Plan du cours

Première partie

- Qu'est-ce qu'un problème SAT ?
- Pourquoi chercher à résoudre SAT efficacement ?
- Pourquoi analyser ce problème ?

Deuxième partie

- Les algorithmes qui ont menés aux meilleurs algorithmes actuels
- Comment le résoudre efficacement aujourd'hui ?

Troisième partie

- Comment résoudre un problème avec SAT ?
- Dans quels domaines on en tire profit ?

Plan du cours

Première partie

- Qu'est-ce qu'un problème SAT ?
- Pourquoi chercher à résoudre SAT efficacement ?
- Pourquoi analyser ce problème ?

Deuxième partie

- Les algorithmes qui ont menés aux meilleurs algorithmes actuels
- Comment le résoudre efficacement aujourd'hui ?

Troisième partie

- Comment résoudre un problème avec SAT ?
- Dans quels domaines on en tire profit ?

PLAN DU COURS

PREMIÈRE PARTIE

- Qu'est-ce qu'un problème SAT ?
- Pourquoi chercher à résoudre SAT efficacement ?
- Pourquoi analyser ce problème ?

DEUXIÈME PARTIE

- Les algorithmes qui ont menés aux meilleurs algorithmes actuels
- Comment le résoudre efficacement aujourd'hui ?

TROISIÈME PARTIE

- Comment résoudre un problème avec SAT ?
- Dans quels domaines on en tire profit ?

ITINÉRAIRE

Itinéraire

LE PROBLÈME SAT

Gauvain Bourgne

$$
\begin{array}{lccccc}
 & \neg x_1 & \vee & \neg x_2 & \vee & x_3 \\
\wedge & & & & & \neg x_3 \\
\wedge & x_1 & \vee & x_2 & & \\
\wedge & & & x_2 & \vee & x_3
\end{array}
$$

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

QUE PEUT-ON DEMANDER ?

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

Le problème SAT

$$
\begin{array}{ccccccc}
 & \neg x_1 & \vee & \neg x_2 & \vee & x_3 \\
\wedge & & & & & \neg x_3 \\
\wedge & x_1 & \vee & x_2 & & \\
\wedge & & & x_2 & \vee & x_3
\end{array}
$$

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

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

Que peut-on demander ?

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

Le problème SAT

$$
\begin{array}{lccccc}
 & \neg x_1 & \vee & \neg x_2 & \vee & x_3 \\
\wedge & & & & & \neg x_3 \\
\wedge & x_1 & \vee & x_2 & & \\
\wedge & & & x_2 & \vee & x_3
\end{array}
$$

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

- 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) ;

Que peut-on demander ?

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

Le problème SAT

$$
\begin{array}{lccccc}
 & \neg x_1 & \vee & \neg x_2 & \vee & x_3 \\
\wedge & & & & & \neg x_3 \\
\wedge & x_1 & \vee & x_2 & & \\
\wedge & & & x_2 & \vee & x_3
\end{array}
$$

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

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

Que peut-on demander ?

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

Le problème SAT

$$\begin{array}{lccccc} & \neg x_1 & \vee & \neg x_2 & \vee & x_3 \\ \wedge & & & & & \neg x_3 \\ \wedge & x_1 & \vee & x_2 & & \\ \wedge & & & x_2 & \vee & x_3 \end{array}$$

x_1	x_2	x_3
$\bot$	$\top$	$\bot$

- 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) ;

Que peut-on demander ?

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

Le langage des formules propositionnelles se défini à partir d'un ensemble de variables propositionnelles $\mathcal{V}$, de deux constantes *vrai* (ou $\top$) et *faux* (ou $\perp$) et d'un ensemble de *connecteurs logiques* : $\neg$ (négation), $\vee$ (disjonction), $\wedge$ (conjonction), $\rightarrow$ (implication), $\leftrightarrow$ (équivalence).

Définition (Le langage des propositions $\mathcal{PL}$)

Étant donné un ensemble de variables propositionnelles $\mathcal{V} = \{x_1, \ldots, x_n\}$, le langage $\mathcal{LP}$ des ***formules propositionnelles*** *(ou* ***propositions****) est le plus petit langage construit inductivement en appliquant un nombre fini de fois les règles :*

- *vrai* $\in \mathcal{PL}$, *faux* $\in \mathcal{PL}$
- $\forall x \in \mathcal{V}$, $x \in \mathcal{PL}$
- $\forall f \in \mathcal{PL}$, $\neg f \in \mathcal{PL}$
- $\forall f, g \in \mathcal{PL}$, $f \vee g$, $f \wedge g$, $f \rightarrow g$, $f \leftrightarrow g$, *et* (f) *sont dans* $\mathcal{PL}$.

La sémantique classique de la logique propositionnelle repose sur la notion *d'interprétation*. Intuitivement, on se donne un ensemble $\mathbb{B} = \{\mathbf{T}, \mathbf{F}\}$, constitué de deux *valeurs de vérité*, et l'on associe chaque variable propositionnelle à une de ces valeurs.

Définition (Interprétation)

*Une **interprétation** I d'un ensemble de variables $V \subseteq \mathcal{V}$ est une application ayant pour domaine V et pour co-domaine $\mathbb{B} = \{\mathbf{T}, \mathbf{F}\}$.*

Lorsque $V \neq \mathcal{V}$, on dit que l'interprétation est *partielle*.
On parle d'interprétation *totale* lorsque $V = \mathcal{V}$.

Définition (Interprétation d'une formule)

Étant donnée une interprétation I d'un ensemble de variables V et une formule propositionnelle f telle que $Var(f) \subseteq V$, la valeur de l'interprétation de f dans I (notée $[\![f]\!]_I$) est définie récursivement par :

- $[\![vrai]\!]_I = \boldsymbol{T}$ et $[\![faux]\!]_I = \boldsymbol{F}$
- si $x \in \mathcal{V}$, $[\![x]\!]_I = I(x)$
- $\forall f, g \in \mathcal{PL}$, $[\![\neg f]\!]_I = \boldsymbol{T}$ si $[\![f]\!]_I = \boldsymbol{F}$ et $\boldsymbol{F}$ sinon
  $[\![f \wedge g]\!]_I = \boldsymbol{F}$ si $[\![f]\!]_I = \boldsymbol{F}$ ou $[\![g]\!]_I = \boldsymbol{F}$, et $\boldsymbol{T}$ sinon
  $[\![f \vee g]\!]_I = \boldsymbol{T}$ si $[\![f]\!]_I = \boldsymbol{T}$ ou $[\![g]\!]_I = \boldsymbol{T}$, et $\boldsymbol{F}$ sinon
  $[\![f \rightarrow g]\!]_I = \boldsymbol{F}$ si $[\![f]\!]_I = \boldsymbol{F}$ et $[\![g]\!]_I = \boldsymbol{T}$, et $\boldsymbol{T}$ sinon
  $[\![f \leftrightarrow g]\!]_I = \boldsymbol{T}$ si $[\![f]\!]_I = [\![g]\!]_I$, et $\boldsymbol{F}$ sinon
  $[\![(f)]\!] = [\![f]\!]$

Définition (Restriction d'une formule)

Soient f une formule et I une interprétation sur $V \subseteq \mathcal{V}$. La ***restriction de** f **à** I* *(notée $f|_I$) est la formule $f\sigma_I$ où σ_I est la substitution caractérisée par* $\sigma = \{x \mapsto \mathit{vrai} / x \in V \text{ et } I(x) = \mathbf{T}\} \cup \{x \mapsto \mathit{faux} / x \in V \text{ et } I(x) = \mathbf{F}\}$

On peut naturellement appliquer les simplifications usuelles pour éliminer toutes les occurrences de *vrai* et *faux* dans $f\sigma_I$. Notons que si $Var(f) \subseteq V$ on a nécessairement soit $f\sigma \equiv \mathit{vrai}$, soit $f\sigma \equiv \mathit{faux}$. Sinon, les variables apparaissant dans $f\sigma$ sont nécessairement non interprétées dans I.

$$((x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg y \vee t))|_{\{x,\neg t\}} = ((y \vee \neg z) \wedge \neg y)$$

La logique en deux mots

Logique propositionnelle

- $V = \{x_1, \ldots, x_n\}$ est l'ensemble des variables propositionnelles, l_i est un littéral, c-à-d une variable x_i ou sa négation $\neg x_i$.
- Une clause est une disjonction de littéraux $c_i = l_1 \vee l_2 ... \vee l_{n_i}$. Une clause unitaire ne contient qu'un seul littéral.
- Une formule Σ est sous forme normale conjonctive ($\mathcal{CNF}$) lorsque Σ s'écrit comme une conjonction de clauses $\Sigma = c_1 \wedge c_2 ... \wedge c_m$ (on peut aussi parler d'ensembles de clauses).
- Problème SAT : existe-t-il une interprétation permettant d'évaluer une formule sous $\mathcal{CNF}$ à vrai ?
- Note : Toute formule peut se réécrire sous $\mathcal{CNF}$.

La logique en deux mots

Logique propositionnelle

- $V = \{x_1, \ldots, x_n\}$ est l'ensemble des variables propositionnelles, l_i est un littéral, c-à-d une variable x_i ou sa négation $\neg x_i$.
- Une clause est une disjonction de littéraux $c_i = l_1 \vee l_2 ... \vee l_{n_i}$. Une clause unitaire ne contient qu'un seul littéral.
- Une formule Σ est sous forme normale conjonctive ($\mathcal{CNF}$) lorsque Σ s'écrit comme une conjonction de clauses $\Sigma = c_1 \wedge c_2 ... \wedge c_m$ (on peut aussi parler d'ensembles de clauses).
- Problème SAT : existe-t-il une interprétation permettant d'évaluer une formule sous $\mathcal{CNF}$ à vrai ?
- Note : Toute formule peut se réécrire sous $\mathcal{CNF}$.

Logique propositionnelle

- $V = \{x_1, \ldots, x_n\}$ est l'ensemble des variables propositionnelles, l_i est un littéral, c-à-d une variable x_i ou sa négation $\neg x_i$.
- Une clause est une disjonction de littéraux $c_i = l_1 \vee l_2 ... \vee l_{n_i}$. Une clause unitaire ne contient qu'un seul littéral.
- Une formule Σ est sous forme normale conjonctive ($\mathcal{CNF}$) lorsque Σ s'écrit comme une conjonction de clauses $\Sigma = c_1 \wedge c_2 ... \wedge c_m$ (on peut aussi parler d'ensembles de clauses).
- Problème SAT : existe-t-il une interprétation permettant d'évaluer une formule sous $\mathcal{CNF}$ à vrai ?
- Note : Toute formule peut se réécrire sous $\mathcal{CNF}$.

La logique en deux mots

Logique propositionnelle

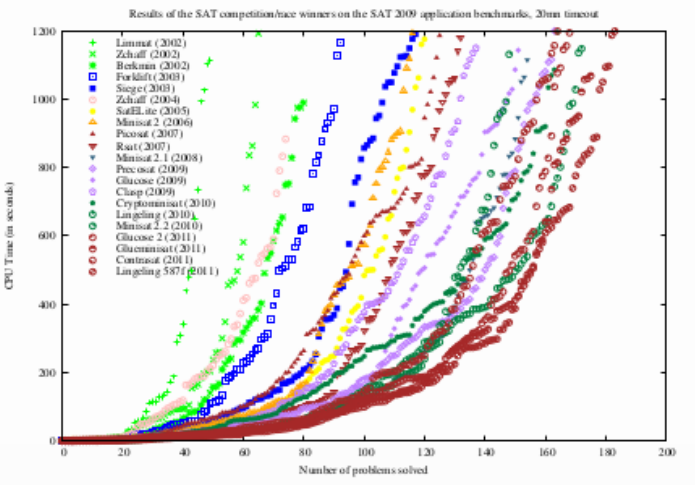
- $V = \{x_1, \ldots, x_n\}$ est l'ensemble des variables propositionnelles, l_i est un littéral, c-à-d une variable x_i ou sa négation $\neg x_i$.
- Une clause est une disjonction de littéraux $c_i = l_1 \vee l_2 ... \vee l_{n_i}$. Une clause unitaire ne contient qu'un seul littéral.
- Une formule Σ est sous forme normale conjonctive ($\mathcal{CNF}$) lorsque Σ s'écrit comme une conjonction de clauses $\Sigma = c_1 \wedge c_2 ... \wedge c_m$ (on peut aussi parler d'ensembles de clauses).
- Problème SAT : existe-t-il une interprétation permettant d'évaluer une formule sous $\mathcal{CNF}$ à vrai ?
- Note : Toute formule peut se réécrire sous $\mathcal{CNF}$.

