Chapitre 5
Satisfaisabilité de formes conjonctives normales

Table de Matières

Le problème

- Satisfaisabilité d'une formule DNF: très facile
- Validité d'une formule CNF: très facile
- Beaucoup de problèmes concernent la satisfaisabilité d'une formule en CNF (ou facilement transformée en CNF):
- Problème de planification :
  - beaucoup de contraintes à résoudre (qui doivent toutes être satisfaites $\Rightarrow$ Conjonction)
  - chaque contrainte consiste en plusieurs alternatives ($\Rightarrow$ Disjonction)

Le problème

- Satisfaisabilité d'une formule DNF: très facile
- Validité d'une formule CNF: très facile
- Beaucoup de problèmes concernent la satisfaisabilité d'une formule en CNF (ou facilement transformée en CNF):
- Problème de planification :
  - beaucoup de contraintes à résoudre (qui doivent toutes être satisfaites $\Rightarrow$ Conjonction)
  - chaque contrainte consiste en plusieurs alternatives ($\Rightarrow$ Disjonction)

Le problème

- Satisfaisabilité d'une formule DNF: très facile
- Validité d'une formule CNF: très facile
- Beaucoup de problèmes concernent la satisfaisabilité d'une formule en CNF (ou facilement transformée en CNF):
- Problème de planification :
  - beaucoup de contraintes à résoudre (qui doivent toutes être satisfaites $\Rightarrow$ Conjonction)
  - chaque contrainte consiste en plusieurs alternatives ($\Rightarrow$ Disjonction)

Le problème

- Satisfaisabilité d'une formule DNF: très facile
- Validité d'une formule CNF: très facile
- Beaucoup de problèmes concernent la satisfaisabilité d'une formule en CNF (ou facilement transformée en CNF):
- Problème de planification :
  - beaucoup de contraintes à résoudre (qui doivent toutes être satisfaites $\Rightarrow$ Conjonction)
  - chaque contrainte consiste en plusieurs alternatives ($\Rightarrow$ Disjonction)

Le problème

- Satisfaisabilité d'une formule DNF: très facile
- Validité d'une formule CNF: très facile
- Beaucoup de problèmes concernent la satisfaisabilité d'une formule en CNF (ou facilement transformée en CNF):
- Problème de planification :
  - beaucoup de contraintes à résoudre (qui doivent toutes être satisfaites $\Rightarrow$ Conjonction)
  - chaque contrainte consiste en plusieurs alternatives ($\Rightarrow$ Disjonction)

Le problème

- Satisfaisabilité d'une formule DNF: très facile
- Validité d'une formule CNF: très facile
- Beaucoup de problèmes concernent la satisfaisabilité d'une formule en CNF (ou facilement transformée en CNF):
- Problème de planification :
  - beaucoup de contraintes à résoudre (qui doivent toutes être satisfaites $\Rightarrow$ Conjonction)
  - chaque contrainte consiste en plusieurs alternatives ($\Rightarrow$ Disjonction)

Table de vérité ?

- La table d'une vérité pour une formule avec n variable a 2^n lignes.
- Avec $n = 300$:

$$2^{300} = 2^{10^{30}} \approx 10^{3^{30}} = 10^{90}$$

- Nombre estimé de particules élémentaires dans l'univers connu : $10^{79} - 10^{81}$
- Des problèmes réalistes peuvent avoir facilement des milliers de variables.

Table de vérité ?

- La table d'une vérité pour une formule avec n variable a 2^n lignes.
- Avec $n = 300$:

$$2^{300} = 2^{10^{30}} \approx 10^{3^{30}} = 10^{90}$$

- Nombre estimé de particules élémentaires dans l'univers connu : $10^{79} - 10^{81}$
- Des problèmes réalistes peuvent avoir facilement des milliers de variables.

Table de vérité ?

- La table d'une vérité pour une formule avec n variable a 2^n lignes.
- Avec $n = 300$:

$$2^{300} = 2^{10^{30}} \approx 10^{3^{30}} = 10^{90}$$

- Nombre estimé de particules élémentaires dans l'univers connu : $10^{79} - 10^{81}$
- Des problèmes réalistes peuvent avoir facilement des milliers de variables.

Table de vérité ?

- La table d'une vérité pour une formule avec n variable a 2^n lignes.
- Avec $n = 300$:

$$2^{300} = 2^{10^{30}} \approx 10^{3^{30}} = 10^{90}$$

- Nombre estimé de particules élémentaires dans l'univers connu : $10^{79} - 10^{81}$
- Des problèmes réalistes peuvent avoir facilement des milliers de variables.

Table de vérité ?

- La table d'une vérité pour une formule avec n variable a 2^n lignes.
- Avec $n = 300$:

$$2^{300} = 2^{10^{30}} \approx 10^{3^{30}} = 10^{90}$$

- Nombre estimé de particules élémentaires dans l'univers connu : $10^{79} - 10^{81}$
- Des problèmes réalistes peuvent avoir facilement des milliers de variables.

Table de vérité ?

- La table d'une vérité pour une formule avec n variable a 2^n lignes.
- Avec $n = 300$:

$$2^{300} = 2^{10^{30}} \approx 10^{3^{30}} = 10^{90}$$

- Nombre estimé de particules élémentaires dans l'univers connu : $10^{79} - 10^{81}$
- Des problèmes réalistes peuvent avoir facilement des milliers de variables.

Table de vérité ?

- La table d'une vérité pour une formule avec n variable a 2^n lignes.
- Avec $n = 300$:

$$2^{300} = 2^{10^{30}} \approx 10^{3^{30}} = 10^{90}$$

- Nombre estimé de particules élémentaires dans l'univers connu : $10^{79} - 10^{81}$
- Des problèmes réalistes peuvent avoir facilement des milliers de variables.

L'algorithme DPLL

- Dû à Martin Davis, Hilary Putnam, George Logemann, Donald Loveland, 1962
- Parfois aussi (incorrectement) appelé algorithme de Davis-Putnam (DP)
- C'est la base des SAT-solveurs actuels.

L'algorithme DPLL

- Dû à Martin Davis, Hilary Putnam, George Logemann, Donald Loveland, 1962
- Parfois aussi (incorrectement) appelé algorithme de Davis-Putnam (DP)
- C'est la base des SAT-solveurs actuels.

L'algorithme DPLL

- Dû à Martin Davis, Hilary Putnam, George Logemann, Donald Loveland, 1962
- Parfois aussi (incorrectement) appelé algorithme de Davis-Putnam (DP)
- C'est la base des SAT-solveurs actuels.

