

Chapitre 5

Satisfaisabilité de formes conjonctives normales

Table de Matières

Vers un algorithme efficace

Une stratégie d'énumération, et une optimisation

Variables qui apparaissent avec une seule polarité

L'algorithme DPLL complet

Un exemple complet

Conclusion

Le problème

12100

22/03

- Satisfaisabilité d'une formule DNF: très facile

Le problème

- ▶ Satisfaisabilité d'une formule DNF: très facile
- ▶ Validité d'une formule CNF: très facile

Le problème

- ▶ Satisfaisabilité d'une formule DNF: très facile
- ▶ Validité d'une formule CNF: très facile
- ▶ Beaucoup de problèmes concernant la satisfaisabilité d'une formule en CNF (ou facilement transformée en CNF):

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 :

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)

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.

Table de vérité ?

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

$$2^{300}$$

Table de vérité ?

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

$$2^{300} = 2^{10^{30}}$$

Table de vérité ?

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

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

Table de vérité ?

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

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

Table de vérité ?

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

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

- ▶ Nombre estimé de particules élémentaires dans l'univers connu : $10^{79} - 10^{81}$

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

L'algorithme DPLL

- ▶ Dû à Martin Davis, Hilary Putnam, George Logemann, Donald Loveland, 1962
- ▶ Parfois aussi (incorrectement) appelé algorithme de Davis-Putnam (DP)

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

$$\dots \vee x \vee \dots$$

Polarité d'une occurrence d'une variable

- ▶ Variable x apparaît avec **polarité positive** dans une clause

$$\dots \vee x \vee \dots$$

- ▶ Variable x apparaît avec **polarité négative** dans une clause

$$\dots \vee \neg x \vee \dots$$

Polarité d'une occurrence d'une variable

- ▶ Variable x apparaît avec **polarité positive** dans une clause

$$\dots \vee x \vee \dots$$

- ▶ Variable x apparaît avec **polarité négative** dans une clause

$$\dots \vee \neg x \vee \dots$$

- ▶ **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 n'apparaît deux fois dans la même clause. Justification :

Hypothèse sur la forme des clauses

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

$$\dots \vee x \vee \dots \vee x \vee \dots$$

on peut simplifier la clause (ne laisser qu'une)

Hypothèse sur la forme des clauses

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

$$\dots \vee x \vee \dots \vee x \vee \dots$$

on peut simplifier la clause (ne laisser qu'une)

- ▶ Si deux occurrences de polarité opposée :

$$\dots \vee x \vee \dots \vee \neg x \vee \dots$$

Tautologie, on peut supprimer la clause.

Mieux que *Générer et tester* ?

- ▶ Table de vérité : algorithme « générer et tester »

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.

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

$$d[x/b] = \left\{ \right.$$

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

$$d[x/b] = \begin{cases} \top & \text{si } b = 1 \text{ et } x \text{ apparait avec polarité positive en } d \\ \text{ } & \text{ } \end{cases}$$

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

$$d[x/b] = \begin{cases} \top & \text{si } b = 1 \text{ et } x \text{ apparait avec polarité positive en } d \\ \top & \text{si } b = 0 \text{ et } x \text{ apparait avec polarité négative en } d \end{cases}$$

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

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

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

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

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

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

Évaluation partielle d'une clause disjonctive

d : clause disjonctive

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

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

$\{2, 2\}$

Rappel important

- ▶ \perp est une abréviation pour la clause disjonctive vide

Rappel important

- ▶ \perp est une abréviation pour la clause disjonctive vide
- ▶ Si on supprime de la clause x le littéral x on obtient la clause vide notée \perp .

Exemples d'évaluation partielle d'une clause

$$(x \vee \neg y \vee z)[x/1] = \top$$

Exemples d'évaluation partielle d'une clause

$$(x \vee \neg y \vee z)[x/1] = \top$$

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

Exemples d'évaluation partielle d'une clause

$$(x \vee \neg y \vee z)[x/1] = \top$$

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

$$(\neg x \vee \neg y \vee z)[x/1] = (\neg y \vee z)$$

Exemples d'évaluation partielle d'une clause

$$(x \vee \neg y \vee z)[x/1] = \top$$

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

$$(\neg x \vee \neg y \vee z)[x/1] = (\neg y \vee z)$$

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

Exemples d'évaluation partielle d'une clause

$$(x \vee \neg y \vee z)[x/1] = \top$$

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

$$(\neg x \vee \neg y \vee z)[x/1] = (\neg y \vee z)$$

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

$$(y_1 \vee \neg y_2 \vee z)[x/1] = (y_1 \vee \neg y_2 \vee z)$$

Exemples d'évaluation partielle d'une clause

$$(x \vee \neg y \vee z)[x/1] = \top$$

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

$$(\neg x \vee \neg y \vee z)[x/1] = (\neg y \vee z)$$

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

$$(y_1 \vee \neg y_2 \vee z)[x/1] = (y_1 \vee \neg y_2 \vee z)$$

$$x[x/0] = \perp$$

Évaluation partielle d'une formule en CNF

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

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

C'est-à-dire :