LOGIQUE PROPOSITIONNELLE

- $V = \{x_1, \ldots, x_n\}$ est l'ensemble des variables propositionnelles, l_i est un littéral, c-à-d une variable x_i ou sa négation $\neg x_i$.
- Une clause est une disjonction de littéraux $c_i = l_1 \vee l_2 ... \vee l_{n_i}$. Une clause unitaire ne contient qu'un seul littéral.
- Une formule Σ est sous forme normale conjonctive ($\mathcal{CNF}$) lorsque Σ s'écrit comme une conjonction de clauses $\Sigma = c_1 \wedge c_2 ... \wedge c_m$ (on peut aussi parler d'ensembles de clauses).
- Problème SAT : existe-t-il une interprétation permettant d'évaluer une formule sous $\mathcal{CNF}$ à vrai ?
- Note : Toute formule peut se réécrire sous $\mathcal{CNF}$.

Progrès rapides

Résultats de la compétition SAT



SAT est le premier problème a avoir été demontré comme NP-Complet [Cook, 1971].

Problème NP canonique

POUR DEMONTRER QU'UN PROBLÈME EST NP

Il suffit de montrer qu'il existe une traduction polynomiale de ce problème en problème SAT.

SAT est le premier problème a avoir été demontré comme NP-Complet [Cook, 1971].

Problème NP canonique

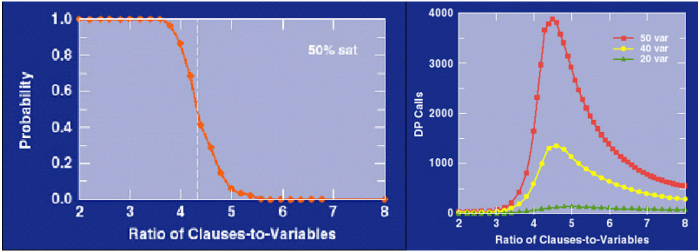
Pour demontrer qu'un problème est NP

Il suffit de montrer qu'il existe une traduction polynomiale de ce problème en problème SAT.

Complexité

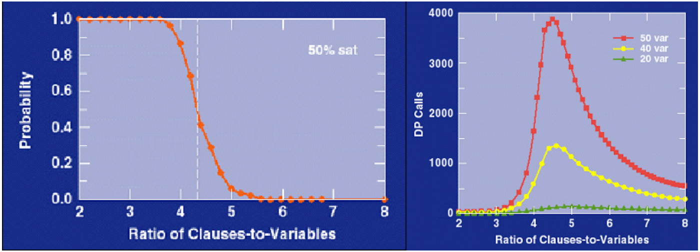
Phénomène de seuil

Un phénomène important observé sur les formule aléatoire 3-SAT.



Phénomène de transition de phase qu'on retrouve en physique des matériaux.

Un phénomène important observé sur les formule aléatoire 3-SAT.



Phénomène de transition de phase qu'on retrouve en physique des matériaux.

Itinéraire

Résoudre un problème SAT
Approches

- Méthodes locales (incomplet)
  - Recherche locale (GSAT, WalkSAT)
    - Bon résultats sur les instances SAT
    - ne peut prouver qu'une instance est UNSAT
- Méthodes systématiques
  - Résolution et varaintes (DP)
  - Méthode de Stalmarck
  - Apprentissage récursif
  - Recherche arborescente (DPLL)
  - Solveurs moderne : CDCL (Conflict Driven Clause Learning)

Davis et Putnam 1960 [DP60]
Raisonner sur les choix possibles

On a $f \equiv (x \wedge f_x) \vee (\neg x \wedge f_{\neg x})$ (décomposition de Shannon)

Notez que x a disparu dans f_x et $f_{\neg x}$

Idée de DP60
Éliminer les variables les unes après les autres.

DP(Σ)
Entrée : Σ une formule f sous FNC
Sortie : $\emptyset$ si un modèle existe, $\{\bot\}$ sinon
début
  Éliminer les clauses sous-sommées de Σ;
  si Σ *ne contient plus de variables* **alors** retourner Σ;
  ;
  Soit x une variable de Σ;
  retourner DP($\Sigma_x \vee \Sigma_{\neg x}$);
fin

Davis et Putnam 1960 [DP60]
Raisonner sur les choix possibles

On a $f \equiv (x \wedge f_x) \vee (\neg x \wedge f_{\neg x})$ (décomposition de Shannon)

Notez que x a disparu dans f_x et $f_{\neg x}$

Idée de DP60
Éliminer les variables les unes après les autres.

DP(Σ)
Entrée : Σ une formule *f* sous FNC
Sortie : ∅ si un modèle existe, {⊥} sinon
début
  Éliminer les clauses sous-sommées de Σ;
  si Σ *ne contient plus de variables* **alors** retourner Σ;
  ;
  Soit x une variable de Σ;
  retourner DP($\Sigma_x \vee \Sigma_{\neg x}$);
fin

DAVIS ET PUTNAM 1960 [DP60]
RAISONNER SUR LES CHOIX POSSIBLES

On a $f \equiv (x \wedge f_x) \vee (\neg x \wedge f_{\neg x})$ (décomposition de Shannon)

Notez que x a disparu dans f_x et $f_{\neg x}$

IDÉE DE DP60

Éliminer les variables les unes après les autres.

DP(Σ)
Entrée : Σ une formule f sous FNC
Sortie : $\emptyset$ si un modèle existe, $\{\perp\}$ sinon
début
  Éliminer les clauses sous-sommées de Σ;
  si Σ *ne contient plus de variables* **alors** retourner Σ;
  ;
  Soit x une variable de Σ;
  retourner DP($\Sigma_x \vee \Sigma_{\neg x}$);
fin

DAVIS ET PUTNAM 1960 [DP60]

RAISONNER SUR LES CHOIX POSSIBLES

DP(Σ)

Entrée : Σ une formule f sous FNC

Sortie : $\emptyset$ si un modèle existe, $\{\bot\}$ sinon

début

> Éliminer les clauses sous-sommées de Σ;
>
> **si** Σ *ne contient plus de variables* **alors** retourner Σ;
>
> ;
>
> Soit x une variable de Σ;
>
> retourner DP($\Sigma_x \vee \Sigma_{\neg x}$);

fin

Calcul de $\Sigma_x \vee \Sigma_{\neg x}$

RÈGLE DE RÉSOLUTION (COUPURE)

Soient $c_1 = (x \vee a_1 \vee \ldots a_n)$ et $c_2 = (\neg x \vee b_1 \vee \ldots b_m)$
$c = (a_1 \vee \ldots a_n \vee b_1 \vee \ldots b_m)$ est obtenu par résolution sur x entre c_1 et c_2.

Davis et Putnam 1960 [DP60]

Raisonner sur les choix possibles

DP(Σ)

Entrée : Σ une formule f sous FNC

Sortie : $\emptyset$ si un modèle existe, $\{\perp\}$ sinon

début

Éliminer les clauses sous-sommées de Σ;

si Σ *ne contient plus de variables* **alors** retourner Σ;

;

Soit x une variable de Σ;

retourner DP($\Sigma_x \vee \Sigma_{\neg x}$);

fin

Calcul de $\Sigma_x \vee \Sigma_{\neg x}$

Règle de Résolution (Coupure)

Soient $c_1 = (x \vee a_1 \vee \ldots a_n)$ et $c_2 = (\neg x \vee b_1 \vee \ldots b_m)$
$c = (a_1 \vee \ldots a_n \vee b_1 \vee \ldots b_m)$ est obtenu par résolution sur x entre c_1 et c_2.

Davis et Putnam 1960 [DP60]

Raisonner sur les choix possibles

DP(Σ)
Entrée : Σ une formule f sous FNC
Sortie : $\emptyset$ si un modèle existe, $\{\bot\}$ sinon
début
    Éliminer les clauses sous-sommées de Σ;
    si Σ *ne contient plus de variables* **alors** retourner Σ;
    ;
    Soit x une variable de Σ;
    retourner DP($\Sigma_x \vee \Sigma_{\neg x}$);
fin

Problèmes

- La mise sous FNC $\Sigma' = \Sigma_x \vee \Sigma_{\neg x}$ est quadratique
- Explosion combinatoire en mémoire
- *trop puissant pour SAT*

Davis et Putnam 1960 [DP60]

Trop de raisonnement

DP(Σ)
Données: Σ une formule f sous FNC
Résultat: **vrai** si un modèle existe, **faux** sinon
début
  Éliminer les clauses sous-sommées de Σ;
  si $\Sigma = \emptyset$ **alors** retourner vrai;
  si $\Sigma = \{\bot\}$ **alors** retourner faux;
  Soit x une variable de Σ;
  retourner DP($\Sigma_x \vee \Sigma_{\neg x}$);
fin

Solution : Changer le $\vee$ en une alternative de choix (OU).

DP(Σ)
Données: Σ une formule f sous FNC
Résultat: **vrai** si un modèle existe, **faux** sinon
début
  Éliminer les clauses sous-sommées de Σ;
  si $\Sigma = \emptyset$ **alors** retourner vrai;
  si $\Sigma = \{\perp\}$ **alors** retourner faux;
  Soit x une variable de Σ;
  retourner DP($\Sigma_x \vee \Sigma_{\neg x}$);
fin

Solution : Changer le $\vee$ en une alternative de choix (OU).

DP(Σ)
Données: Σ une formule f sous FNC
Résultat: **vrai** si un modèle existe, **faux** sinon
début

> Éliminer les clauses sous-sommées de Σ;
> **si** $\Sigma = \emptyset$ **alors** retourner vrai;
> **si** $\Sigma = \{\perp\}$ **alors** retourner faux;
> Soit x une variable de Σ;
> retourner **DP(Σ_x) OU DP($\Sigma_{\neg x}$)**;

fin

Solution : Changer le $\vee$ en une alternative de choix (OU).

Davis et Putnam 1960 [DP60]

Trop de raisonnement

DP(Σ)
Données: Σ une formule *f* sous FNC
Résultat: **vrai** si un modèle existe, **faux** sinon
début
  Éliminer les clauses sous-sommées de Σ;
  si $\Sigma = \emptyset$ **alors** retourner vrai;
  si $\Sigma = \{\bot\}$ **alors** retourner faux;
  Soit *l* un littéral de Σ;
  retourner **DP(Σ_l) OU DP($\Sigma_{\neg l}$)**;
fin

Solution : Changer le $\vee$ en une alternative de choix (OU).

DAVIS, LOGEMANN & LOVELAND [DPLL62]

ESSAYER LES CHOIX POSSIBLES, SYSTÉMATIQUEMENT

DPLL(Σ,(I))
Entrées : Σ (formule CNF), $\mathcal{I}$ (interprétation partielle)
Sortie : SAT / UNSAT
début

- **si** $\Sigma|\mathcal{I}$ *est vide* **alors** retourner SAT;
- **si** $\Sigma|\mathcal{I}$ *contient* $\perp$ **alors** retourner UNSAT;
- **si** $\Sigma|\mathcal{I}$ *contient un littéral pur* l **alors** retourner DPLL($\Sigma, \mathcal{I}.l$);
- **si** $\Sigma|\mathcal{I}$ *contient une clause unitaire* l **alors** retourner DPLL($\Sigma, \mathcal{I}.l$);
- Soit l un littéral de $\Sigma|\mathcal{I}$;
- **si** *DPLL($\Sigma, \mathcal{I}.l$) retourne* SAT **alors** retourner SAT;
- retourner DPLL($\Sigma, \mathcal{I}.\neg l$)

fin

Note : $\Sigma|\mathcal{I}$ est la formule réduite par $\mathcal{I}$

DAVIS, LOGEMANN & LOVELAND [DPLL62]
ESSAYER LES CHOIX POSSIBLES, SYSTÉMATIQUEMENT

DPLL(Σ,(I))
Entrées : Σ (formule CNF), $\mathcal{I}$ (interprétation partielle)
Sortie : SAT / UNSAT
début
- **si** $\Sigma|\mathcal{I}$ *est vide* **alors** retourner SAT;
- **si** $\Sigma|\mathcal{I}$ *contient* $\perp$ **alors** retourner UNSAT;
- **si** $\Sigma|\mathcal{I}$ *contient un littéral pur* l **alors** retourner DPLL($\Sigma, \mathcal{I}.l$);
- **si** $\Sigma|\mathcal{I}$ *contient une clause unitaire* l **alors** retourner DPLL($\Sigma, \mathcal{I}.l$);
- Soit l un littéral de $\Sigma|\mathcal{I}$;
- **si** *DPLL*($\Sigma, \mathcal{I}.l$) *retourne* SAT **alors** retourner SAT;
- retourner DPLL($\Sigma, \mathcal{I}.\neg l$)

fin

Note : $\Sigma|\mathcal{I}$ est la formule réduite par $\mathcal{I}$

DPLL(Σ,(I))
Entrées : Σ (formule CNF), $\mathcal{I}$ (interprétation partielle)
Sortie : SAT / UNSAT
début
- **si** $\Sigma|\mathcal{I}$ *est vide* **alors** retourner SAT;
- **si** $\Sigma|\mathcal{I}$ *contient* $\perp$ **alors** retourner UNSAT;
- **si** $\Sigma|\mathcal{I}$ *contient un littéral pur l* **alors** retourner DPLL($\Sigma, \mathcal{I}.l$);
- **si** $\Sigma|\mathcal{I}$ *contient une clause unitaire l* **alors** retourner DPLL($\Sigma, \mathcal{I}.l$);
- Soit l un littéral de $\Sigma|\mathcal{I}$;
- **si** *DPLL($\Sigma, \mathcal{I}.l$) retourne* SAT **alors** retourner SAT;
- retourner DPLL($\Sigma, \mathcal{I}.\neg l$)

fin

Note : $\Sigma|\mathcal{I}$ est la formule réduite par $\mathcal{I}$

DPLL est un algorithme standard de recherche avec retour arrière.