Polarité d'une occurrence d'une variable

- Variable x apparaît avec polarité positive dans une clause

$$\ldots \vee x \vee \ldots$$

- Variable x apparaît avec polarité négative dans une clause

$$\ldots \vee \neg x \vee \ldots$$

- A priori, une variable peut apparaître à la fois avec polarité positive et avec polarité négative dans une clause :

$$y_1 \vee x \vee y_2 \vee \neg x \vee y_3$$

Polarité d'une occurrence d'une variable

- Variable x apparaît avec polarité positive dans une clause

$$\ldots \vee x \vee \ldots$$

- Variable x apparaît avec polarité négative dans une clause

$$\ldots \vee \neg x \vee \ldots$$

- A priori, une variable peut apparaître à la fois avec polarité positive et avec polarité négative dans une clause :

$$y_1 \vee x \vee y_2 \vee \neg x \vee y_3$$

Polarité d'une occurrence d'une variable

- Variable x apparaît avec polarité positive dans une clause

$$\ldots \vee x \vee \ldots$$

- Variable x apparaît avec polarité négative dans une clause

$$\ldots \vee \neg x \vee \ldots$$

- A priori, une variable peut apparaître à la fois avec polarité positive et avec polarité négative dans une clause :

$$y_1 \vee x \vee y_2 \vee \neg x \vee y_3$$

Hypothèse sur la forme des clauses

- Hypothèse : aucune variable apparaît deux fois dans la même clause. Justification :
- Si deux occurrences de la même polarité :

$$\ldots \vee x \vee \ldots \vee x \vee \ldots$$

on peut simplifier la clause

- Si deux occurrences de polarité opposée :

$$\ldots \vee x \vee \ldots \vee \neg x \vee \ldots$$

Tautologie, on peut supprimer la clause.

Hypothèse sur la forme des clauses

- Hypothèse : aucune variable apparaît deux fois dans la même clause. Justification :
- Si deux occurrences de la même polarité :

$$\ldots \vee x \vee \ldots \vee x \vee \ldots$$

on peut simplifier la clause

- Si deux occurrences de polarité opposée :

$$\ldots \vee x \vee \ldots \vee \neg x \vee \ldots$$

Tautologie, on peut supprimer la clause.

Hypothèse sur la forme des clauses

- Hypothèse : aucune variable apparaît deux fois dans la même clause. Justification :
- Si deux occurrences de la même polarité :

$$\ldots \vee x \vee \ldots \vee x \vee \ldots$$

on peut simplifier la clause

- Si deux occurrences de polarité opposée :

$$\ldots \vee x \vee \ldots \vee \neg x \vee \ldots$$

Tautologie, on peut supprimer la clause.

Mieux que *Générer et tester* ?

- Table de vérité : algorithme « générer et tester »
- Avantage de ce paradigme : séparation de l'algorithme en deux parties clairement distinguées.
- Inconvénient : parfois, on peut détecter qu'une formule n'est pas satisfaisable en essayant des valeurs pour quelques unes de ses variables seulement :

$$x \wedge (\neg x \vee y_1 \vee \neg y_2 \vee y_3) \wedge \neg x \wedge (y_2 \vee \neg y_4 \vee y_5)$$

(essayer les valeurs possibles pour x !)

Mieux que *Générer et tester* ?

- Table de vérité : algorithme « générer et tester »
- Avantage de ce paradigme : séparation de l'algorithme en deux parties clairement distinguées.
- Inconvénient : parfois, on peut détecter qu'une formule n'est pas satisfaisable en essayant des valeurs pour quelques unes de ses variables seulement :

$$x \wedge (\neg x \vee y_1 \vee \neg y_2 \vee y_3) \wedge \neg x \wedge (y_2 \vee \neg y_4 \vee y_5)$$

(essayer les valeurs possibles pour x !)

Mieux que *Générer et tester* ?

- Table de vérité : algorithme « générer et tester »
- Avantage de ce paradigme : séparation de l'algorithme en deux parties clairement distinguées.
- Inconvénient : parfois, on peut détecter qu'une formule n'est pas satisfaisable en essayant des valeurs pour quelques unes de ses variables seulement :

$$x \wedge (\neg x \vee y_1 \vee \neg y_2 \vee y_3) \wedge \neg x \wedge (y_2 \vee \neg y_4 \vee y_5)$$

(essayer les valeurs possibles pour x !)

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

$$d[x/b] = \begin{cases} \texttt{True} & \text{si } b = 1 \text{ et x apparait avec polarité positive en } d \\ \texttt{True} & \text{si } b = 0 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{\neg x\} & \text{si } b = 1 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{x\} & \text{si } b = 0 \text{ et x apparait avec polarité positive en } d \\ d & \text{si } x \text{ n'apparait pas en } d. \end{cases}$$

Ici : $d \setminus l$ est la clause d sans le littéral l

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

$$d[x/b] = \begin{cases} \texttt{True} & \text{si } b = 1 \text{ et x apparait avec polarité positive en } d \\ \texttt{True} & \text{si } b = 0 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{\neg x\} & \text{si } b = 1 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{x\} & \text{si } b = 0 \text{ et x apparait avec polarité positive en } d \\ d & \text{si } x \text{ n'apparait pas en } d. \end{cases}$$

Ici : $d \setminus l$ est la clause d sans le littéral l

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

$$d[x/b] = \begin{cases} \texttt{True} & \text{si } b = 1 \text{ et x apparait avec polarité positive en } d \\ \texttt{True} & \text{si } b = 0 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{\neg x\} & \text{si } b = 1 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{x\} & \text{si } b = 0 \text{ et x apparait avec polarité positive en } d \\ d & \text{si } x \text{ n'apparait pas en } d. \end{cases}$$

Ici : $d \setminus l$ est la clause d sans le littéral l

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

$$d[x/b] = \begin{cases} \texttt{True} & \text{si } b = 1 \text{ et x apparait avec polarité positive en } d \\ \texttt{True} & \text{si } b = 0 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{\neg x\} & \text{si } b = 1 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{x\} & \text{si } b = 0 \text{ et x apparait avec polarité positive en } d \\ d & \text{si } x \text{ n'apparait pas en } d. \end{cases}$$

Ici : $d \setminus l$ est la clause d sans le littéral l

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

$$d[x/b] = \begin{cases} \texttt{True} & \text{si } b = 1 \text{ et x apparait avec polarité positive en } d \\ \texttt{True} & \text{si } b = 0 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{\neg x\} & \text{si } b = 1 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{x\} & \text{si } b = 0 \text{ et x apparait avec polarité positive en } d \\ d & \text{si } x \text{ n'apparait pas en } d. \end{cases}$$

Ici : $d \setminus l$ est la clause d sans le littéral l

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

$$d[x/b] = \begin{cases} \texttt{True} & \text{si } b = 1 \text{ et x apparait avec polarité positive en } d \\ \texttt{True} & \text{si } b = 0 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{\neg x\} & \text{si } b = 1 \text{ et x apparait avec polarité négative en } d \\ d \setminus \{x\} & \text{si } b = 0 \text{ et x apparait avec polarité positive en } d \\ d & \text{si } x \text{ n'apparait pas en } d. \end{cases}$$

Ici : $d \setminus l$ est la clause d sans le littéral l

Rappel important

- `False` est une abréviation pour la clause disjonctive vide
- Si on supprime de la clause x le littéral x on obtient la clause vide notée `False`.

Rappel important

- `False` est une abréviation pour la clause disjonctive vide
- Si on supprime de la clause x le littéral x on obtient la clause vide notée `False`.

