1 \vee x_2$	$\neg x_4 \vee \neg x_5$	$x_7 \vee \neg x_6 \vee \neg x_8$	$x_{10} \vee \neg x_9 \vee x_{11}$	$\neg x_6 \vee x_{12} \vee x_{15}$
$\neg x_2 \vee \neg x_3$	$x_5 \vee x_6$	$\neg x_4 \vee x_8 \vee x_9$	$\neg x_{11} \vee x_8 \vee \neg x_{12}$	$x_{13} \vee \neg x_{14} \vee \neg x_{16}$
			$x_{12} \vee \neg x_{13}$	$\neg x_{15} \vee \neg x_{14} \vee x_{16}$
			$x_{12} \vee x_{14}$	

Le graphe d'implication



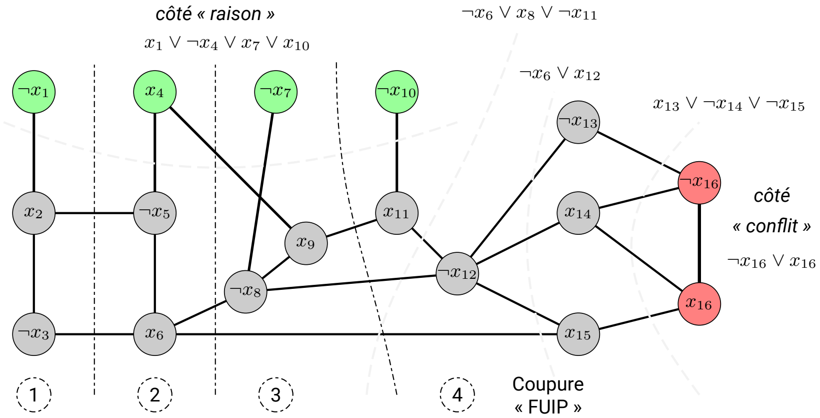
1	2	3	4	
$x_1 \lor x_2$	$\neg x_4 \lor \neg x_5$	$x_7 \lor \neg x_6 \lor \neg x_8$	$x_{10} \lor \neg x_9 \lor x_{11}$	$\neg x_6 \lor x_{12} \lor x_{15}$
$\neg x_2 \lor \neg x_3$	$x_5 \lor x_6$	$\neg x_4 \lor x_8 \lor x_9$	$\neg x_{11} \lor x_8 \lor \neg x_{12}$	$x_{13} \lor \neg x_{14} \lor \neg x_{16}$
			$x_{12} \lor \neg x_{13}$	$\neg x_{15} \lor \neg x_{14} \lor x_{16}$
			$x_{12} \lor x_{14}$	

Le graphe d'implication



$x_1 \vee x_2$	$\neg x_4 \vee \neg x_5$	$x_7 \vee \neg x_6 \vee \neg x_8$	$x_{10} \vee \neg x_9 \vee x_{11}$	$\neg x_6 \vee x_{12} \vee x_{15}$
$\neg x_2 \vee \neg x_3$	$x_5 \vee x_6$	$\neg x_4 \vee x_8 \vee x_9$	$\neg x_{11} \vee x_8 \vee \neg x_{12}$	$x_{13} \vee \neg x_{14} \vee \neg x_{16}$
			$x_{12} \vee \neg x_{13}$	$\neg x_{15} \vee \neg x_{14} \vee x_{16}$
			$x_{12} \vee x_{14}$	