- ▶ Évaluation partielle de toutes les clauses
- ▶ Supprimer les clauses évaluées comme \top

Exemples d'évaluation partielle d'une formule en CNF

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

Exemples d'évaluation partielle d'une formule en CNF

$$\begin{array}{lcl}
 c & = & x \wedge (\neg x \vee y_1 \vee \neg y_2 \vee y_3) \wedge \neg x \wedge (y_2 \vee \neg y_4 \vee y_5) \\
 c[x/0] & = & \perp \wedge \top \wedge \top \wedge (y_2 \vee \neg y_4 \vee y_5)
 \end{array}$$

\uparrow
 faux

Exemples d'évaluation partielle d'une formule en CNF

$$\begin{aligned}
 c &= x \wedge (\neg x \vee y_1 \vee \neg y_2 \vee y_3) \wedge \neg x \wedge (y_2 \vee \neg y_4 \vee y_5) \\
 c[x/0] &= \perp \wedge \top \wedge \top \wedge (y_2 \vee \neg y_4 \vee y_5) \\
 &= \perp \rightarrow \text{car } \text{False} \wedge \text{True} \wedge \text{True} \wedge \text{?} \\
 &= \text{pas satisfaisable}
 \end{aligned}$$

Exemples d'évaluation partielle d'une formule en CNF

$$\begin{array}{lcl}
 c & = & x \wedge (\neg x \vee y_1 \vee \neg y_2 \vee y_3) \wedge \neg x \wedge (y_2 \vee \neg y_4 \vee y_5) \\
 c[x/0] & = & \perp \wedge \top \wedge \top \wedge (y_2 \vee \neg y_4 \vee y_5) \\
 & = & \perp \\
 c[x/1] & = & \top \wedge (y_1 \vee \neg y_2 \vee y_3) \wedge \perp \wedge (y_2 \vee \neg y_4 \vee y_5)
 \end{array}$$

Exemples d'évaluation partielle d'une formule en CNF

$$\begin{array}{lcl}
 c & = & x \wedge (\neg x \vee y_1 \vee \neg y_2 \vee y_3) \wedge \neg x \wedge (y_2 \vee \neg y_4 \vee y_5) \\
 c[x/0] & = & \perp \wedge \top \wedge \top \wedge (y_2 \vee \neg y_4 \vee y_5) \\
 & = & \perp \\
 c[x/1] & = & \top \wedge (y_1 \vee \neg y_2 \vee y_3) \wedge \perp \wedge (y_2 \vee \neg y_4 \vee y_5) \\
 & = & \perp
 \end{array}$$

Propriétés de l'évaluation partielle

Donné c , une formule en forme CNF. La propriété qu'on va utiliser est la suivante :

c satisfaisable $\Leftrightarrow c[x/0]$ satisfaisable ou $c[x/1]$ satisfaisable

Rappel : Pour toute affectation v et valeur booléenne b ,

$$\llbracket c[x/b] \rrbracket v = \llbracket c \rrbracket (v[x/b])$$

Preuve.

Propriétés de l'évaluation partielle

Donné c , une formule en forme CNF. La propriété qu'on va utiliser est la suivante :

c satisfaisable $\Leftrightarrow c[x/0]$ satisfaisable ou $c[x/1]$ satisfaisable

Rappel : Pour toute affectation v et valeur booléenne b ,

$$\llbracket c[x/b] \rrbracket v = \llbracket c \rrbracket (v[x/b])$$

Preuve.