Exemples d'évaluation partielle d'une clause

$$
\begin{aligned}
(x \vee \neg y \vee z)[x/1] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/0] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/1] &= (\neg y \vee z) \\
(x \vee \neg y \vee z)[x/0] &= (\neg y \vee z) \\
(y_1 \vee \neg y_2 \vee z)[x/1] &= (y_1 \vee \neg y_2 \vee z) \\
x[x/0] &= \texttt{False}
\end{aligned}
$$

Exemples d'évaluation partielle d'une clause

$$
\begin{aligned}
(x \vee \neg y \vee z)[x/1] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/0] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/1] &= (\neg y \vee z) \\
(x \vee \neg y \vee z)[x/0] &= (\neg y \vee z) \\
(y_1 \vee \neg y_2 \vee z)[x/1] &= (y_1 \vee \neg y_2 \vee z) \\
x[x/0] &= \texttt{False}
\end{aligned}
$$

Exemples d'évaluation partielle d'une clause

$$
\begin{aligned}
(x \vee \neg y \vee z)[x/1] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/0] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/1] &= (\neg y \vee z) \\
(x \vee \neg y \vee z)[x/0] &= (\neg y \vee z) \\
(y_1 \vee \neg y_2 \vee z)[x/1] &= (y_1 \vee \neg y_2 \vee z) \\
x[x/0] &= \texttt{False}
\end{aligned}
$$

Exemples d'évaluation partielle d'une clause

$$
\begin{aligned}
(x \vee \neg y \vee z)[x/1] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/0] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/1] &= (\neg y \vee z) \\
(x \vee \neg y \vee z)[x/0] &= (\neg y \vee z) \\
(y_1 \vee \neg y_2 \vee z)[x/1] &= (y_1 \vee \neg y_2 \vee z) \\
x[x/0] &= \texttt{False}
\end{aligned}
$$

Exemples d'évaluation partielle d'une clause

$$
\begin{aligned}
(x \vee \neg y \vee z)[x/1] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/0] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/1] &= (\neg y \vee z) \\
(x \vee \neg y \vee z)[x/0] &= (\neg y \vee z) \\
(y_1 \vee \neg y_2 \vee z)[x/1] &= (y_1 \vee \neg y_2 \vee z) \\
x[x/0] &= \texttt{False}
\end{aligned}
$$

Exemples d'évaluation partielle d'une clause

$$
\begin{aligned}
(x \vee \neg y \vee z)[x/1] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/0] &= \texttt{True} \\
(\neg x \vee \neg y \vee z)[x/1] &= (\neg y \vee z) \\
(x \vee \neg y \vee z)[x/0] &= (\neg y \vee z) \\
(y_1 \vee \neg y_2 \vee z)[x/1] &= (y_1 \vee \neg y_2 \vee z) \\
x[x/0] &= \texttt{False}
\end{aligned}
$$

Évaluation partielle d'une formule en CNF

c : formule en forme CNF de la forme $d_1 \wedge \ldots \wedge d_n$

$$c[x/b] = \bigwedge_{\substack{1 \leq i \leq n \\ d_i[x/b] \neq \texttt{True}}} d_i[x/b]$$

C'est-à-dire :

- Évaluation partielle de toutes les clauses
- Supprimer les clauses évalues comme `True`

Exemples d'évaluation partielle d'une formule en CNF

$$
\begin{array}{lcccccc}
c & = & x & \wedge(\neg x \vee y_1 \vee \neg y_2 \vee y_3) & \wedge\neg x & \wedge(y_2 \vee \neg y_4 \vee y_5) \\
c[x/0] & = & \texttt{False} & \wedge\,\texttt{True} & \wedge\,\texttt{True} & \wedge\,(y_2 \vee \neg y_4 \vee y_5) \\
 & = & \texttt{False} & & & \\
c[x/1] & = & \texttt{True} & \wedge\,(y_1 \vee \neg y_2 \vee y_3) & \wedge\,\texttt{False} & \wedge\,(y_2 \vee \neg y_4 \vee y_5) \\
 & = & \texttt{False} & & &
\end{array}
$$

Exemples d'évaluation partielle d'une formule en CNF

$$
\begin{array}{lcccccc}
c & = & x & \wedge(\neg x \vee y_1 \vee \neg y_2 \vee y_3) & \wedge\neg x & \wedge(y_2 \vee \neg y_4 \vee y_5) \\
c[x/0] & = & \texttt{False} & \wedge\,\texttt{True} & \wedge\,\texttt{True} & \wedge\,(y_2 \vee \neg y_4 \vee y_5) \\
 & = & \texttt{False} & & & \\
c[x/1] & = & \texttt{True} & \wedge\,(y_1 \vee \neg y_2 \vee y_3) & \wedge\,\texttt{False} & \wedge\,(y_2 \vee \neg y_4 \vee y_5) \\
 & = & \texttt{False} & & &
\end{array}
$$

Exemples d'évaluation partielle d'une formule en CNF

$$
\begin{array}{lcccccc}
c & = & x & \wedge(\neg x \vee y_1 \vee \neg y_2 \vee y_3) & \wedge\neg x & \wedge(y_2 \vee \neg y_4 \vee y_5) \\
c[x/0] & = & \texttt{False} & \wedge\,\texttt{True} & \wedge\,\texttt{True} & \wedge\,(y_2 \vee \neg y_4 \vee y_5) \\
 & = & \texttt{False} & & & \\
c[x/1] & = & \texttt{True} & \wedge\,(y_1 \vee \neg y_2 \vee y_3) & \wedge\,\texttt{False} & \wedge\,(y_2 \vee \neg y_4 \vee y_5) \\
 & = & \texttt{False} & & &
\end{array}
$$

Exemples d'évaluation partielle d'une formule en CNF

$$
\begin{array}{lcccccc}
c & = & x & \wedge(\neg x \vee y_1 \vee \neg y_2 \vee y_3) & \wedge\neg x & \wedge(y_2 \vee \neg y_4 \vee y_5) \\
c[x/0] & = & \texttt{False} & \wedge\,\texttt{True} & \wedge\,\texttt{True} & \wedge\,(y_2 \vee \neg y_4 \vee y_5) \\
 & = & \texttt{False} & & & \\
c[x/1] & = & \texttt{True} & \wedge\,(y_1 \vee \neg y_2 \vee y_3) & \wedge\,\texttt{False} & \wedge\,(y_2 \vee \neg y_4 \vee y_5) \\
 & = & \texttt{False} & & &
\end{array}
$$

Exemples d'évaluation partielle d'une formule en CNF

$$
\begin{array}{lcccccc}
c & = & x & \wedge(\neg x \vee y_1 \vee \neg y_2 \vee y_3) & \wedge \neg x & \wedge(y_2 \vee \neg y_4 \vee y_5) \\
c[x/0] & = & \texttt{False} & \wedge \texttt{True} & \wedge \texttt{True} & \wedge (y_2 \vee \neg y_4 \vee y_5) \\
 & = & \texttt{False} & & & \\
c[x/1] & = & \texttt{True} & \wedge (y_1 \vee \neg y_2 \vee y_3) & \wedge \texttt{False} & \wedge (y_2 \vee \neg y_4 \vee y_5) \\
 & = & \texttt{False} & & &
\end{array}
$$

Propriétés de l'évaluation partielle

c : formule en forme CNF

- $$[\![c[x/b]]\!]v = [\![c]\!](v[x/b])$$

  pour toute affectation v et valeur booléenne b

- $$c \models c[x/0] \vee c[x/1]$$

Propriétés de l'évaluation partielle

c : formule en forme CNF

- $$[\![c[x/b]]\!]v = [\![c]\!](v[x/b])$$

  pour toute affectation v et valeur booléenne b

- $$c \models c[x/0] \vee c[x/1]$$

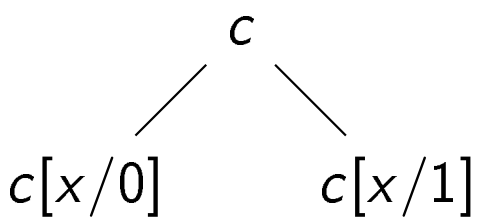
Un arbre de recherche

- Donc, c est satisfaisable si $c[x/0]$ est satisfaisable ou si $c[x/1]$ est satisfaisable.
- Cela donne lieu à un arbre de recherche car il faut a priori exploiter les deux possibilités :

- L'arbre de recherche décrit le déroulement du programme, il n'est pas entièrement représenté en mémoire.