Le graphe d'implication



$x_1 \vee x_2$

$\neg x_2 \vee \neg x_3$

$\neg x_4 \vee \neg x_5$

$x_5 \vee x_6$

$x_7 \vee \neg x_6 \vee \neg x_8$

$\neg x_4 \vee x_8 \vee x_9$

$x_{10} \vee \neg x_9 \vee x_{11}$

$\neg x_{11} \vee x_8 \vee \neg x_{12}$

$x_{12} \vee \neg x_{13}$

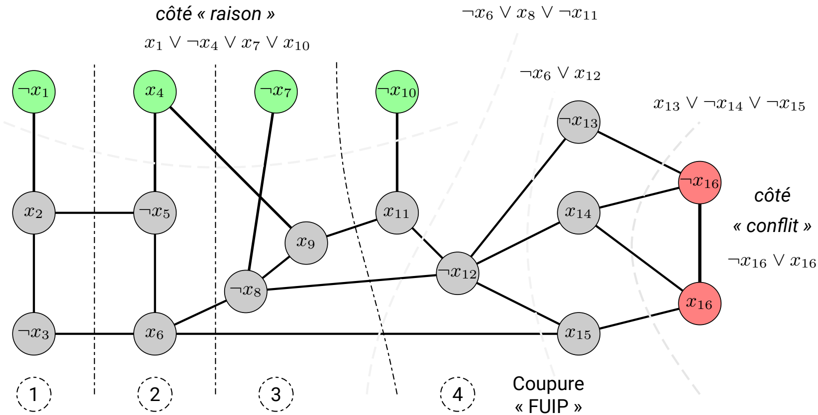
$x_{12} \vee x_{14}$

$\neg x_6 \vee x_{12} \vee x_{15}$

$x_{13} \vee \neg x_{14} \vee \neg x_{16}$

$\neg x_{15} \vee \neg x_{14} \vee x_{16}$

Le graphe d'implication



$x_1 \vee x_2$
$\neg x_2 \vee \neg x_3$

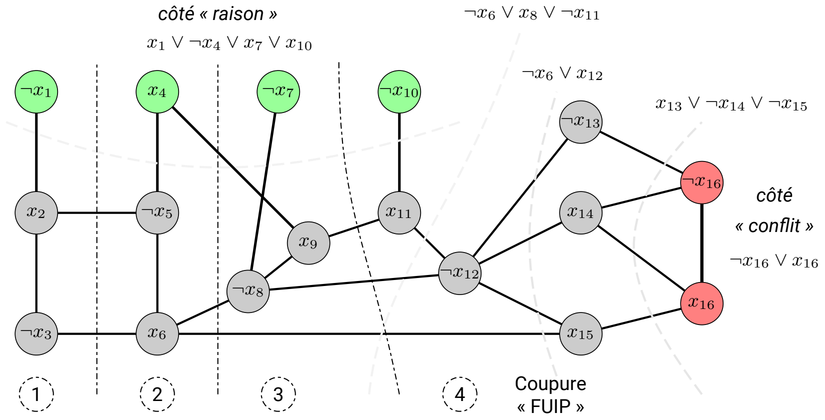
$\neg x_4 \vee \neg x_5$
$x_5 \vee x_6$

$x_7 \vee \neg x_6 \vee \neg x_8$
$\neg x_4 \vee x_8 \vee x_9$

$x_{10} \vee \neg x_9 \vee x_{11}$
$\neg x_{11} \vee x_8 \vee \neg x_{12}$
$x_{12} \vee \neg x_{13}$
$x_{12} \vee x_{14}$

$\neg x_6 \vee x_{12} \vee x_{15}$
$x_{13} \vee \neg x_{14} \vee \neg x_{16}$
$\neg x_{15} \vee \neg x_{14} \vee x_{16}$

Le graphe d'implication



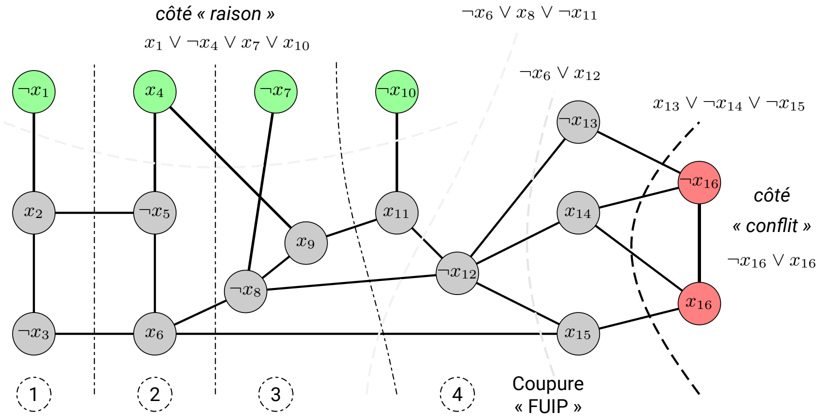
$x_1 \vee x_2$ $\neg x_4 \vee \neg x_5$ $x_7 \vee \neg x_6 \vee \neg x_8$ $x_{10} \vee \neg x_9 \vee x_{11}$ $\neg x_6 \vee x_{12} \vee x_{15}$