A chaque étape

- **[DECISION]** Choix d'une variable de décision
- **[DEDUCTION]** Application de la propagation unitaire (voire des littéraux purs)
- **[DIAGNOSTIC]** Test :
  - Si Conflit : BACKTRACK ou, si impossible (racine), renvoie UNSAT
  - Si Formule satisfaite (vide) : renvoie SAT
  - Sinon : refaire une DECISION;

DPLL62

UNE RECHERCHE ARBORESCENTE (BACKTRACK)

DPLL est un algorithme standard de recherche avec retour arrière.

A CHAQUE ÉTAPE

- **[DECISION]** Choix d'une variable de décision
- **[DEDUCTION]** Application de la propagation unitaire (voire des littéraux purs)
- **[DIAGNOSTIC]** Test :
  - Si Conflit : BACKTRACK ou, si impossible (racine), renvoie UNSAT
  - Si Formule satisfaite (vide) : renvoie SAT
  - Sinon : refaire une DECISION;

DPLL est un algorithme standard de recherche avec retour arrière.

A CHAQUE ÉTAPE

- **[DECISION]** Choix d'une variable de décision
- **[DEDUCTION]** Application de la propagation unitaire (voire des littéraux purs)
- **[DIAGNOSTIC]** Test :
  - Si Conflit : BACKTRACK ou, si impossible (racine), renvoie UNSAT
  - Si Formule satisfaite (vide) : renvoie SAT
  - Sinon : refaire une DECISION;

DPLL62

Une recherche arborescente (Backtrack)

DPLL est un algorithme standard de recherche avec retour arrière.

A chaque étape

- **[DECISION]** Choix d'une variable de décision
- **[DEDUCTION]** Application de la propagation unitaire (voire des littéraux purs)
- **[DIAGNOSTIC]** Test :
  - Si Conflit : BACKTRACK ou, si impossible (racine), renvoie UNSAT
  - Si Formule satisfaite (vide) : renvoie SAT
  - Sinon : refaire une DECISION;

DPLL62

Un exemple détaillé

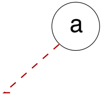
cf fichier `example DPLL.pdf`

An Example of DPLL

$$
\begin{aligned}
\varphi \quad = \quad & (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge \\
& (\neg b \vee \neg d \vee \neg e) \wedge \\
& (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge \\
& (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)
\end{aligned}
$$

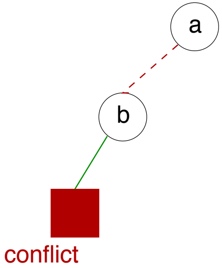
An Example of DPLL

$$
\begin{aligned}
\varphi \quad = \quad & (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge \\
& (\neg b \vee \neg d \vee \neg e) \wedge \\
& (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge \\
& (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)
\end{aligned}
$$



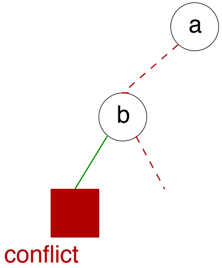
An Example of DPLL

$$\begin{aligned}\varphi \quad = \quad & (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge \\ & (\neg b \vee \neg d \vee \neg e) \wedge \\ & (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge \\ & (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)\end{aligned}$$



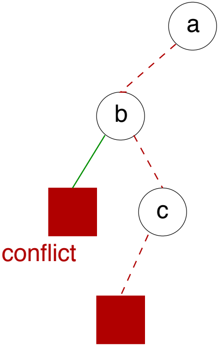
An Example of DPLL

$$
\begin{aligned}
\varphi \quad = \quad & (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge \\
& (\neg b \vee \neg d \vee \neg e) \wedge \\
& (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge \\
& (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)
\end{aligned}
$$



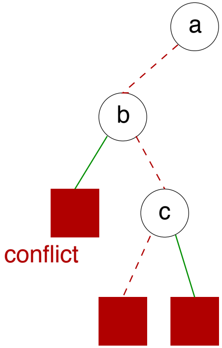
An Example of DPLL

$$\begin{aligned}\varphi \quad = \quad & (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge \\ & (\neg b \vee \neg d \vee \neg e) \wedge \\ & (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge \\ & (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)\end{aligned}$$



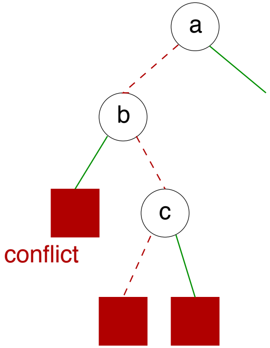
An Example of DPLL

$$
\begin{aligned}
\varphi \quad = \quad & (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge \\
& (\neg b \vee \neg d \vee \neg e) \wedge \\
& (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge \\
& (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)
\end{aligned}
$$



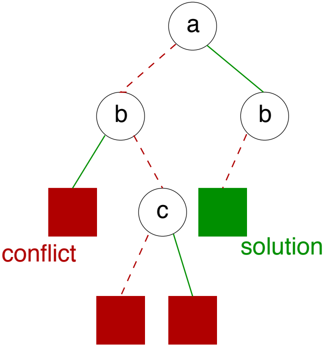
An Example of DPLL

$$\begin{aligned}\varphi \quad = \quad & (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge \\ & (\neg b \vee \neg d \vee \neg e) \wedge \\ & (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge \\ & (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)\end{aligned}$$



An Example of DPLL

$$\begin{aligned}\varphi \quad = \quad & (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge \\ & (\neg b \vee \neg d \vee \neg e) \wedge \\ & (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge \\ & (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)\end{aligned}$$



DPLL : diminuer la taille de l'arbre pour gagner

Constation

- Si le problème est **UNSAT** :
  Il faut parcourir les deux alternatives.
- Si le problème est **SAT** :
  On ne peut pas toujours choisir le bon littéral.

Dans tous les cas : il faut simplifier les deux sous-arbres.

Heuristique de choix du litéral

- Variable la plus fréquente dans les clauses courtes
- Variable générant le plus de propagations unitaires

Problème : il faut maintenir beaucoup de compteurs
Possible uniquement sur les problèmes *normaux*

DPLL : diminuer la taille de l'arbre pour gagner

Constatation

- Si le problème est **UNSAT** :
  Il faut parcourir les deux alternatives.
- Si le problème est **SAT** :
  On ne peut pas toujours choisir le bon littéral.

Dans tous les cas : il faut simplifier les deux sous-arbres.

Heuristique de choix du litéral

- Variable la plus fréquente dans les clauses courtes
- Variable générant le plus de propagations unitaires

Problème : il faut maintenir beaucoup de compteurs
Possible uniquement sur les problèmes *normaux*

DPLL : diminuer la taille de l'arbre pour gagner

Constation

- Si le problème est **UNSAT** :
  Il faut parcourir les deux alternatives.
- Si le problème est **SAT** :
  On ne peut pas toujours choisir le bon littéral.

Dans tous les cas : il faut simplifier les deux sous-arbres.

Heuristique de choix du litéral

- Variable la plus fréquente dans les clauses courtes
- Variable générant le plus de propagations unitaires

Problème : il faut maintenir beaucoup de compteurs
Possible uniquement sur les problèmes *normaux*

DPLL : diminuer la taille de l'arbre pour gagner

Constation

- Si le problème est **UNSAT** :
  Il faut parcourir les deux alternatives.
- Si le problème est **SAT** :
  On ne peut pas toujours choisir le bon littéral.

Dans tous les cas : il faut simplifier les deux sous-arbres.

Heuristique de choix du litéral

- Variable la plus fréquente dans les clauses courtes
- Variable générant le plus de propagations unitaires

Problème : il faut maintenir beaucoup de compteurs

Possible uniquement sur les problèmes *normaux*

DPLL : diminuer la taille de l'arbre pour gagner

Constation

- Si le problème est **UNSAT** :
  Il faut parcourir les deux alternatives.
- Si le problème est **SAT** :
  On ne peut pas toujours choisir le bon littéral.

Dans tous les cas : il faut simplifier les deux sous-arbres.

Heuristique de choix du litéral

- Variable la plus fréquente dans les clauses courtes
- Variable générant le plus de propagations unitaires

Problème : il faut maintenir beaucoup de compteurs
Possible uniquement sur les problèmes *normaux*

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

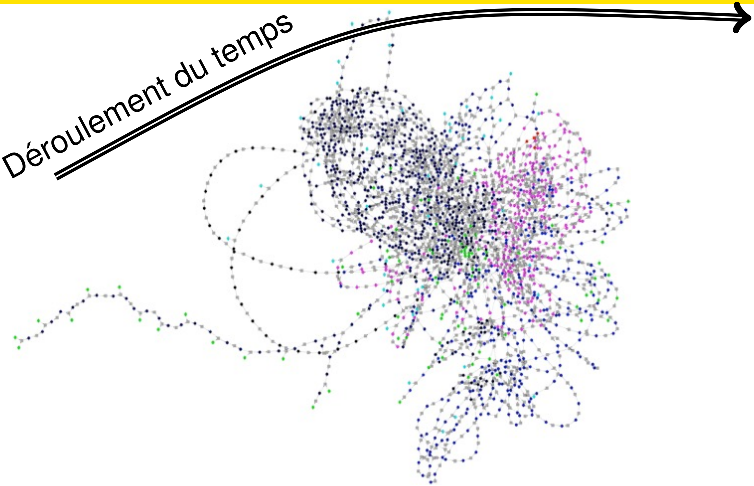
Années 2000, entrée de SAT dans l'ère industrielle

- 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

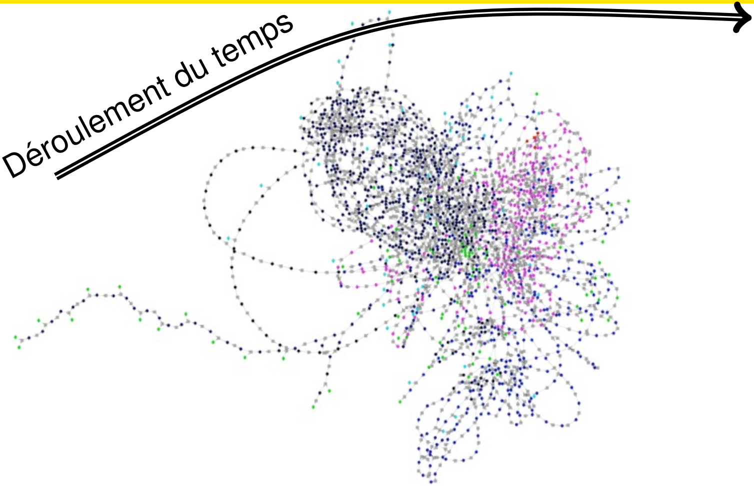
Il ne manque plus que des problèmes... industriels

Représentation d'un problème issu de IBM

Représentation d'un problème issu de IBM



Représentation d'un problème issu de IBM



Présence de backdoors

LES FORMULES GIGANTESQUES ARRIVENT

Solution pragmatique : les structures de données.

On ne détecte les propagations unitaires que lorsqu'elles se déclenchent.

CONSÉQUENCES

Les algorithmes fonctionnent à l'aveugle.

Toute l'architecture des solveurs CDCL est tournée vers l'apprentissage

CHAFF, LE CHANGEMENT DE PARADIGME

LES FORMULES GIGANTESQUES ARRIVENT

Solution pragmatique : les structures de données.
On ne détecte les propagations unitaires que lorsqu'elles se déclenchent.

CONSÉQUENCES

Les algorithmes fonctionnent à l'aveugle.

Toute l'architecture des solveurs CDCL est tournée vers l'apprentissage

Chaff, le changement de paradigme

Les formules gigantesques arrivent

Solution pragmatique : les structures de données.

On ne détecte les propagations unitaires que lorsqu'elles se déclenchent.

Conséquences

Les algorithmes fonctionnent à l'aveugle.

Toute l'architecture des solveurs CDCL est tournée vers l'apprentissage

Chaff, le changement de paradigme

Les formules gigantesques arrivent

Solution pragmatique : les structures de données.
On ne détecte les propagations unitaires que lorsqu'elles se déclenchent.

Conséquences

Les algorithmes fonctionnent à l'aveugle.

Toute l'architecture des solveurs CDCL est tournée vers l'apprentissage

PRINCIPALE AMÉLIORATION DES PERFORMANCES

GRASP a introduit dans le formalisme SAT l'idée d'apprentissage au niveau des conflits, lors de la recherche arborescente.

Cette idée, déjà bien connue en CSP, a révolutionné les démonstrateurs SAT de manière profonde.