Un arbre de recherche

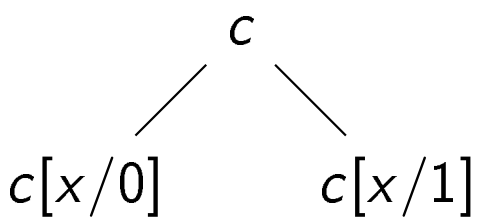
- Donc, c est satisfaisable si $c[x/0]$ est satisfaisable ou si $c[x/1]$ est satisfaisable.
- Cela donne lieu à un arbre de recherche car il faut a priori exploiter les deux possibilités :



- L'arbre de recherche décrit le déroulement du programme, il n'est pas entièrement représenté en mémoire.

Un arbre de recherche

- Donc, c est satisfaisable si $c[x/0]$ est satisfaisable ou si $c[x/1]$ est satisfaisable.
- Cela donne lieu à un arbre de recherche car il faut a priori exploiter les deux possibilités :



- L'arbre de recherche décrit le déroulement du programme, il n'est pas entièrement représenté en mémoire.

Un premier algorithme

1. Si c est la conjonction vide, notée `True`, alors c est même une tautologie, et donc en particulier satisfaisable.
2. Si c contient une clause qui est la disjonction vide, notée `False`, alors c n'est pas satisfaisable.
3. Sinon, choisir une variable pivot x, et continuer sur $c[x/0]$ et $c[x/1]$ (branchement dans l'arbre de recherche).

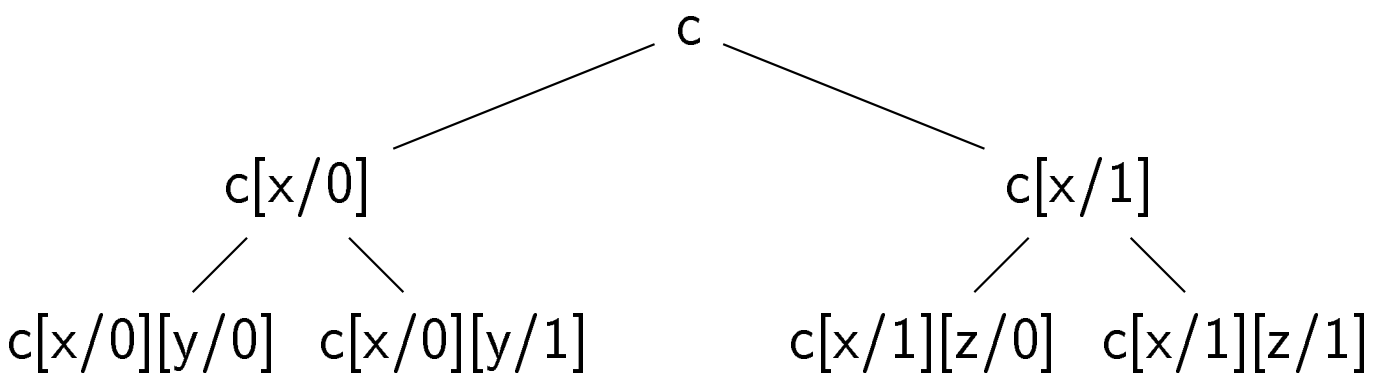
Un premier algorithme

1. Si c est la conjonction vide, notée `True`, alors c est même une tautologie, et donc en particulier satisfaisable.
2. Si c contient une clause qui est la disjonction vide, notée `False`, alors c n'est pas satisfaisable.
3. Sinon, choisir une variable pivot x, et continuer sur $c[x/0]$ et $c[x/1]$ (branchement dans l'arbre de recherche).

Un premier algorithme

1. Si c est la conjonction vide, notée `True`, alors c est même une tautologie, et donc en particulier satisfaisable.
2. Si c contient une clause qui est la disjonction vide, notée `False`, alors c n'est pas satisfaisable.
3. Sinon, choisir une variable pivot x, et continuer sur $c[x/0]$ et $c[x/1]$ (branchement dans l'arbre de recherche).

Exemple pour le premier algorithme



La construction de l'arbre s'arrête quand on peut conclure par l'évaluation partielle.

Comment choisir les variables de pivot ?

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

- Mauvais choix : $x_1 - x_2$
- Bon choix : x_3

Comment choisir les variables de pivot ?

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

- Mauvais choix : $x_1 - x_2$
- Bon choix : x_3

Comment choisir les variables de pivot ?

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

- Mauvais choix : $x_1 - x_2$
- Bon choix : x_3

Un bon choix de la variable pivot

- S'il y a une clause qui consiste en un seul littéral x ou $\neg x$ alors on choisit la variable x.

- Definition
  Une clause (conjonctive ou disjonctive) qui consiste en un seul littéral est appelée unitaire.

Un bon choix de la variable pivot

- S'il y a une clause qui consiste en un seul littéral x ou $\neg x$ alors on choisit la variable x.

- **Definition**
  Une clause (conjonctive ou disjonctive) qui consiste en un seul littéral est appelée unitaire.

Optimisation : Résolution unitaire

Si c contient la clause unitaire x :

- Le choix $x = 0$ donne forcement une formule c qui contient la clause `False`.
- c est satisfaisable si et seulement si $c[x/1]$ est satisfaisable.
- On peut donc passer de la formule c directement à la formule $c[x/1]$ qui est une formule plus petite, et cela sans d'avoir fait un choix.
- Analogue dans le cas d'une clause unaire $\neg x$

Optimisation : Résolution unitaire

Si c contient la clause unitaire x :

- Le choix $x = 0$ donne forcement une formule c qui contient la clause `False`.
- c est satisfaisable si et seulement si $c[x/1]$ est satisfaisable.
- On peut donc passer de la formule c directement à la formule $c[x/1]$ qui est une formule plus petite, et cela sans d'avoir fait un choix.
- Analogue dans le cas d'une clause unaire $\neg x$

Optimisation : Résolution unitaire

Si c contient la clause unitaire x :

- Le choix $x = 0$ donne forcement une formule c qui contient la clause `False`.
- c est satisfaisable si et seulement si $c[x/1]$ est satisfaisable.
- On peut donc passer de la formule c directement à la formule $c[x/1]$ qui est une formule plus petite, et cela sans d'avoir fait un choix.
- Analogue dans le cas d'une clause unaire $\neg x$