\Rightarrow) À prouver : $c \models c[x/0] \vee c[x/1]$

*Si c est satisfaisable alors — satisfaisable
→ conséquence logique*

$$\llbracket c \rrbracket v = 1 \Rightarrow \llbracket c \rrbracket (v[x/0]) = 1 \text{ ou } \llbracket c \rrbracket (v[x/1]) = 1 \Rightarrow$$

$$\llbracket c[x/0] \rrbracket v = 1 \text{ ou } \llbracket c[x/1] \rrbracket v = 1 \Rightarrow \llbracket c[x/0] \vee c[x/1] \rrbracket v = 1$$

Propriétés de l'évaluation partielle

Donné c , une formule en forme CNF. La propriété qu'on va utiliser est la suivante :

c satisfaisable $\Leftrightarrow c[x/0]$ satisfaisable ou $c[x/1]$ satisfaisable

Rappel : Pour toute affectation v et valeur booléenne b ,

$$\llbracket c[x/b] \rrbracket v = \llbracket c \rrbracket (v[x/b])$$

Preuve.

\Leftarrow) À prouver : c contradiction $\Rightarrow c[x/0] \wedge c[x/1]$ contradiction.

$$\forall v, \llbracket c \rrbracket v = 0 \Rightarrow \forall v, \llbracket c \rrbracket (v[x/0]) = 0 \text{ et } \llbracket c \rrbracket (v[x/1]) = 0 \Rightarrow$$

$$\forall v, \llbracket c[x/0] \rrbracket v = 0 \text{ et } \llbracket c[x/1] \rrbracket v = 0 \Rightarrow \forall v, \llbracket c[x/0] \wedge [x/1] \rrbracket v = 0$$



Propriétés de l'évaluation partielle

Remarque : L'évaluation partielle conserve la satisfaisabilité, mais ne produit pas nécessairement une formule équivalente ! En effet, en général $c[x/0] \vee c[x/1] \not\equiv c$

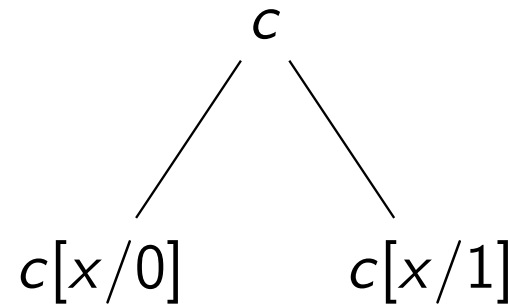
- ▶ Par exemple, la formule $x \wedge (\neg x \vee y)$ est transformée en y . Or l'affectation $[x \mapsto 0, y \mapsto 1]$ satisfait y mais pas $x \wedge (\neg x \vee y)$.
- ▶ Intuition : la formule produite a une variable de moins.
- ▶ Voir le td.

Un arbre de recherche

- ▶ c satisfaisable $\Leftrightarrow c[x/0]$ satisfaisable ou $c[x/1]$ satisfaisable.

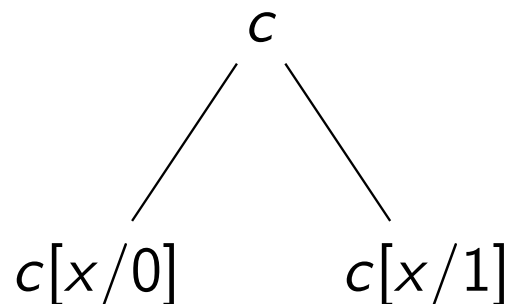
Un arbre de recherche

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



Un arbre de recherche

- ▶ c satisfaisable $\Leftrightarrow c[x/0]$ satisfaisable ou $c[x/1]$ 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 \top , alors c est même une tautologie, et donc en particulier satisfaisable.