Toute l'architecture des solveurs est tournée vers l'apprentissage

- Les heuristiques de choix
- Les retours arrières / la complétude
- Le redémarrage rapide

Principes généraux de l'apprentissage

Ne plus retourner au même endroit, pour les mêmes raisons
Lien entre l'apprentissage et la résolution.

$$\phi_1 = x_1 \vee x_4$$
$$\phi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\phi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\phi_4 = x_2 \vee x_{11}$$
$$\phi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\phi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\phi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

Schéma de fonctionnement des CDCL
Décisions – Propagations

DL 1 $\quad \overline{x_1} \longrightarrow \boxed{\overline{x_1}, x_4[\phi_1]}$

$$\phi_1 = x_1 \vee x_4$$
$$\phi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\phi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\phi_4 = x_2 \vee x_{11}$$
$$\phi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\phi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\phi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

Schéma de fonctionnement des CDCL

Décisions – Propagations

DL 1 $\quad \overline{x_1} \longrightarrow \boxed{\overline{x_1}, x_4[\phi_1]}$

$$
\begin{aligned}
\phi_1 &= x_1 \vee x_4 \\
\phi_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
\phi_3 &= x_1 \vee x_8 \vee x_{12} \\
\phi_4 &= x_2 \vee x_{11} \\
\phi_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
\phi_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
\phi_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
\end{aligned}
$$

Schéma de fonctionnement des CDCL

Décisions – Propagations

$$\phi_1 = x_1 \vee x_4$$
$$\phi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\phi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\phi_4 = x_2 \vee x_{11}$$
$$\phi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\phi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\phi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

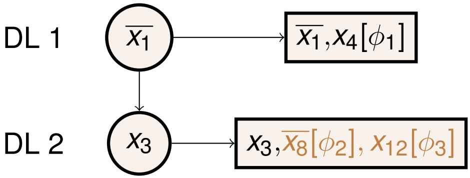


Schéma de fonctionnement des CDCL

Décisions – Propagations

$$\phi_1 = x_1 \vee x_4$$
$$\phi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\phi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\phi_4 = x_2 \vee x_{11}$$
$$\phi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\phi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\phi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

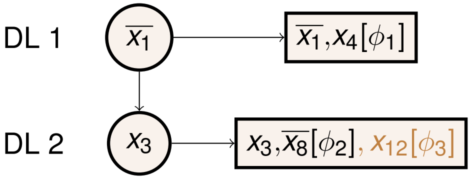


Schéma de fonctionnement des CDCL
Décisions – Propagations

$$
\begin{aligned}
\phi_1 &= x_1 \vee x_4 \\
\phi_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
\phi_3 &= x_1 \vee x_8 \vee x_{12} \\
\phi_4 &= x_2 \vee x_{11} \\
\phi_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
\phi_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
\phi_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
\end{aligned}
$$

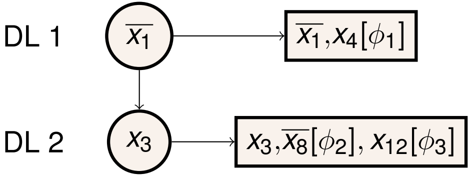


Schéma de fonctionnement des CDCL
Décisions – Propagations

$$\phi_1 = x_1 \vee x_4$$
$$\phi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\phi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\phi_4 = x_2 \vee x_{11}$$
$$\phi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\phi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\phi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

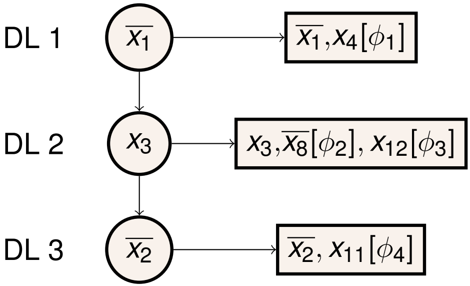
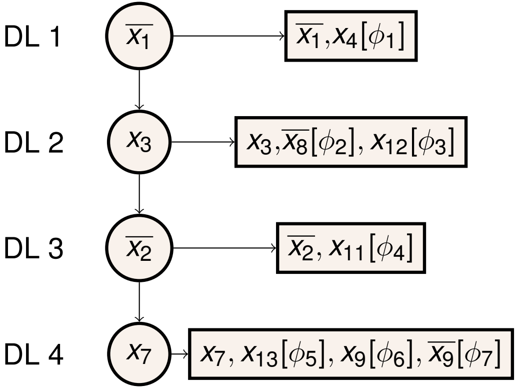


Schéma de fonctionnement des CDCL

Décisions – Propagations

$$\phi_1 = x_1 \vee x_4$$
$$\phi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\phi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\phi_4 = x_2 \vee x_{11}$$
$$\phi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\phi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\phi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$



DL 4 $\quad x_7 \longrightarrow \boxed{x_7, x_{13}[\phi_5], x_9[\phi_6], \overline{x_9}[\phi_7]}$

$$\phi_1 = x_1 \vee x_4$$
$$\phi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\phi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\phi_4 = x_2 \vee x_{11}$$
$$\phi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\phi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\phi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

Schéma de fonctionnement des CDCL
Analyse des conflits

$$\text{DL 4} \qquad x_7 \longrightarrow \boxed{x_7, x_{13}[\phi_5], x_9[\phi_6], \overline{x_9}[\phi_7]}$$

$$\beta_1 = res(x_9, \phi_7, \phi_6) = \overline{x_3} \vee x_8 \vee \overline{x_7} \vee \overline{x_{13}}$$

$$\phi_1 = x_1 \vee x_4$$
$$\phi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\phi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\phi_4 = x_2 \vee x_{11}$$
$$\phi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\phi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\phi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

DL 4 $\quad x_7 \longrightarrow \boxed{x_7, x_{13}[\phi_5], x_9[\phi_6], \overline{x_9}[\phi_7]}$

$$\beta_1 = res(x_9, \phi_7, \phi_6) = \overline{x_3} \vee x_8 \vee \overline{x_7} \vee \overline{x_{13}}$$
$$\beta = res(x_{13}, \beta_1, \phi_5) = \overline{x_3} \vee x_8 \vee \overline{x_7}$$

$$\phi_1 = x_1 \vee x_4$$
$$\phi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$\phi_3 = x_1 \vee x_8 \vee x_{12}$$
$$\phi_4 = x_2 \vee x_{11}$$
$$\phi_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$\phi_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$\phi_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

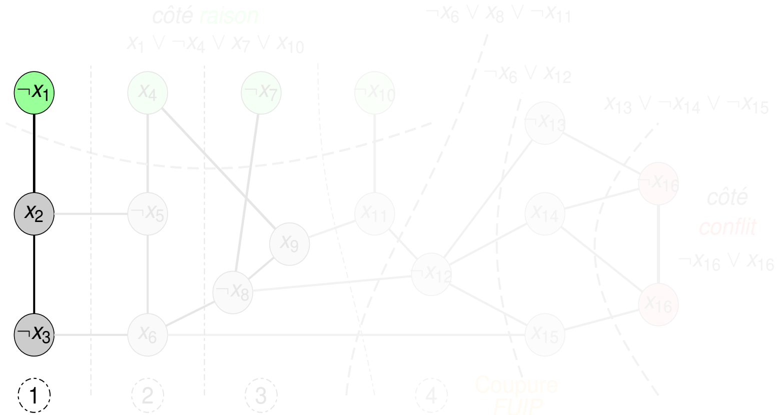
Schéma de fonctionnement des CDCL
Analyse des conflits

DL 4 $\quad x_7 \longrightarrow \boxed{x_7, x_{13}[\phi_5], x_9[\phi_6], \overline{x_9}[\phi_7]}$

$$\beta_1 = res(x_9, \phi_7, \phi_6) = \overline{x_3} \vee x_8 \vee \overline{x_7} \vee \overline{x_{13}}$$
$$\beta = res(x_{13}, \beta_1, \phi_5) = \overline{x_3} \vee x_8 \vee \overline{x_7}$$

- Stoppe dès que le résolvant contient un seul littéral du dernier niveau de décision (FUIP).
- β est ajouté à la base de clauses apprises (garantit le parcours systématique)

Détail sur le graphe des conflits
Le raisonnement sur l'échec



$x_1 \vee x_2$

$\neg x_2 \vee \neg x_3$

$\neg x_4 \vee \neg x_5$

$x_5 \vee x_6$

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

$\neg x_4 \vee x_8 \vee x_9$

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

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

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

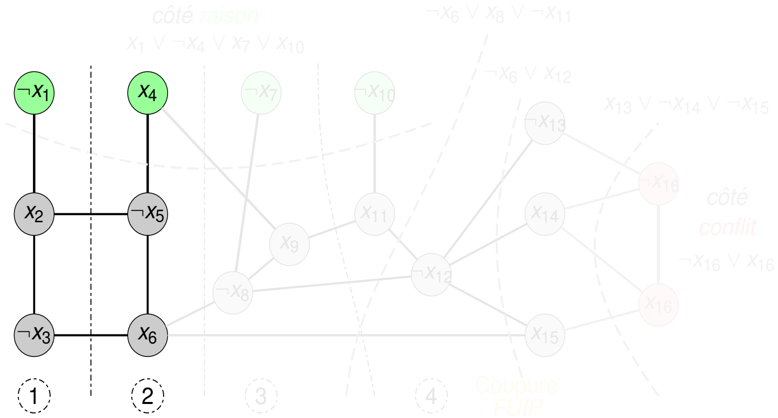
$x_{12} \vee x_{14}$

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

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

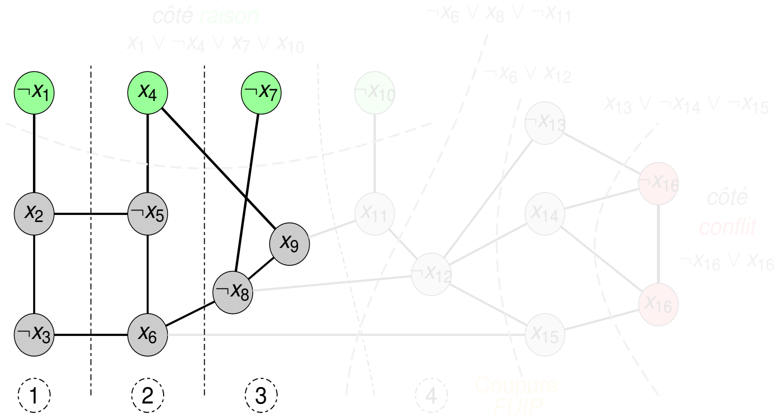
$\neg x_{14} \vee \neg x_{15} \vee x_{16}$

Détail sur le graphe des conflits
Le raisonnement sur l'échec



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

Détail sur le graphe des conflits
Le raisonnement sur l'échec



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

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

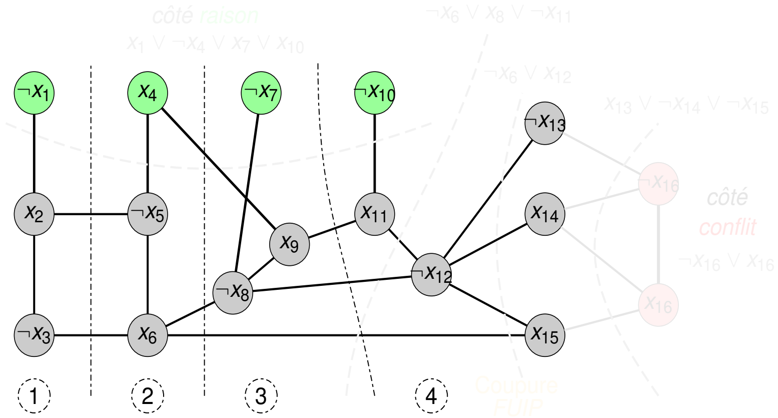
$x_7 \vee \neg x_6 \vee \neg x_8$
$\neg x_4 \vee x_8 \vee x_9$