Optimisation : Résolution unitaire

Si c contient la clause unitaire x :

- Le choix $x = 0$ donne forcement une formule c qui contient la clause `False`.
- c est satisfaisable si et seulement si $c[x/1]$ est satisfaisable.
- On peut donc passer de la formule c directement à la formule $c[x/1]$ qui est une formule plus petite, et cela sans d'avoir fait un choix.
- Analogue dans le cas d'une clause unaire $\neg x$

Résolution unitaire

- Passer de $C_1 \wedge x \wedge C_2$ à $C_1[x/1] \wedge C_2[x/1]$
- Passer de $C_1 \wedge \neg x \wedge C_2$ à $C_1[x/0] \wedge C_2[x/0]$

Exemple de résolution unitaire

$$
\begin{array}{ll}
 & (x \vee y) \wedge \neg y \wedge (\neg x \vee y \vee \neg z) \quad \text{est satisfaisable} \\
\text{ssi} & x \wedge (\neg x \vee \neg z) \quad \text{est satisfaisable} \\
\text{ssi} & \neg z \quad \text{est satisfaisable} \\
\text{ssi} & \texttt{True} \quad \text{est satisfaisable}
\end{array}
$$

Exemple de résolution unitaire

$$
\begin{array}{lll}
 & (x \vee y) \wedge \neg y \wedge (\neg x \vee y \vee \neg z) & \text{est satisfaisable} \\
\text{ssi} & x \wedge (\neg x \vee \neg z) & \text{est satisfaisable} \\
\text{ssi} & \neg z & \text{est satisfaisable} \\
\text{ssi} & \texttt{True} & \text{est satisfaisable}
\end{array}
$$

Exemple de résolution unitaire

$$
\begin{array}{lll}
 & (x \vee y) \wedge \neg y \wedge (\neg x \vee y \vee \neg z) & \text{est satisfaisable} \\
\text{ssi} & x \wedge (\neg x \vee \neg z) & \text{est satisfaisable} \\
\text{ssi} & \neg z & \text{est satisfaisable} \\
\text{ssi} & \texttt{True} & \text{est satisfaisable}
\end{array}
$$

Exemple de résolution unitaire

$$
\begin{array}{lll}
 & (x \vee y) \wedge \neg y \wedge (\neg x \vee y \vee \neg z) & \text{est satisfaisable} \\
\text{ssi} & x \wedge (\neg x \vee \neg z) & \text{est satisfaisable} \\
\text{ssi} & \neg z & \text{est satisfaisable} \\
\text{ssi} & \texttt{True} & \text{est satisfaisable}
\end{array}
$$

Remarques sur l'exemple

- La résolution unitaire conserve la satisfaisabilité, mais ce n'est pas une transformation d'équivalence !
- Si la formule obtenue à la fin est satisfaisable alors la formule de départ l'est aussi.
- Par contre, si la formule à la fin est une tautologie il se peut que la formule de départ ne l'est pas.

Remarques sur l'exemple

- La résolution unitaire conserve la satisfaisabilité, mais ce n'est pas une transformation d'équivalence !
- Si la formule obtenue à la fin est satisfaisable alors la formule de départ l'est aussi.
- Par contre, si la formule à la fin est une tautologie il se peut que la formule de départ ne l'est pas.

Remarques sur l'exemple

- La résolution unitaire conserve la satisfaisabilité, mais ce n'est pas une transformation d'équivalence !
- Si la formule obtenue à la fin est satisfaisable alors la formule de départ l'est aussi.
- Par contre, si la formule à la fin est une tautologie il se peut que la formule de départ ne l'est pas.

Pourquoi n'est ce pas une transformation d'équivalence ?

- $x \wedge (\neg x \vee y)$ est transformée en y
- Regardez l'affectation $[x \mapsto 0, y \mapsto 1]$

Pourquoi n'est ce pas une transformation d'équivalence ?

- $x \wedge (\neg x \vee y)$ est transformée en y
- Regardez l'affectation $[x \mapsto 0, y \mapsto 1]$

Qu'est-ce qu'on a gagné ?

- La résolution unitaire peut faire apparaître de nouvelles clauses unitaires.
- Si on a la chance de cela, ça fait des simplifications en cascade.
- Quoi faire quand toutes les clauses contiennent au moins deux littéraux ?

Qu'est-ce qu'on a gagné ?

- La résolution unitaire peut faire apparaître de nouvelles clauses unitaires.
- Si on a la chance de cela, ça fait des simplifications en cascade.
- Quoi faire quand toutes les clauses contiennent au moins deux littéraux ?

Qu'est-ce qu'on a gagné ?

- La résolution unitaire peut faire apparaître de nouvelles clauses unitaires.
- Si on a la chance de cela, ça fait des simplifications en cascade.
- Quoi faire quand toutes les clauses contiennent au moins deux littéraux ?

Une optimisation

Proposition : Soit c en forme conjonctive normale.

- Si la variable x apparaît seulement avec polarité positive en c alors c est satisfaisable si et seulement si $c[x/1]$ est satisfaisable.
- Si la variable x apparaît seulement avec polarité négative en c alors c est satisfaisable si et seulement si $c[x/0]$ est satisfaisable.

Idée derrière cette optimisation

- Si une variable n'apparraît qu'avec polarité positive, alors ça ne peut pas faire du mal de la mettre à la valeur 1
- Si une variable n'apparraît qu'avec polarité négative, alors ça ne peut pas faire du mal de la mettre à la valeur 0

Idée derrière cette optimisation

- Si une variable n'apparraît qu'avec polarité positive, alors ça ne peut pas faire du mal de la mettre à la valeur 1
- Si une variable n'apparraît qu'avec polarité négative, alors ça ne peut pas faire du mal de la mettre à la valeur 0

Démonstration du premier énoncé

- Si $c[x/1]$ est satisfaisable, disons $v \models c[x/1]$, alors on a que

$$1 = [\![c[x/1]]\!]v = [\![c]\!](v[x/1])$$

et c est alors également satisfaisable.

- Soit c satisfaisable, disons $v \models c$. On peut montrer facilement que pour toute clause d en c on a que $[\![d]\!](v[x/1]) \geq [\![d]\!]v$ car x ne peut apparaître que positivement en d, et par conséquent que

$$[\![c[x/1]]\!]v = [\![c]\!](v[x/1]) \geq 1$$

Donc, $c[x/1]$ est satisfaisable.

Démonstration du premier énoncé

- Si $c[x/1]$ est satisfaisable, disons $v \models c[x/1]$, alors on a que

$$1 = [\![c[x/1]]\!]v = [\![c]\!](v[x/1])$$

et c est alors également satisfaisable.

- Soit c satisfaisable, disons $v \models c$. On peut montrer facilement que pour toute clause d en c on a que $[\![d]\!](v[x/1]) \geq [\![d]\!]v$ car x ne peut apparaître que positivement en d, et par conséquent que

$$[\![c[x/1]]\!]v = [\![c]\!](v[x/1]) \geq 1$$

Donc, $c[x/1]$ est satisfaisable.

Propriétés de cette règle

- Elle conserve la satisfaisabilité.
- Elle n'est pas une équivalence.
- Exemple $(x \vee y) \wedge (x \vee z)$ donne `True`.
  Affectation $[x \mapsto 0, y \mapsto 0, z \mapsto 0]$
- Cette règle peut déclencher une reduction en cascade (voir l'exemple suivant).