$\neg x_2 \vee \neg x_3$ $x_5 \vee x_6$ $\neg x_4 \vee x_8 \vee x_9$ $\neg x_{11} \vee x_8 \vee \neg x_{12}$ $x_{13} \vee \neg x_{14} \vee \neg x_{16}$

$x_{12} \vee \neg x_{13}$ $\neg x_{15} \vee \neg x_{14} \vee x_{16}$

$x_{12} \vee x_{14}$

Le graphe d'implication



$x_1 \vee x_2$

$\neg x_2 \vee \neg x_3$

$\neg x_4 \vee \neg x_5$

$x_5 \vee x_6$

$x_7 \vee \neg x_6 \vee \neg x_8$

$\neg x_4 \vee x_8 \vee x_9$

$x_{10} \vee \neg x_9 \vee x_{11}$

$\neg x_{11} \vee x_8 \vee \neg x_{12}$

$x_{12} \vee \neg x_{13}$

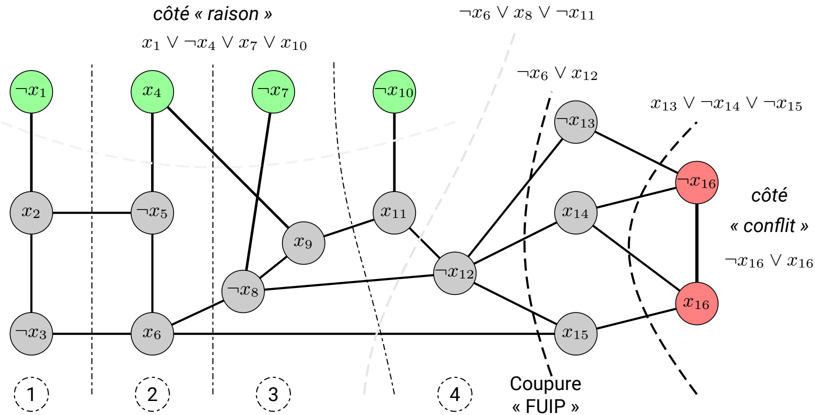
$x_{12} \vee x_{14}$

$\neg x_6 \vee x_{12} \vee x_{15}$

$x_{13} \vee \neg x_{14} \vee \neg x_{16}$

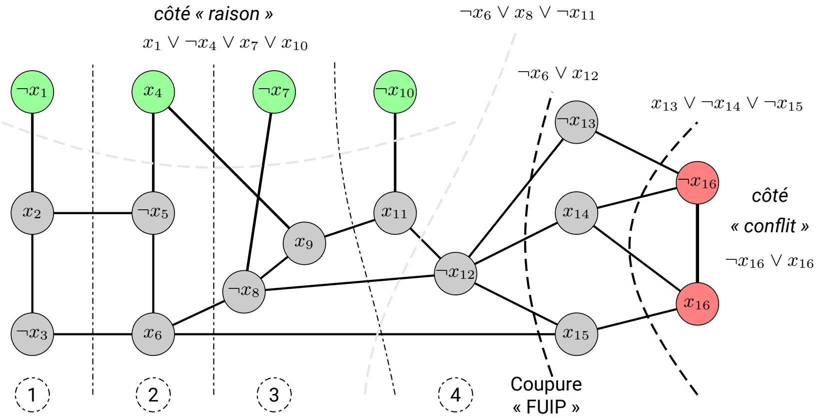
$\neg x_{15} \vee \neg x_{14} \vee x_{16}$

Le graphe d'implication



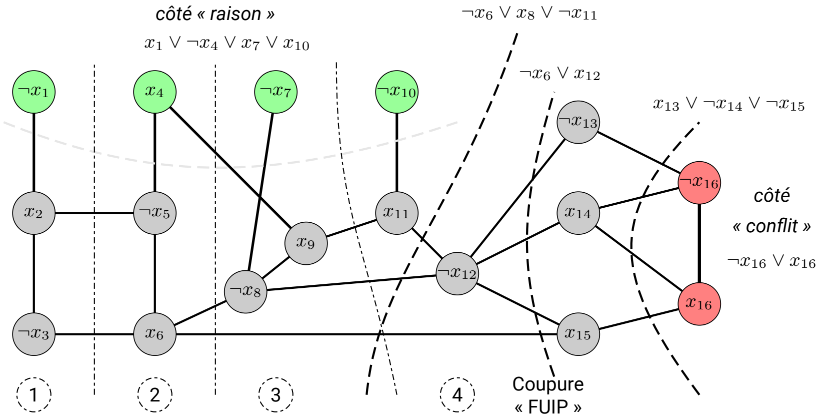
1	2	3	4	
$x_1 \vee x_2$	$\neg x_4 \vee \neg x_5$	$x_7 \vee \neg x_6 \vee \neg x_8$	$x_{10} \vee \neg x_9 \vee x_{11}$	$\neg x_6 \vee x_{12} \vee x_{15}$
$\neg x_2 \vee \neg x_3$	$x_5 \vee x_6$	$\neg x_4 \vee x_8 \vee x_9$	$\neg x_{11} \vee x_8 \vee \neg x_{12}$	$x_{13} \vee \neg x_{14} \vee \neg x_{16}$
			$x_{12} \vee \neg x_{13}$	$\neg x_{15} \vee \neg x_{14} \vee x_{16}$
			$x_{12} \vee x_{14}$	