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

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

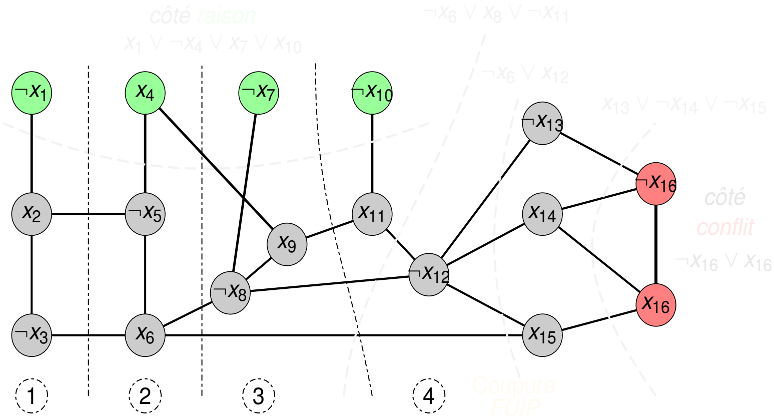
DÉTAIL SUR LE GRAPHE DES CONFLITS

LE RAISONNEMENT SUR L'ÉCHEC



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

DÉTAIL SUR LE GRAPHE DES CONFLITS
LE RAISONNEMENT SUR L'ÉCHEC



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

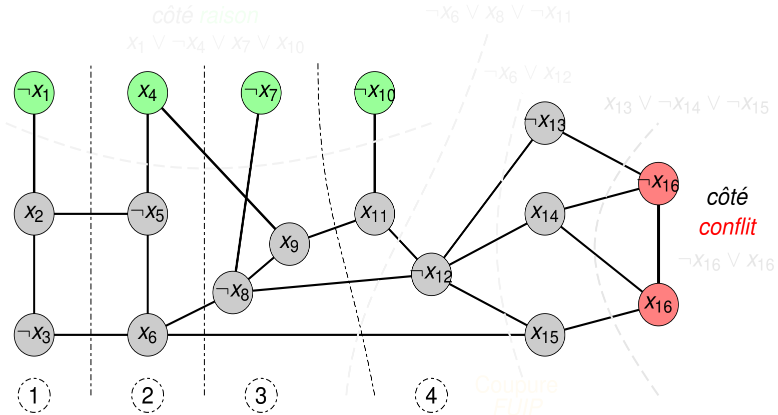
$\neg x_4 \vee \neg x_5$
$x_5 \vee x_6$

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

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

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

Détail sur le graphe des conflits
Le raisonnement sur l'échec



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

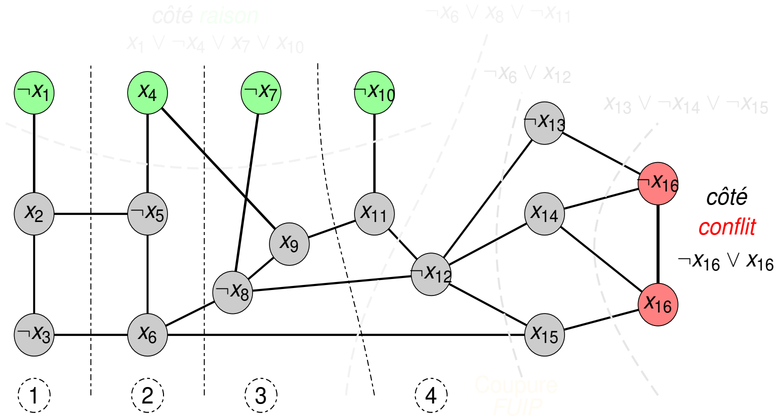
$\neg x_4 \vee \neg x_5$
$x_5 \vee x_6$

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

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

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

DÉTAIL SUR LE GRAPHE DES CONFLITS
LE RAISONNEMENT SUR L'ÉCHEC



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

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

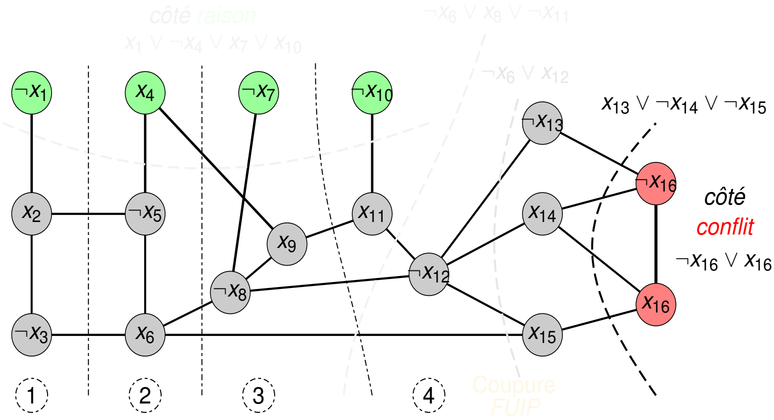
$x_7 \vee \neg x_6 \vee \neg x_8$
$\neg x_4 \vee x_8 \vee x_9$

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

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

Détail sur le graphe des conflits

Le raisonnement sur l'échec



$x_1 \vee x_2$

$\neg x_2 \vee \neg x_3$

$\neg x_4 \vee \neg x_5$

$x_5 \vee x_6$

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

$\neg x_4 \vee x_8 \vee x_9$

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

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

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

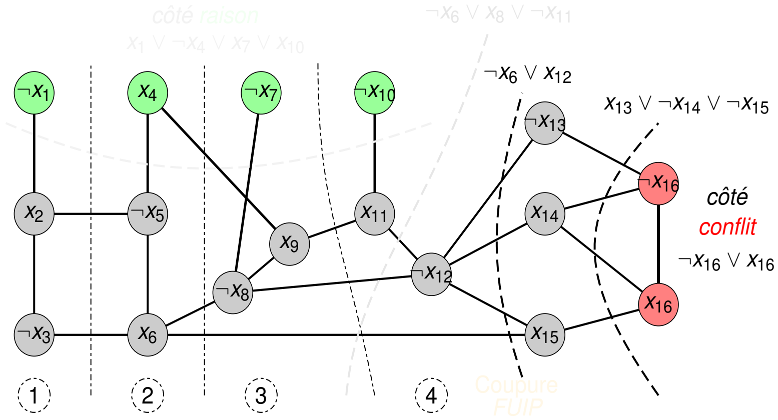
$x_{12} \vee x_{14}$

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

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

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

DÉTAIL SUR LE GRAPHE DES CONFLITS
LE RAISONNEMENT SUR L'ÉCHEC



$x_1 \vee x_2$

$\neg x_2 \vee \neg x_3$

$\neg x_4 \vee \neg x_5$

$x_5 \vee x_6$

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

$\neg x_4 \vee x_8 \vee x_9$

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

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

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

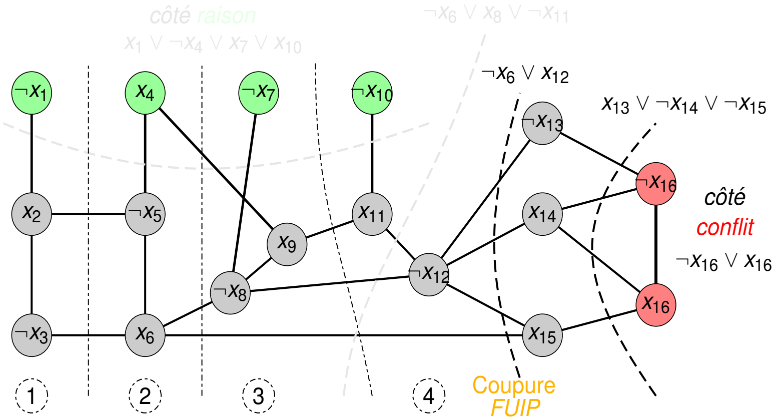
$x_{12} \vee x_{14}$

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

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

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

Détail sur le graphe des conflits
Le raisonnement sur l'échec



$x_1 \vee x_2$

$\neg x_2 \vee \neg x_3$

$\neg x_4 \vee \neg x_5$

$x_5 \vee x_6$

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

$\neg x_4 \vee x_8 \vee x_9$

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

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

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

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

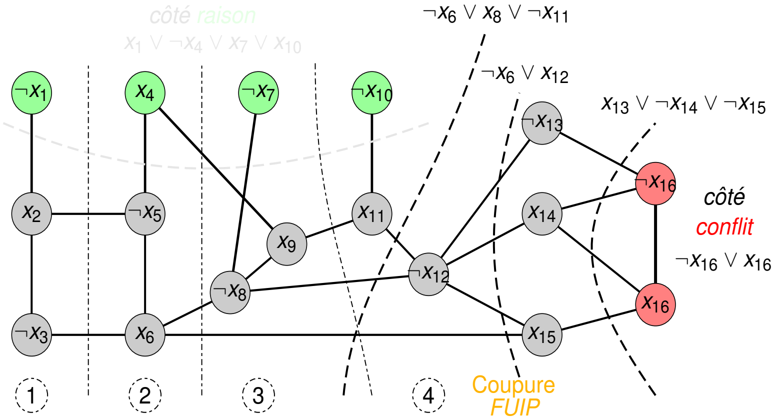
$\neg x_6 \vee x_{12} \vee x_{15}$

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

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

DÉTAIL SUR LE GRAPHE DES CONFLITS

LE RAISONNEMENT SUR L'ÉCHEC



$x_1 \vee x_2$

$\neg x_2 \vee \neg x_3$

$\neg x_4 \vee \neg x_5$

$x_5 \vee x_6$

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

$\neg x_4 \vee x_8 \vee x_9$

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

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

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

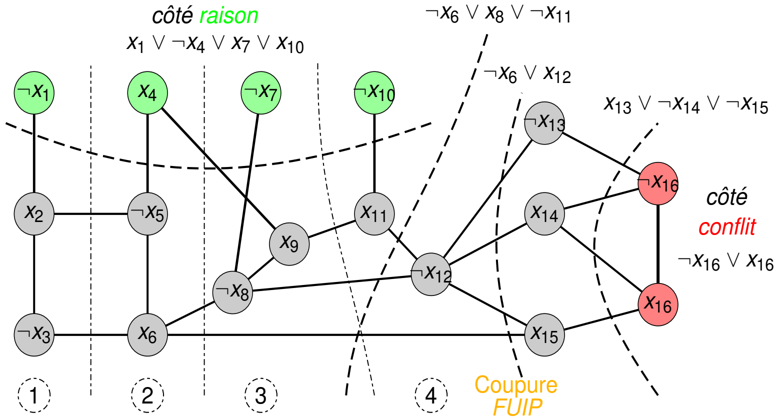
$x_{12} \vee x_{14}$

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

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

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

Détail sur le graphe des conflits
Le raisonnement sur l'échec



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

Schéma de fonctionnement des CDCL
Retour arrière non chronologique

$$
\begin{aligned}
\phi_1 &= x_1 \vee x_4 \\
\phi_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
\phi_3 &= x_1 \vee x_8 \vee x_{12} \\
\phi_4 &= x_2 \vee x_{11} \\
\phi_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
\phi_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
\phi_7 &= \overline{x_8} \vee \overline{x_7} \vee \overline{x_9}
\end{aligned}
$$

DL 1 : $\overline{x_1} \rightarrow \boxed{\overline{x_1}, x_4[\phi_1]}$

DL 2 : $x_3 \rightarrow \boxed{x_3, \overline{x_8}[\phi_2], x_{12}[\phi_3], \overline{x_7}[\beta]\ldots}$

DL 3 : $\overline{x_2} \rightarrow \boxed{\overline{x_2}, x_{11}[\phi_4]}$

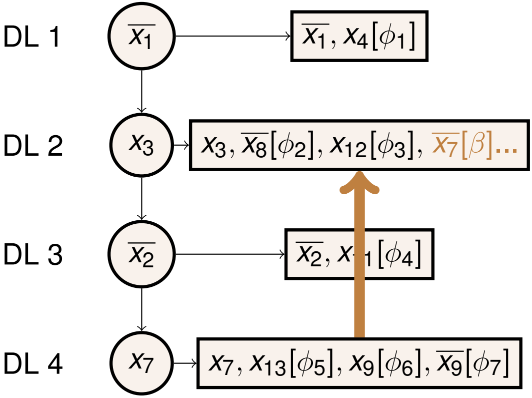
DL 4 : $x_7 \rightarrow \boxed{x_7, x_{13}[\phi_5], x_9[\phi_6], \overline{x_9}[\phi_7]}$

$$\beta = \overline{x_3} \vee x_8 \vee \overline{x_7}$$

Schéma de fonctionnement des CDCL

Retour arrière non chronologique

$$
\begin{aligned}
\phi_1 &= x_1 \vee x_4\\
\phi_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8}\\
\phi_3 &= x_1 \vee x_8 \vee x_{12}\\
\phi_4 &= x_2 \vee x_{11}\\
\phi_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13}\\
\phi_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9\\
\phi_7 &= \overline{x_8} \vee \overline{x_7} \vee \overline{x_9}
\end{aligned}
$$

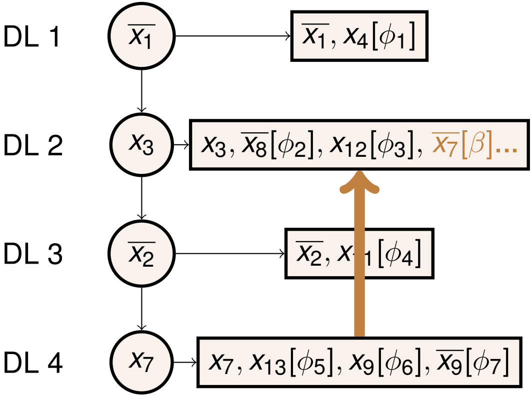


$$\beta = \overline{x_3} \vee x_8 \vee \overline{x_7}$$

Schéma de fonctionnement des CDCL

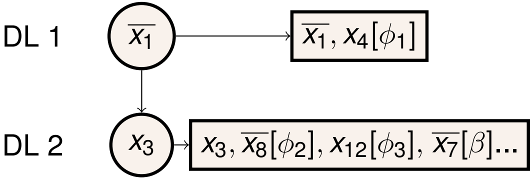
Retour arrière non chronologique

$$
\begin{aligned}
\phi_1 &= x_1 \vee x_4 \\
\phi_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
\phi_3 &= x_1 \vee x_8 \vee x_{12} \\
\phi_4 &= x_2 \vee x_{11} \\
\phi_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
\phi_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
\phi_7 &= \overline{x_8} \vee \overline{x_7} \vee \overline{x_9}
\end{aligned}
$$



$$\beta = \overline{x_3} \vee x_8 \vee \overline{x_7}$$

SCHÉMA DE FONCTIONNEMENT DES CDCL
RETOUR ARRIÈRE NON CHRONOLOGIQUE



$$\begin{aligned}
\phi_1 &= x_1 \vee x_4 \\
\phi_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
\phi_3 &= x_1 \vee x_8 \vee x_{12} \\
\phi_4 &= x_2 \vee x_{11} \\
\phi_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
\phi_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
\phi_7 &= \overline{x_8} \vee \overline{x_7} \vee \overline{x_9}
\end{aligned}$$

$$\beta = \overline{x_3} \vee x_8 \vee \overline{x_7}$$

Avant `zchaff [2001] :` *lookahead*

"si on veut savoir où aller, on doit savoir où on est"

- Équilibrer les branches
- Maintenir les compteurs heuristiques

Avant `zchaff [2001] :` *lookback*

"on sait ce qu'on a fait, mais pas où on est"

- Apprentissage, détection paresseuse des clauses unitaires
- Redémarrages (ultra) rapides
- Heuristique (ultra) dynamique (durée de vie : 1/10s)
- (Prétraitements efficaces)
- ...