Propriétés de cette règle

- Elle conserve la satisfaisabilité.
- Elle n'est pas une équivalence.
- Exemple $(x \vee y) \wedge (x \vee z)$ donne `True`.
  Affectation $[x \mapsto 0, y \mapsto 0, z \mapsto 0]$
- Cette règle peut déclencher une reduction en cascade (voir l'exemple suivant).

Propriétés de cette règle

- Elle conserve la satisfaisabilité.
- Elle n'est pas une équivalence.
- Exemple $(x \vee y) \wedge (x \vee z)$ donne `True`.
  Affectation $[x \mapsto 0, y \mapsto 0, z \mapsto 0]$
- Cette règle peut déclencher une reduction en cascade (voir l'exemple suivant).

Propriétés de cette règle

- Elle conserve la satisfaisabilité.
- Elle n'est pas une équivalence.
- Exemple $(x \vee y) \wedge (x \vee z)$ donne `True`.
  Affectation $[x \mapsto 0, y \mapsto 0, z \mapsto 0]$
- Cette règle peut déclencher une reduction en cascade (voir l'exemple suivant).

Exemple

$$
\begin{array}{lll}
 & (x \vee \neg y) \wedge (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2) & \text{est satisfaisable} \\
\text{ssi} & (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2) & \text{est satisfaisable} \\
\text{ssi} & (\neg z_1 \vee \neg z_2) & \text{est satisfaisable} \\
\text{ssi} & \texttt{True} & \text{est satisfaisable}
\end{array}
$$

La formule de départ est donc satisfaisable.

Exemple

$$
\begin{array}{lll}
 & (x \vee \neg y) \wedge (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2) & \text{est satisfaisable} \\
\text{ssi} & (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2) & \text{est satisfaisable} \\
\text{ssi} & (\neg z_1 \vee \neg z_2) & \text{est satisfaisable} \\
\text{ssi} & \texttt{True} & \text{est satisfaisable}
\end{array}
$$

La formule de départ est donc satisfaisable.

Exemple

$$
\begin{array}{ll}
 & (x \vee \neg y) \wedge (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2) \quad \text{est satisfaisable} \\
\text{ssi} & (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2) \quad \text{est satisfaisable} \\
\text{ssi} & (\neg z_1 \vee \neg z_2) \quad \text{est satisfaisable} \\
\text{ssi} & \texttt{True} \quad \text{est satisfaisable}
\end{array}
$$

La formule de départ est donc satisfaisable.

Exemple

$$
\begin{array}{lll}
 & (x \vee \neg y) \wedge (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2) & \text{est satisfaisable} \\
\text{ssi} & (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2) & \text{est satisfaisable} \\
\text{ssi} & (\neg z_1 \vee \neg z_2) & \text{est satisfaisable} \\
\text{ssi} & \texttt{True} & \text{est satisfaisable}
\end{array}
$$

La formule de départ est donc satisfaisable.

c est une formule en forme CNF

function $dp(c) =$ case

(1) if $c = \texttt{True}$ then *true*

(2) if c contient une clause `False` then *false*

(3) if c contient une clause unitaire x then $dp(c[x/1])$

(4) if c contient une clause unitaire $\neg x$ then $dp(c[x/0])$

(5) if x n'apparaît qu'avec polarité positive en c then $dp(c[x/1])$

(6) if x n'apparaît qu'avec polarité négative en c then $dp(c[x/0])$

(7) else choisir $x \in \mathcal{V}(c)$; $dp(c[x/0])$ or $dp(c[x/1])$

c est une formule en forme CNF

function $dp(c) =$ `case`

(1) `if` $c =$ `True` `then` *true*

(2) `if` c contient une clause `False` `then` *false*

(3) `if` c contient une clause unitaire x `then` $dp(c[x/1])$

(4) `if` c contient une clause unitaire $\neg x$ `then` $dp(c[x/0])$

(5) `if` x n'apparaît qu'avec polarité positive en c `then` $dp(c[x/1])$

(6) `if` x n'apparaît qu'avec polarité négative en c `then` $dp(c[x/0])$

(7) `else choisir` $x \in \mathcal{V}(c)$; $dp(c[x/0])$ `or` $dp(c[x/1])$

c est une formule en forme CNF

function $dp(c) =$ `case`

(1) `if` $c = \texttt{True}$ `then` *true*

(2) `if` c contient une clause `False` `then` *false*

(3) `if` c contient une clause unitaire x `then` $dp(c[x/1])$

(4) `if` c contient une clause unitaire $\neg x$ `then` $dp(c[x/0])$

(5) `if` x n'apparaît qu'avec polarité positive en c `then` $dp(c[x/1])$

(6) `if` x n'apparaît qu'avec polarité négative en c `then` $dp(c[x/0])$

(7) `else choisir` $x \in \mathcal{V}(c)$; $dp(c[x/0])$ `or` $dp(c[x/1])$

c est une formule en forme CNF

function $dp(c) =$ `case`

(1) `if` $c = \texttt{True}$ `then` *true*

(2) `if` c contient une clause `False` `then` *false*

(3) `if` c contient une clause unitaire x `then` $dp(c[x/1])$

(4) `if` c contient une clause unitaire $\neg x$ `then` $dp(c[x/0])$

(5) `if` x n'apparaît qu'avec polarité positive en c `then` $dp(c[x/1])$

(6) `if` x n'apparaît qu'avec polarité négative en c `then` $dp(c[x/0])$

(7) `else choisir` $x \in \mathcal{V}(c)$; $dp(c[x/0])$ `or` $dp(c[x/1])$

c est une formule en forme CNF

function $dp(c) =$ `case`

(1) `if` $c = \texttt{True}$ `then` *true*

(2) `if` c contient une clause `False` `then` *false*

(3) `if` c contient une clause unitaire x `then` $dp(c[x/1])$

(4) `if` c contient une clause unitaire $\neg x$ `then` $dp(c[x/0])$

(5) `if` x n'apparaît qu'avec polarité positive en c `then` $dp(c[x/1])$

(6) `if` x n'apparaît qu'avec polarité négative en c `then` $dp(c[x/0])$

(7) `else choisir` $x \in \mathcal{V}(c)$; $dp(c[x/0])$ `or` $dp(c[x/1])$

c est une formule en forme CNF

function $dp(c) =$ `case`

(1)	`if` $c = \texttt{True}$	`then`	*true*
(2)	`if` c contient une clause `False`	`then`	*false*
(3)	`if` c contient une clause unitaire x	`then`	$dp(c[x/1])$
(4)	`if` c contient une clause unitaire $\neg x$	`then`	$dp(c[x/0])$
(5)	`if` x n'apparaît qu'avec polarité positive en c	`then`	$dp(c[x/1])$
(6)	`if` x n'apparaît qu'avec polarité négative en c	`then`	$dp(c[x/0])$
(7)	`else choisir` $x \in \mathcal{V}(c)$; $dp(c[x/0])$ `or` $dp(c[x/1])$		