Le graphe d'implication



1	2	3	4	
$x_1 \vee x_2$	$\neg x_4 \vee \neg x_5$	$x_7 \vee \neg x_6 \vee \neg x_8$	$x_{10} \vee \neg x_9 \vee x_{11}$	$\neg x_6 \vee x_{12} \vee x_{15}$
$\neg x_2 \vee \neg x_3$	$x_5 \vee x_6$	$\neg x_4 \vee x_8 \vee x_9$	$\neg x_{11} \vee x_8 \vee \neg x_{12}$	$x_{13} \vee \neg x_{14} \vee \neg x_{16}$
			$x_{12} \vee \neg x_{13}$	$\neg x_{15} \vee \neg x_{14} \vee x_{16}$
			$x_{12} \vee x_{14}$	

Le graphe d'implication



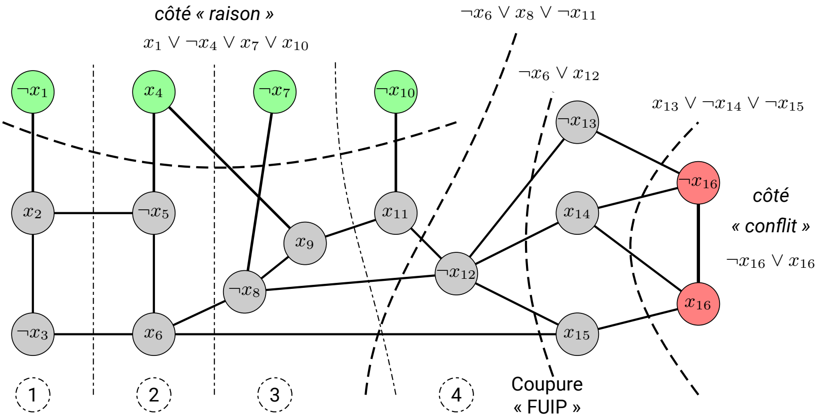
$x_1 \vee x_2$ $\neg x_4 \vee \neg x_5$ $x_7 \vee \neg x_6 \vee \neg x_8$ $x_{10} \vee \neg x_9 \vee x_{11}$ $\neg x_6 \vee x_{12} \vee x_{15}$

$\neg x_2 \vee \neg x_3$ $x_5 \vee x_6$ $\neg x_4 \vee x_8 \vee x_9$ $\neg x_{11} \vee x_8 \vee \neg x_{12}$ $x_{13} \vee \neg x_{14} \vee \neg x_{16}$

$x_{12} \vee \neg x_{13}$ $\neg x_{15} \vee \neg x_{14} \vee x_{16}$

$x_{12} \vee x_{14}$

Le graphe d'implication



$x_1 \vee x_2$ $\neg x_4 \vee \neg x_5$ $x_7 \vee \neg x_6 \vee \neg x_8$ $x_{10} \vee \neg x_9 \vee x_{11}$ $\neg x_6 \vee x_{12} \vee x_{15}$

$\neg x_2 \vee \neg x_3$ $x_5 \vee x_6$ $\neg x_4 \vee x_8 \vee x_9$ $\neg x_{11} \vee x_8 \vee \neg x_{12}$ $x_{13} \vee \neg x_{14} \vee \neg x_{16}$

$x_{12} \vee \neg x_{13}$ $\neg x_{15} \vee \neg x_{14} \vee x_{16}$

$x_{12} \vee x_{14}$

Un graphe d'implication... réel