LES DÉMONSTRATEURS SAT MODERNES

AVANT `zchaff [2001]` : *lookahead*

"si on veut savoir où aller, on doit savoir où on est"

- Équilibrer les branches
- Maintenir les compteurs heuristiques

AVANT `zchaff [2001]` : *lookback*

"on sait ce qu'on a fait, mais pas où on est"

- Apprentissage, détection paresseuse des clauses unitaires
- Redémarrages (ultra) rapides
- Heuristique (ultra) dynamique (durée de vie : 1/10s)
- (Prétraitements efficaces)
- ...

VOICI (ENFIN) LA VRAIE PROCÉDURE SAT

EN ANGLAIS DANS LE TEXTE

$\mathcal{I} = \emptyset$, *level* $= 0$;
while *True* **do**
  Execute UP on $(\Sigma, \mathcal{I})$;
  if *a Conflict was reached* **then**
    if *level* $= 0$ **then** Retourner UNSAT;
    ;
    C = the derived conflict clause;
    l = the sole literal of C set at the conflict level;
    $level = max\{level(x) : x \in C/\{l\}\}$;
    $\mathcal{I} = \mathcal{I}$ less all assignments made at level greater than *level*;
    $(\Sigma, \mathcal{I}) = (\Sigma \cup \{C\}, \mathcal{I}.l)$;
  end
  if $\mathcal{I}$ *is total* **then** Return SAT;
  ;
  Choose a decision literal l occuring in $\Sigma|\mathcal{I}$;
  $\mathcal{I} = \mathcal{I}.l$;
  Increment *level*
end

Algorithm 1: Conflict-Driven Clause Learning (CDCL)

Tout est dirigé par l'apprentissage

La seule phase de raisonnement est dans la phase d'analyse du conflit.
Le calcul du backjump est effectué aussi dans cette phase.
Le dernier point de choix est remplacé par le FUIP du dernier point de choix.

Oublier un peu
Mais pas trop

Problème des chaff-like

L'apprentissage induit une consommation mémoire trop importante. Souvent, les algorithmes rendent l'âme à cause d'elle.

Point important : Oublier les clauses inintéressantes.

Les clauses trop longues ? Les clauses les moins actives ? On perd la complétude !

Dans Minisat

L'effacement est aggressif. La moitié des clauses apprises sont régulièrement effacées, en prenant d'abord les clauses d'activité moindre.

Dans Glucose

L'effacement est encore plus aggressif, grâce à une mesure de qualité des clauses.

Oublier un peu
Mais pas trop

Problème des chaff-like

L'apprentissage induit une consommation mémoire trop importante. Souvent, les algorithmes rendent l'âme à cause d'elle.

Point important : Oublier les clauses inintéressantes. Les clauses trop longues ? Les clauses les moins actives ? On perd la complétude !

Dans Minisat

L'effacement est aggressif. La moitié des clauses apprises sont régulièrement effacées, en prenant d'abord les clauses d'activité moindre.

Dans Glucose

L'effacement est encore plus aggressif, grâce à une mesure de qualité des clauses.

Oublier un peu
Mais pas trop

Problème des chaff-like

L'apprentissage induit une consommation mémoire trop importante. Souvent, les algorithmes rendent l'âme à cause d'elle.

Point important : Oublier les clauses inintéressantes. Les clauses trop longues ? Les clauses les moins actives ? On perd la complétude !

Dans Minisat

L'effacement est aggressif. La moitié des clauses apprises sont régulièrement effacées, en prenant d'abord les clauses d'activité moindre.

Dans Glucose

L'effacement est encore plus aggressif, grâce à une mesure de qualité des clauses.

Soyez paresseux

Mais surveillez le travail à faire

Idée

Il ne faut surveiller que les clauses qui deviennent unitaire à cause d'une dernière affectation.

Concessions: On ne saura plus, pendant la recherche, combien de clauses sont

- Satisfaites
- Réduites (ni même si elles sont binaires)

Des problèmes pour les heuristiques classiques à venir !

Une heuristique redoutable : VSIDS
Variable State Independant Decaying Sum

Une heuristique indépendante de l'affectation courante des variables (enfin pas directement).

En un mot

À chaque fois qu'une variable est vue durant l'analyse de conflits, son score est augmenté.

Variantes (moins bonnes au final) :

- Seules les variables de la clause apprise sont incrémentées

Une heuristique redoutable : VSIDS

Variable State Independant Decaying Sum

Une heuristique indépendante de l'affectation courante des variables (enfin pas directement)**.**

En un mot

À chaque fois qu'une variable est vue durant l'analyse de conflits, son score est augmenté.

Variantes (moins bonnes au final) :

- Seules les variables de la clause apprise sont incrémentées

Se focaliser sur les conflits réçents

Il faut autoriser le solveur à avancer dans sa recherche, donc oublier les variables qui ont été un temps primordiales.

- Dans zchaff : régulièrement, tous les scores (entiers) sont divisés par une constante
- Dans minisat : après chaque conflit, le score ajouté à chaque visite est augmenté (multiplié par 1.05 environ). On manipule maintenant des flottants.

Utilisation d'une structure de tas (heap) pour gérer l'heuristique.

Se focaliser sur les conflits réçents

Il faut autoriser le solveur à avancer dans sa recherche, donc oublier les variables qui ont été un temps primordiales.

- Dans zchaff : régulièrement, tous les scores (entiers) sont divisés par une constante
- Dans minisat : après chaque conflit, le score ajouté à chaque visite est augmenté (multiplié par 1.05 environ). On manipule maintenant des flottants.

Utilisation d'une structure de tas (heap) pour gérer l'heuristique.

DERNIERS ÉLÉMENTS D'ANATOMIE

- Faire quelques affectations au hasard (jusqu'à 0.02% des choix peuvent être aléatoires).
- Dans minisat, l'heuristique ne concerne que les variables. Les littéraux ne sont pas pris en compte : on branche toujours sur faux, puis sur la dernière branche vue.

SAUVEGARDE DE PHASE

- Une ligne de code de changement dans minisat, beaucoup d'améliorations...
- L'idée est que le backjump peut défaire aussi des choix qui étaient bons.
- Par défaut, on refait les mêmes choix que la dernière fois (jusqu'à une certaine limite de mémoire).

Sauvegarde de phase

- Une ligne de code de changement dans minisat, beaucoup d'améliorations...
- L'idée est que le backjump peut défaire aussi des choix qui étaient bons.
- Par défaut, on refait les mêmes choix que la dernière fois (jusqu'à une certaine limite de mémoire).

SAUVEGARDE DE PHASE

- Une ligne de code de changement dans minisat, beaucoup d'améliorations...
- L'idée est que le backjump peut défaire aussi des choix qui étaient bons.
- Par défaut, on refait les mêmes choix que la dernière fois (jusqu'à une certaine limite de mémoire).

RESTARTS RAPIDES
(LUBY ET AUTRES)

DÉBUT DES ANNÉES 2000

En *randomisant* les lancements, on observe que les distributions d'effort de recherche ne sont pas normales... mais ont plutôt des *longues traines*.

Pour casser ce phénomène, il faut redémarrer et tout reprendre à zéro (mais ailleurs).

Restarts rapides
(Luby et autres)

Début des années 2000

En *randomisant* les lancements, on observe que les distributions d'effort de recherche ne sont pas normales... mais ont plutôt des *longues traines*.

Pour casser ce phénomène, il faut redémarrer et tout reprendre à zéro (mais ailleurs).

RESTARTS RAPIDES
(LUBY ET AUTRES)

DÉBUT DES ANNÉES 2000

En *randomisant* les lancements, on observe que les distributions d'effort de recherche ne sont pas normales... mais ont plutôt des *longues traines*.

Pour casser ce phénomène, il faut redémarrer et tout reprendre à zéro (mais ailleurs).

Itinéraire

Domaines d'applications
Où utilise-t-on les solveurs SAT?

- Méthode formelles (en particulier Model Checking) :
  Verification fonctionnelle de circuits intégrés ou de programmes (utilisations industrielles), génération de tests,...
- Intelligence Artificielle :
  Planification, Représentation des connaissances, jeux et puzzle (sudoku, n-reines)
- Bioinformatique :
  inférences d'haplotype, analyse de réseaux de régulation génétiques, ...
- Problèmes combinatoires :
  Scheduling, packing, coloration de graphe, voyageur de commerce.
- Et bien d'autres :
  Diagnostic, Configuration de produits, Sécurité, Topographie...

Domaines d'applications

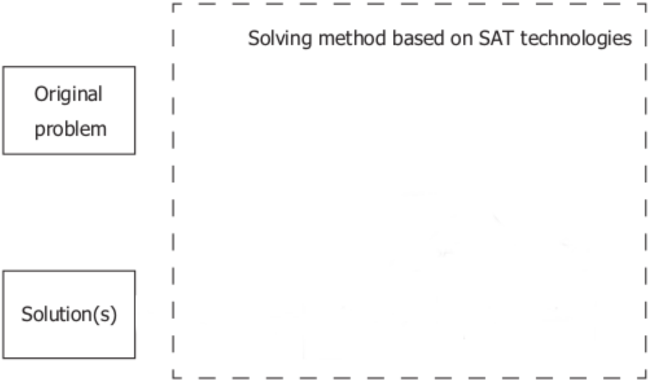
Où utilise-t-on les solveurs SAT?

- Brique de base d'autres solveurs: Pseudo-Booléens, QBF, ASP (CModels, CLASP), Max-SAT, MUS, SMT, Prouveur de théorème (HOL, Isabelle), CSP (Sugar)
- Utilisé dans des programmes répandus :
  - Suze 10.1 : gestionnaire de dépendances basé sur un solveur SAT
  - Eclipse : système d'allocation autmatique de ressources (provisioning) basé sur un solveur pseudo-booléen (extension de SAT)

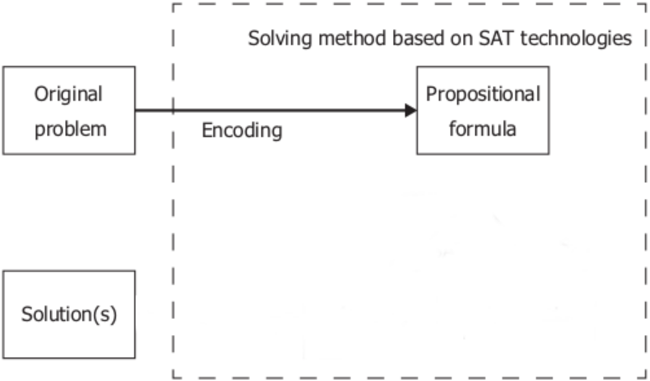
APPLICATIONS
COMMENT UTILISER SAT?