c est une formule en forme CNF

function $dp(c) =$ `case`

(1) `if` $c = \texttt{True}$ `then` *true*

(2) `if` c contient une clause `False` `then` *false*

(3) `if` c contient une clause unitaire x `then` $dp(c[x/1])$

(4) `if` c contient une clause unitaire $\neg x$ `then` $dp(c[x/0])$

(5) `if` x n'apparaît qu'avec polarité positive en c `then` $dp(c[x/1])$

(6) `if` x n'apparaît qu'avec polarité négative en c `then` $dp(c[x/0])$

(7) `else choisir` $x \in \mathcal{V}(c)$; $dp(c[x/0])$ `or` $dp(c[x/1])$

c est une formule en forme CNF

function $dp(c) =$ `case`

(1) `if` $c = \texttt{True}$ `then` *true*

(2) `if` c contient une clause `False` `then` *false*

(3) `if` c contient une clause unitaire x `then` $dp(c[x/1])$

(4) `if` c contient une clause unitaire $\neg x$ `then` $dp(c[x/0])$

(5) `if` x n'apparaît qu'avec polarité positive en c `then` $dp(c[x/1])$

(6) `if` x n'apparaît qu'avec polarité négative en c `then` $dp(c[x/0])$

(7) `else choisir` $x \in \mathcal{V}(c)$; $dp(c[x/0])$ `or` $dp(c[x/1])$

Remarques sur l'algorithme

- Les tests différents dans l'instruction case peuvent être faits dans un ordre quelconque.
- On a donc la liberté de choisir sa stratégie dans l'implémentation de cet algorithme. Intéressant pour la priorité entre les cas (3) à (6).
- Les deux cas (5) et (6) sont parfois omis dans des implémentations car trop chers.

Remarques sur l'algorithme

- Les tests différents dans l'instruction case peuvent être faits dans un ordre quelconque.
- On a donc la liberté de choisir sa stratégie dans l'implémentation de cet algorithme. Intéressant pour la priorité entre les cas (3) à (6).
- Les deux cas (5) et (6) sont parfois omis dans des implémentations car trop chers.

Remarques sur l'algorithme

- Les tests différents dans l'instruction case peuvent être faits dans un ordre quelconque.
- On a donc la liberté de choisir sa stratégie dans l'implémentation de cet algorithme. Intéressant pour la priorité entre les cas (3) à (6).
- Les deux cas (5) et (6) sont parfois omis dans des implémentations car trop chers.

Terminaison de l'algorithme

- Dans chaque appel récursif de l'algorithme, le nombre de variables dans la formule est décrémenté.
- Le temps d'exécution peut quand même, dans le pire des cas, être exponentiel dans le nombre de variables (à cause du branchement).

Terminaison de l'algorithme

- Dans chaque appel récursif de l'algorithme, le nombre de variables dans la formule est décrémenté.
- Le temps d'exécution peut quand même, dans le pire des cas, être exponentiel dans le nombre de variables (à cause du branchement).

Formule de départ

$$
\begin{aligned}
&(x_1 \vee \neg x_2 \vee y_1 \vee \neg y_2 \vee \neg z_2 \vee \neg z_4) \\
&\wedge (x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2) \\
&\wedge (x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3) \\
&\wedge (\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1) \\
&\qquad \wedge (\neg x_2 \vee \neg z_1 \vee z_2) \\
&\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)
\end{aligned}
$$

Formule de départ

$$
\begin{aligned}
&(x_1 \vee \neg x_2 \vee y_1 \vee \neg y_2 \vee \neg z_2 \vee \neg z_4) \\
&\wedge (x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2) \\
&\wedge (x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3) \\
&\wedge (\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1) \\
&\qquad \wedge (\neg x_2 \vee \neg z_1 \vee z_2) \\
&\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)
\end{aligned}
$$

x_1 apparaît seulement avec polarité positive $\Rightarrow$ supprimer la première clause

Formule de départ

$$(x_1 \vee \neg x_2 \vee y_1 \vee \neg y_2 \vee \neg z_2 \vee \neg z_4)$$
$$\wedge(x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge(x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge(\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1)$$
$$\wedge(\neg x_2 \vee \neg z_1 \vee z_2)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

x_1 apparaît seulement avec polarité positive $\Rightarrow$ supprimer la première clause

Après suppression de la première clause

$$(x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge (x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge (\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1)$$
$$\wedge (\neg x_2 \vee \neg z_1 \vee z_2)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Après suppression de la première clause

$$(x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge (x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge (\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1)$$
$$\wedge (\neg x_2 \vee \neg z_1 \vee z_2)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Seulement (7) s'applique $\Rightarrow$ choisir comme variable de pivot x_2

- Cas 1 : $x_2 = 0$
- Cas 2 : $x_2 = 1$

Cas 1 : $x_2 = 0$

$$(x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge (x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge (\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1)$$
$$\wedge (\neg x_2 \vee \neg z_1 \vee z_2)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

- Supprimer x_2 dans toutes les clauses
- Supprimer toutes les clauses qui contiennent $\neg x_2$

Cas 1 : $x_2 = 0$

$$(x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge (x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge (\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1)$$
$$\wedge (\neg x_2 \vee \neg z_1 \vee z_2)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

- Supprimer x_2 dans toutes les clauses
- Supprimer toutes les clauses qui contiennent $\neg x_2$

Cas 1 : $x_2 = 0$

$$(x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge (x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge (\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1)$$
$$\wedge (\neg x_2 \vee \neg z_1 \vee z_2)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

- Supprimer x_2 dans toutes les clauses
- Supprimer toutes les clauses qui contiennent $\neg x_2$

Cas 1 : Après avoir mis $x_2 = 0$

$$y_1 \wedge (y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (\neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge(\neg y_1 \vee z_3 \vee \neg z_4) \wedge (\neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 1 : Après avoir mis $x_2 = 0$

$$y_1 \wedge (y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (\neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge(\neg y_1 \vee z_3 \vee \neg z_4) \wedge (\neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Clause unitaire: $y_1 \Rightarrow$ résolution unitaire

Cas 1 : Après avoir mis $x_2 = 0$

$$y_1 \wedge (y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (\neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge(\neg y_1 \vee z_3 \vee \neg z_4) \wedge (\neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 1 : $x_2 = 0$, après résolution unitaire avec y_1

$$(\neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge (z_3 \vee \neg z_4) \wedge (\neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 1 : $x_2 = 0$, après résolution unitaire avec y_1

$$(\neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge (z_3 \vee \neg z_4) \wedge (\neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

y_2 n'apparait qu'avec polarité négative $\Rightarrow$ supprimer les deux clauses qui contiennent le littéral $\neg y_2$

Cas 1 : $x_2 = 0$, après résolution unitaire avec y_1

$$(\neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge (z_3 \vee \neg z_4) \wedge (\neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

y_2 n'apparait qu'avec polarité négative $\Rightarrow$ supprimer les deux clauses qui contiennent le littéral $\neg y_2$

Cas 1 : après avoir mis $y_2 = 0$

$$(z_3 \vee \neg z_4)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 1 : après avoir mis $y_2 = 0$

$$(z_3 \vee \neg z_4)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas (7), variable de pivot: z_3

- Sous-cas 1 : $z_3 = 0$
- Sous-cas 2 : $z_3 = 1$

Cas 1, sous-cas 1: $z_3 = 0$

$$(z_3 \vee \neg z_4) \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 1, sous-cas 1: $z_3 = 0$

$$(z_3 \vee \neg z_4) \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

- Supprimer z_3 dans toutes les clause
- Supprimer toutes les clauses qui contiennet $\neg z_3$

Cas 1, sous-cas 1: $z_3 = 0$

$$(z_3 \vee \neg z_4)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

- Supprimer z_3 dans toutes les clause
- Supprimer toutes les clauses qui contiennet $\neg z_3$

Cas 1, sous-cas 1 : $z_3 = 0$

$$\neg z_4 \wedge z_4$$

Cas 1, sous-cas 1 : $z_3 = 0$

$$\neg z_4 \wedge z_4$$

Donne *false* par application de (3) (ou de (4))

Cas 1, sous-cas 2: $z_3 = 1$

$$(z_3 \vee \neg z_4)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 1, sous-cas 2: $z_3 = 1$

$$(z_3 \vee \neg z_4)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

- Supprimer $\neg z_3$ dans toutes les clause
- Supprimer toutes les clauses qui contiennet z_3

Cas 1, sous-cas 2: $z_3 = 1$

$$(z_3 \vee \neg z_4)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

- Supprimer $\neg z_3$ dans toutes les clause
- Supprimer toutes les clauses qui contiennet z_3

Cas 1, sous-cas 2 : $z_3 = 1$

$$\neg z_4 \wedge z_4$$

Cas 1, sous-cas 2 : $z_3 = 1$

$$\neg z_4 \wedge z_4$$

Donne *false* par application de (3) (ou de (4))

Cas 2 : $x_2 = 1$

$$(x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge (x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge(\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1)$$
$$\wedge(\neg x_2 \vee \neg z_1 \vee z_2)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

- Supprimer $\neg x_2$ dans toutes les clauses
- Supprimer toutes les clauses qui contiennent x_2

Cas 2 : $x_2 = 1$

$$(x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge (x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge(\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1)$$
$$\wedge(\neg x_2 \vee \neg z_1 \vee z_2)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

- Supprimer $\neg x_2$ dans toutes les clauses
- Supprimer toutes les clauses qui contiennent x_2

Cas 2 : $x_2 = 1$

$$(x_2 \vee y_1) \wedge (x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2)$$
$$\wedge(x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3)$$
$$\wedge(\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1)$$
$$\wedge(\neg x_2 \vee \neg z_1 \vee z_2)$$
$$\wedge(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

- Supprimer $\neg x_2$ dans toutes les clauses
- Supprimer toutes les clauses qui contiennent x_2

Cas 2 : Après avoir choisi $x_2 = 1$

$$\neg y_1 \wedge (\neg y_1 \vee \neg y_2) \wedge (y_1 \vee y_2) \wedge (\neg y_2 \vee z_1) \wedge (\neg z_1 \vee z_2)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2 : Après avoir choisi $x_2 = 1$

$$\neg y_1 \wedge (\neg y_1 \vee \neg y_2) \wedge (y_1 \vee y_2) \wedge (\neg y_2 \vee z_1) \wedge (\neg z_1 \vee z_2)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Clause unitaire $\neg y_1$

Cas 2 : Après avoir choisi $x_2 = 1$

$$\neg y_1 \wedge (\neg y_1 \vee \neg y_2) \wedge (y_1 \vee y_2) \wedge (\neg y_2 \vee z_1) \wedge (\neg z_1 \vee z_2)$$
$$\wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, après résolution unitaire

$$y_2 \wedge (\neg y_2 \vee z_1) \wedge (\neg z_1 \vee z_2) \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, après résolution unitaire

$$y_2 \wedge (\neg y_2 \vee z_1) \wedge (\neg z_1 \vee z_2) \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Clause unitaire y_2

Cas 2, après résolution unitaire

$$y_2 \wedge (\neg y_2 \vee z_1) \wedge (\neg z_1 \vee z_2) \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, après nouvelle résolution unitaire

$$z_1 \wedge (\neg z_1 \vee z_2) \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, après nouvelle résolution unitaire

$$z_1 \wedge (\neg z_1 \vee z_2) \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Clause unitaire z_1

Cas 2, après nouvelle résolution unitaire

$$z_1 \wedge (\neg z_1 \vee z_2) \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, après encore une résolution unitaire

$$z_2 \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, après encore une résolution unitaire

$$z_2 \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Clause unitaire z_2

Cas 2, après encore une résolution unitaire

$$z_2 \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, après encore une résolution unitaire

$$(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, après encore une résolution unitaire

$$(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Appliquer (7), variable de pivot : z_3

- Sous-cas 1 : $z_3 = 0$
- Sous-cas 2 : $z_3 = 1$

Cas 2, sous-cas 1 : choisir $z_3 = 0$

$$(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, sous-cas 1 : choisir $z_3 = 0$

$$(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, sous-cas 1 : choisir $z_3 = 0$

$$(\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)$$

Cas 2, sous-cas 1 : Après avoir choisi $z_3 = 0$

z_4

Cas 2, sous-cas 1 : Après avoir choisi $z_3 = 0$

z_4

Application de (3) (ou alternativement (5)) donne *true*

Cas 2, sous-cas 1 : Après avoir choisi $z_3 = 0$

z_4

Application de (3) (ou alternativement (5)) donne *true*
Résultat : **satisfaisable**

DPLL et DPLL dual

- DPLL : satisfaisabilité de formules en CNF
- DPLL dual : validité de formules en DNF

Formule en	DNF	CNF
Satisfaisabilité :	trivial	DPLL
Validité :	DPLL dual	trivial

DPLL et DPLL dual

- DPLL : satisfaisabilité de formules en CNF
- DPLL dual : validité de formules en DNF

Formule en	DNF	CNF
Satisfaisabilité :	trivial	DPLL
Validité :	DPLL dual	trivial

DPLL et DPLL dual

- DPLL : satisfaisabilité de formules en CNF
- DPLL dual : validité de formules en DNF

Formule en	DNF	CNF
Satisfaisabilité :	trivial	DPLL
Validité :	DPLL dual	trivial

L'état de l'art aujourd'hui

- DPLL n'est pas suffisamment efficace pour des très grandes formules.
- À la fin des années 80 : SAT-solveurs basés sur la notion d'apprentissage : analyse des raison d'échec quand une branche mène vers une formule non-satisfaisable.
- Quelques outils : `Chaff`, `GRASP`, MiniSat.

L'état de l'art aujourd'hui

- DPLL n'est pas suffisamment efficace pour des très grandes formules.
- À la fin des années 80 : SAT-solveurs basés sur la notion d'apprentissage : analyse des raison d'échec quand une branche mène vers une formule non-satisfaisable.
- Quelques outils : `Chaff`, `GRASP`, MiniSat.

L'état de l'art aujourd'hui

- DPLL n'est pas suffisamment efficace pour des très grandes formules.
- À la fin des années 80 : SAT-solveurs basés sur la notion d'apprentissage : analyse des raison d'échec quand une branche mène vers une formule non-satisfaisable.
- Quelques outils : `Chaff`, `GRASP`, MiniSat.