- Traduction directe : encoder un problème en SAT
- Méthode itérative : trouver une borne par de multiples problèmes SAT
- Enrichir l'expressivité : extensions de SAT

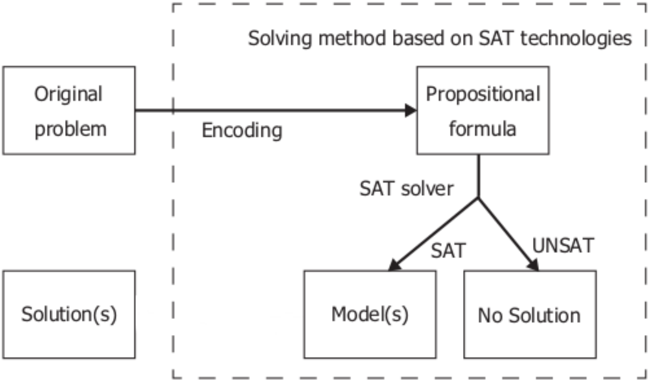
Résolution de problème avec SAT
Traduction directe



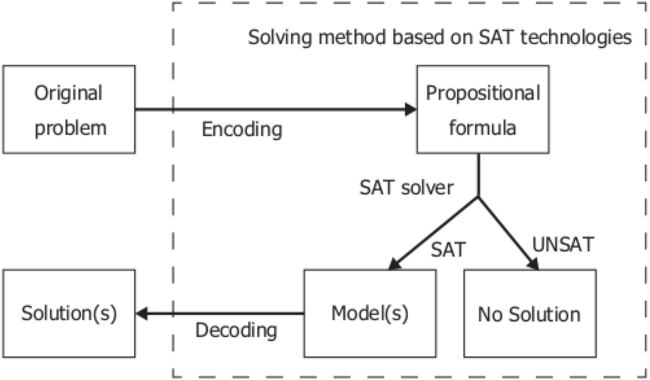
Résolution de problème avec SAT
Traduction directe

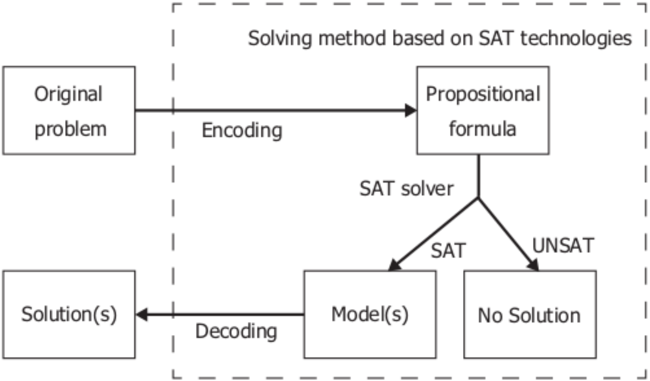


Résolution de problème avec SAT
Traduction directe



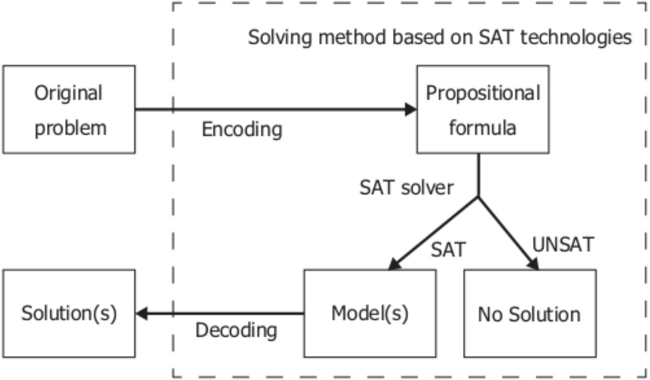
Résolution de problème avec SAT
Traduction directe





- Nécessité de traduire le problème en CNF : Encodage

  Encodage et décodage indépendant du solveur : échangeable facilement pour remplacer par un meilleur/plus adapté.



- Nécessité de traduire le problème en CNF : Encodage
- Encodage et décodage indépendant du solveur : échangeable facilement pour remplacer par un meilleur/plus adapté.

Problème :

On ne peut réussir sa thèse qu'avec un gros travail, ce qui fatigue. Quand on réussit sa thèse, on le fête. Qui fête sa réussite mange des petits fours et boit du champagne, qui est un alcool, ou du Champomy. Conduire quand on est fatigué et qu'on a bu de l'alcool est un moyen sûr d'avoir un accident. Ne pas conduire revient à dormir sur place.

Question : Comment éviter un accident après avoir réussi sa thèse?

Formalisation en logique propositionnelle

(1) ReussirThese $\rightarrow$ GrosTravail
(2) ReussirThese $\rightarrow$ Fete
(3) GrosTravail $\rightarrow$ Fatigue
(4) Fete $\rightarrow$ (BoireChampagne $\vee$ BoireChampomy) $\wedge$ MangerPetitsFours
(5) BoireChampagne $\rightarrow$ Alcool
(6) Fatigue $\wedge$ Alcool $\wedge$ Conduire $\rightarrow$ Accident
(7) $\neg$ Conduire $\leftrightarrow$ DormirSurPlace
(Q) ReussirThese $\wedge\neg$ Accident

Mise sous forme CNF

(1) (¬ ReussirThese ∨ GrosTravail) ∧
(2) (¬ ReussirThese ∨ Fete) ∧
(3) (¬ GrosTravail ∨ Fatigue) ∧
(4) (¬ Fete ∨ BoireChampagne ∨ BoireChampomy) ∧
(¬ Fete ∨ MangerPetitsFours) ∧
(5) (¬ BoireChampagne ∨ Alcool) ∧
(6) (¬ Fatigue ∨¬ Alcool ∨¬ Conduire ∨ Accident) ∧
(7) (Conduire ∨ DormirSurPlace) ∧ (¬ Conduire ∨ ¬ DormirSurPlace) ∧
(Q) (ReussirThese) ∧ (¬ Accident)

Modèles satisfaisant la formule

On note $PU = \{$ReussirThese, GrosTravail, FeterReussite, Fatigue, MangerPetitsFours$\}$

$M_1 = PU \cup \{$BoireChampomy, DormirSurPlace$\}$

. (Faux: Accident, BoireChampagne, Alcool, Conduire)

$M_2 = PU \cup \{$BoireChampomy, Conduire$\}$

. (Faux: Accident, BoireChampagne, Alcool, DormirSurPlace)

$M_3 = PU \cup \{$BoireChampomy, Alcool, DormirSurPlace$\}$

. (Faux: Accident, BoireChampagne, Conduire)

$M_4 = PU \cup \{$BoireChampagne, Alcool, DormirSurPlace$\}$

. (Faux: Accident, BoireChampomy, Conduire)

$M_5 = PU \cup \{$BoireChampagne, BoireChampomy, Alcool, DormirSurPlace$\}$

. (Faux: Accident, Conduire)

Solutions du problème:

1. Dormir sur place, ne pas conduire et boire ce qu'on veut : du champomy uniquement (M_1), du Champagne uniquement (M_4), ou des deux (M_5).
2. Conduire, ne pas dormir sur place et boire uniquement du Champomy (M_2).

Note : M_3 peut paraître étrange (on boit du Champomy, mais on est alcoolisé), il manque une règle pour préciser que le seul alcool est du Champagne.

Encodage

Il s'agit de traduire un problème en une formule propositionnelle (et de la mettre sous forme de clauses).

- Choix de variables propositionnelles
- Ecriture des règles et contraintes à prendre en compte
  Typiquement : "Si on a A et B, on aura C" s'écrit $\neg A \vee \neg B \vee C$

EXEMPLES

- *Vous devez choisir au moins une de ces UEs (une des 7 UEs du S2).*

- *Chacun (des 5 élèves) peut prendre au plus un dessert (parmi les 4 possibles).*

- *Un unique candidat doit être élu (parmi Pierre, Paul, Jacques).*

EXEMPLES

- *Vous devez choisir au moins une de ces UEs (une des 7 UEs du S2).*
  Variables prop. : $UE = \{\text{afr, bdr, bi, fpr, pldac, iamsi, tal}\}$ (afr signifiant "vous avez choisi l'UE AFR", etc.)

- *Chacun (des 5 élèves) peut prendre au plus un dessert (parmi les 4 possibles).*
  Variables prop. : $p_{e,d}$ signifie que l'élève e prend le dessert d

- *Un unique candidat doit être élu (parmi Pierre, Paul, Jacques).*
  Variables prop. : elu_p signifie que le candidat $p \in C = \{\text{Pierre, Paul, Jacques}\}$ est élu.

EXEMPLES

- $\Sigma_{u \in UE} u \geq 1$
  Variables prop. : $UE = \{$afr, bdr, bi, fpr, pldac, iamsi, tal$\}$ (afr signifiant "vous avez choisi l'UE AFR", etc.)

- $\forall e \in \{1, \ldots, 5\}, \Sigma_{d \in \{1,\ldots,4\}} p_{e,d} \leq 1$
  Variables prop. : $p_{e,d}$ signifie que l'élève e prend le dessert d

- $\Sigma_{p \in C} elu_p = 1$
  Variables prop. : elu_p signifie que le candidat $p \in C = \{$Pierre, Paul, Jacques$\}$ est élu.

- "Exactement 1" : $\Sigma_{i \in \{1,\ldots,n\}} x_i = 1$
- "Au moins 1" : $\Sigma_{i \in \{1,\ldots,n\}} x_i \geq 1$
- "Au plus 1" : $\Sigma_{i \in \{1,\ldots,n\}} x_i \leq 1$

- ”Exactement 1” : $\Sigma_{i\in\{1,\ldots,n\}}x_i = 1$

Encodage : $(\Sigma_{i\in\{1,\ldots,n\}}x_i \geq 1) \wedge (\Sigma_{i\in\{1,\ldots,n\}}x_i \leq 1)$

- ”Au moins 1” : $\Sigma_{i\in\{1,\ldots,n\}}x_i \geq 1$

- ”Au plus 1” : $\Sigma_{i\in\{1,\ldots,n\}}x_i \leq 1$

- "Exactement 1" : $\Sigma_{i\in\{1,\ldots,n\}} x_i = 1$

Encodage : $(\Sigma_{i\in\{1,\ldots,n\}} x_i \geq 1) \wedge (\Sigma_{i\in\{1,\ldots,n\}} x_i \leq 1)$

- "Au moins 1" : $\Sigma_{i\in\{1,\ldots,n\}} x_i \geq 1$

Encodage : $(x_1 \vee x_2 \vee \ldots \vee x_n)$

- "Au plus 1" : $\Sigma_{i\in\{1,\ldots,n\}} x_i \leq 1$

Encodage
Contraintes de cardinalité

- "Exactement 1" : $\Sigma_{i\in\{1,\ldots,n\}} x_i = 1$

Encodage : $(\Sigma_{i\in\{1,\ldots,n\}} x_i \geq 1) \wedge (\Sigma_{i\in\{1,\ldots,n\}} x_i \leq 1)$

- "Au moins 1" : $\Sigma_{i\in\{1,\ldots,n\}} x_i \geq 1$

Encodage : $(x_1 \vee x_2 \vee \ldots \vee x_n)$

- "Au plus 1" : $\Sigma_{i\in\{1,\ldots,n\}} x_i \leq 1$

Encodage :

- Par paires ($O(n^2)$ clauses) : $\bigwedge_{1\leq i<n}^{i<j\leq n} (\neg x_i \vee \neg x_j)$
- Bitwise ($O(nlogn)$ clauses, $O(n)$ variables supplémentaires)

Encodage

Contraintes de cardinalité - Bitwise encoding

Contrainte "Au plus 1" : $\Sigma_{i \in \{1,\ldots,n\}} x_i \leq 1$

- Variables auxiliaires : $v_0, \ldots, v_{r-1}$ où $r = log_2\ n$[1].
  $v_0 \ldots v_n$ doit coder k tel que x_{k+1} vraie (s'il y en a)
- De ce fait, si $x_j = 1$ on doit avoir chaque $v_i = b_i^j$ où $b_0^j \ldots b_r^j$ est le codage binaire de $j - 1$.
- Pour chaque $j \in \{1, \ldots, n\}$, pour chaque $i \in \{0, \ldots, r-1\}$, on ajoute la clause $C_{i,j} : (\neg x_j \vee l_i^j)$ où $l_i^j \equiv v_i$ si le ième bit b_i^j du codage binaire de $(j-1)$ vaut 1 et $l_i^j \equiv \neg v_i$ sinon.
- Si x_j est vrai :
  - Tous les v_i doivent correspondre à l'encodage de (j-1) (pour satisfaire $C_{i,j}$)
  - Tous les autres x_k doivent être faux (pour ne pas contredire au moins un des $C_{i,k}$)
- Si tous les x_j sont faux :
  - Tous les $C_{i,j}$ sont satisfaits, et n'importe quel assignement des v_i est cohérent.

---

[1] r : nombre de bit nécessaire à coder n en binaire

Contrainte "Au plus 1" : $\Sigma_{i \in \{1,\dots,n\}} x_i \leq 1$

- Variables auxiliaires : $v_0, \dots, v_{r-1}$ où $r = log_2\ n$[1].
  $v_0 \dots v_n$ doit coder k tel que x_{k+1} vraie (s'il y en a)
- De ce fait, si $x_j = 1$ on doit avoir chaque $v_i = b_i^j$ où $b_0^j \dots b_r^j$ est le codage binaire de $j - 1$.
- Pour chaque $j \in \{1, \dots, n\}$, pour chaque $i \in \{0, \dots, r-1\}$, on ajoute la clause $C_{i,j} : (\neg x_j \vee l_i^j)$ où $l_i^j \equiv v_i$ si le ième bit b_i^j du codage binaire de $(j-1)$ vaut 1 et $l_i^j \equiv \neg v_i$ sinon.
- Si x_j est vrai :
  - Tous les v_i doivent correspondre à l'encodage de (j-1) (pour satisfaire $C_{i,j}$)
  - Tous les autres x_k doivent être faux (pour ne pas contredire au moins un des $C_{i,k}$)
- Si tous les x_j sont faux :
  - Tous les $C_{i,j}$ sont satisfaits, et n'importe quel assignement des v_i est cohérent.

[1] r : nombre de bit nécessaire à coder n en binaire

Encodage

Contraintes de cardinalité - Bitwise encoding

Contrainte "Au plus 1" : $\Sigma_{i\in\{1,\ldots,n\}} x_i \leq 1$

- Variables auxiliaires : $v_0, \ldots, v_{r-1}$ où $r = log_2\ n$[1].
  $v_0 \ldots v_n$ doit coder k tel que x_{k+1} vraie (s'il y en a)
- De ce fait, si $x_j = 1$ on doit avoir chaque $v_i = b_i^j$ où $b_0^j \ldots b_r^j$ est le codage binaire de $j-1$.
- Pour chaque $j \in \{1, \ldots, n\}$, pour chaque $i \in \{0, \ldots, r-1\}$, on ajoute la clause $C_{i,j} : (\neg x_j \vee l_i^j)$ où $l_i^j \equiv v_i$ si le ième bit b_i^j du codage binaire de $(j-1)$ vaut 1 et $l_i^j \equiv \neg v_i$ sinon.
- Si x_j est vrai :
  - Tous les v_i doivent correspondre à l'encodage de (j-1) (pour satisfaire $C_{i,j}$)
  - Tous les autres x_k doivent être faux (pour ne pas contredire au moins un des $C_{i,k}$)
- Si tous les x_j sont faux :
  - Tous les $C_{i,j}$ sont satisfaits, et n'importe quel assignement des v_i est cohérent.

[1] r : nombre de bit nécessaire à coder n en binaire

Contrainte "Au plus 1" : $\Sigma_{i\in\{1,\ldots,n\}} x_i \leq 1$

- Variables auxiliaires : $v_0, \ldots, v_{r-1}$ où $r = log_2\ n$[1].
  $v_0 \ldots v_n$ doit coder k tel que x_{k+1} vraie (s'il y en a)
- De ce fait, si $x_j = 1$ on doit avoir chaque $v_i = b^j_i$ où $b^j_0 \ldots b^j_r$ est le codage binaire de $j - 1$.
- Pour chaque $j \in \{1, \ldots, n\}$, pour chaque $i \in \{0, \ldots, r-1\}$, on ajoute la clause $C_{i,j} : (\neg x_j \vee l^j_i)$ où $l^j_i \equiv v_i$ si le ième bit b^j_i du codage binaire de $(j-1)$ vaut 1 et $l^j_i \equiv \neg v_i$ sinon.
- Si x_j est vrai :
  - Tous les v_i doivent correspondre à l'encodage de (j-1) (pour satisfaire $C_{i,j}$)
  - Tous les autres x_k doivent être faux (pour ne pas contredire au moins un des $C_{i,k}$)
- Si tous les x_j sont faux :
  - Tous les $C_{i,j}$ sont satisfaits, et n'importe quel assignement des v_i est cohérent.

[1] r : nombre de bit nécessaire à coder n en binaire

EXEMPLE

- Contrainte à traduire : $x_1 + x_2 + x_3 \leq 1$
- Variables auxiliaires : v_0, v_1 ($r = 2$ car $2^1 < 3 \leq 2^2$)
  - 00 ($\neg v_0 \wedge \neg v_1$) code x_1 ($j - 1 = 0$)
  - 01 ($\neg v_0 \wedge v_1$) code x_2 ($j - 1 = 1$)
  - 10 ($v_0 \wedge \neg v_1$) code x_3 ($j - 1 = 2$)
- Clauses introduites :
  - pour x_1 : $\neg x_1 \vee \neg v_0$ et $\neg x_1 \vee \neg v_1$
  - pour x_2 : $\neg x_2 \vee \neg v_0$ et $\neg x_2 \vee v_1$
  - pour x_1 : $\neg x_3 \vee v_0$ et $\neg x_3 \vee \neg v_1$
- Propagation unitaire de $x_1 = 1$
  - ($x_1 = 1$): $\neg x_1 \vee \neg v_0$ et $\neg x_1 \vee \neg v_1$ donnent $v_0 = 0$ et $v_1 = 0$
  - ($v_0 = 0$): $\neg x_3 \vee v_0$ donne $x_3 = 0$
  - ($v_1 = 0$): $\neg x_2 \vee v_1$ donne $x_2 = 0$

- "L'attribut *a* vaut *v*" : $a = v$ où $v \in D_a$ (par exemple $D_a = \{1, 2, 3\}$)
- Différent codage plus ou moins compacts / efficaces pour l'arithmétique
  - Encodage direct
  - Encodage par ordre (order encoding)
  - Encodage log (log encoding)

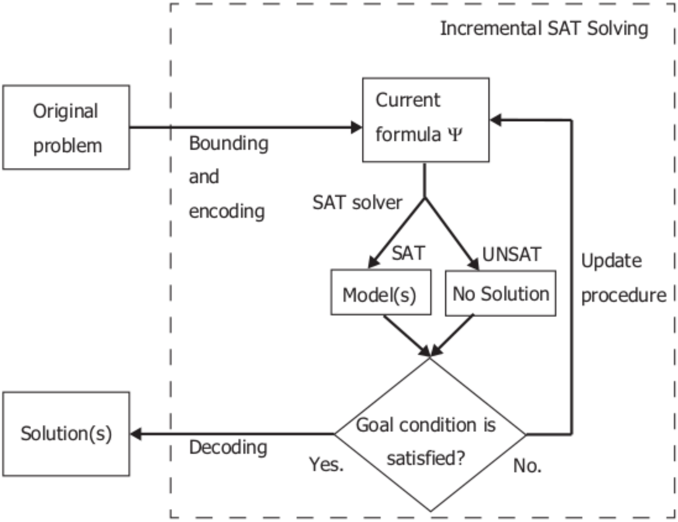
- Encodage direct :
  - pour chaque valeur possible $v \in D_a$ de a, on a une variable $x^{=}_{a,v}$ qui est vraie ssi $a = v$.
  - on a $\Sigma_{v \in D_a} x^{=}_{a,v} = 1$ (un attribut a une et une seule valeur)
  - $a = 2$ s'écrit : $x^{=}_{a,2}$
  - $a \neq 2$ s'écrit : $\neg x^{=}_{a,2}$
  - $a \leq 2$ s'écrit : $x^{=}_{a,1} \vee x^{=}_{a,2}$ ou $\neg x^{=}_{a,3}$

- Encodage par ordre (order encoding) :
  - pour chaque valeur possible $v \in D_a$ de a, on a une variable $x^{\leq}_{a,v}$ qui est vraie ssi $a \leq v$.
  - on a pour chaque $v \in D_a$ $\neg x^l eq_{a,v-1} \vee x^{\leq}_{a,v}$ (qui signifie $a \leq (v-1) \rightarrow a \leq v$)
  - $a = 2$ s'écrit : $x^{\leq}_{a,2} \wedge \neg x^{\leq}_{a,1}$ $(1 < a \leq 2)$
  - $a \neq 2$ s'écrit : $x^{\leq}_{a,1} \vee \neg x^{\leq}_{a,2}$ $(a \leq 1$ ou $a > 2)$
  - $a \leq 2$ s'écrit : $x^{\leq}_{a,2}$

- Encodage log (log encoding) :
  - On pose $r_a = log_2 Card(D_a)$. Pour chaque $i > r_a$, on a une variable $x^b_{a,i}$ qui est vraie ssi le ième bit du codage binaire de la valeur v de a vaut 1.
  - Si $Card(D_a)$ n'est pas une puissance de 2, il faut exclure les codes binaires des valeurs impossibles
  - $a = 2$ s'écrit : $\neg x^b_{a,0} \wedge x^b_{a,1}$
  - $a \neq 2$ s'écrit : $x^b_{a,0} \vee \neg x^b_{a,1}$
  - $a \leq 2$ s'écrit : $\neg x^b_{a,1} \vee \neg x^b_{a,0}$ (on aurait en plus $\neg x^b_{a,i}$ pour tout $i > 1$ si D_a était plus grand)

- Ces encodages sont à la base des traductions de CSP en SAT
  - Un CSP est donné par un triplet $\langle A, D, C \rangle$ où
    - A est un ensemble de variables a_i
    - D est un ensemble de domaines de valeurs D_a
    - C est un ensemble de contraintes sur les valeurs relatives des a_i, par exemple $a_1 \neq a_2$, $2a_1 = 10a_2 + a_3$ ou $a_1 a_2 \leq a_3$.
  - il s'agit de trouver un ensemble de valeurs pour les a_i prises dans les domaines de ces variables qui satisfassent toutes les contraintes.
- Exemple de solveur CSP basé sur un encodage en SAT : Sugar.

Résolution de problème itérative avec SAT
Principe



1. On pose k=1
2. Problème de planif à l'horizon *k* (trouver un plan en *k* actions)
3. On traduit en CNF
4. Si le problème est SAT, on traduit le modèle en plan et on renvoit la solution.
5. Sinon, on incrémente *k* de 1 et on revient à 2.

- Un état est un ensemble de finis de symboles propositionnels appelés fluents.
- Une action a est un triple $\langle pr, ad, de \rangle$ où pr, ad, de sont des ensembles de fluents. On les dénote respectivement par $Prec(a)$, $Add(a)$ et $Del(a)$. Il représentent les préconditions, effets d'ajout et effets de retraits de l'action a.
- Un problème de classification est un quadruplet $\langle F, A, I, B \rangle$ où
  - F est un ensemble fini de fluents,
  - A est un ensemble fini d'actions construites à partir de F,
  - $I \subset F$ représente l'état initial,
  - $B \subset F$ représente le but à atteindre.

- $P = \langle F, A, I, B \rangle$
- A horizon fini k, on considère qu'il y a k états successifs E_k. On a $E_0 = I$, on veut $E_k \supseteq B$, et les transitions entre E_i et E_{i+1} sont régies par l'application des actions de l'étape i.
- Variables propositionnelles
  - Pour chaque fluent $f \in F$ et étape $i \in \{0, \ldots, k\}$, on a la variable propositionnelle f_i qui est vraie ssi $f \in E_i$.
  - Pour chaque action $a \in A$ et étape $i \in \{0, \ldots, k\}$, on a la variable propositionnelle a_i qui est vraie ssi l'action a est executée à l'étape i (ce qui implique que $Prec(a) \subseteq E_{i-1}$ et que les effets de a soient appliqués à E_{i-1} pour obtenir E_i).

- Etat initial :

$$\Big(\bigwedge_{f\in I} f_0\Big) \wedge \Big(\bigwedge_{f\in F\setminus I} \neg f_0\Big)$$

- But atteint à l'état final :

$$\bigwedge_{f\in B} f_k$$

- Préconditions et effets des actions :

$$\bigwedge_{i\in\{1,\dots,k\}} \bigwedge_{a\in A} \Big[a_i \rightarrow \Big(\Big(\bigwedge_{f\in Prec(a)} f_{i-1}\Big) \wedge \Big(\bigwedge_{f\in Add(a)} f_i\Big) \wedge \Big(\bigwedge_{f\in Del(a)} \neg f_i\Big)\Big)\Big]$$

- Frames axiomes explicatifs de retraits : *un fluent ne peut devenir faux d'un état au suivant que du fait de l'application d'une action*

$$\bigwedge_{i\in\{1,\dots,k\}} \bigwedge_{f\in(I\cup F_{add})\cap F_{del}} \left[(f_{i-1} \wedge \neg f_i) \rightarrow \bigvee_{a\in\{b\in A | f\in Del(b)\}} a_i\right]$$

- Frames axiomes explicatifs d'ajouts : *idem pour le passage à Vrai*

$$\bigwedge_{i\in\{1,\dots,k\}} \bigwedge_{f\in((F\setminus I)\cup F_{del})\cap F_{add}} \left[(\neg f_{i-1} \wedge f_i) \rightarrow \bigvee_{a\in\{b\in A | f\in Add(b)\}} a_i\right]$$

- Interférences : *empêche l'exécution à la même étape de deux actions incompatibles*

$$\bigwedge_{i \in \{1,\dots,k\}} \ \bigwedge_{(a,a') \in \{(b,c) \in A^2 | (b||_e c) \wedge \neg (b||_i c)\}} [\neg a_i \vee \neg a'_i]$$

où

- $(a||_e b)$ ssi $(Add(a) \cap Del(b)) \cup (Add(b) \cap Del(a)) = \emptyset$
- $(a||_i b)$ ssi $(Prec(a) \cap Del(b)) \cup (Prec(b) \cap Del(a)) = \emptyset$.

Quelques extensions de SAT

Un large panel selon l'expressivité recherchée

- Traductions : CSP, ASP
- Extensions :
  - Max-SAT
  - Génération de MUS (Minimally Unsatisfiable Subformulae)
  - Formule booléennes quantifiées (QBF)
  - Gestion de contraintes pseudo-booléennes
  - SMT