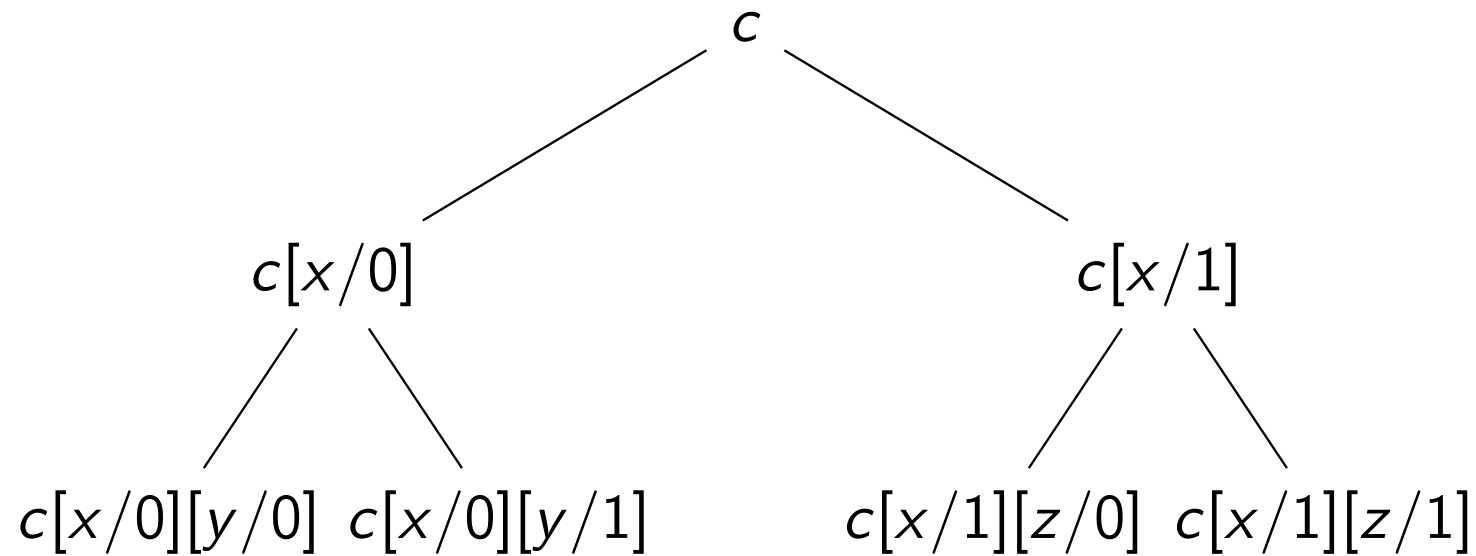
Un premier algorithme

1. Si c est la conjonction vide, notée \top , alors c est même une tautologie, et donc en particulier satisfaisable.
2. Si c contient une clause qui est la disjonction vide, notée \perp , alors c n'est pas satisfaisable.

Un premier algorithme

1. Si c est la conjonction vide, notée \top , alors c est même une tautologie, et donc en particulier satisfaisable.
2. Si c contient une clause qui est la disjonction vide, notée \perp , 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).
 \rightsquigarrow Ce dernier s'appelle **résolution**.

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 ?

29 / 63 10 u 22

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

Comment choisir les variables de pivot ?

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

► Mauvais choix : $x_1 - x_2$

Comment choisir les variables de pivot ?

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

- 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 .

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 forcément une formule c qui contient la clause \perp .

Optimisation : Résolution unitaire

Si c contient la clause unitaire x :

- ▶ Le choix $x = 0$ donne forcément une formule c qui contient la clause \perp .
- ▶ c est satisfaisable si et seulement si $c[x/1]$ est satisfaisable.

Optimisation : Résolution unitaire

Si c contient la clause unitaire x :

- ▶ Le choix $x = 0$ donne forcément une formule c qui contient la clause \perp .
- ▶ 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.

Optimisation : Résolution unitaire

Si c contient la clause unitaire x :

- ▶ Le choix $x = 0$ donne forcément une formule c qui contient la clause \perp .
- ▶ 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.
- ▶ Analogie dans le cas d'une clause unitaire $\neg x$

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

$(x \vee y) \wedge \neg y \wedge (\neg x \vee y \vee \neg z)$ est satisfaisable

Exemple de résolution unitaire

$(x \vee y) \wedge \neg y \wedge (\neg x \vee y \vee \neg z)$ est satisfaisable
ssi $x \wedge (\neg x \vee \neg z)$ est satisfaisable

Exemple de résolution unitaire

| | | |
|-----|---|-------------------|
| | $(x \vee y) \wedge \neg y \wedge (\neg x \vee y \vee \neg z)$ | est satisfaisable |
| ssi | $x \wedge (\neg x \vee \neg z)$ | est satisfaisable |
| ssi | $\neg z$ | est satisfaisable |

Exemple de résolution unitaire

| | | |
|-----|---|-------------------|
| | $(x \vee y) \wedge \neg y \wedge (\neg x \vee y \vee \neg z)$ | est satisfaisable |
| ssi | $x \wedge (\neg x \vee \neg z)$ | est satisfaisable |
| ssi | $\neg z$ | est satisfaisable |
| ssi | \top | est satisfaisable |

Qu'est-ce qu'on a gagné ?

- ▶ La résolution unitaire peut faire apparaître de nouvelles clauses unitaires.

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.

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'apparaît qu'avec polarité positive, alors ça ne peut pas faire du mal de la mettre à la valeur 1

Idée derrière cette optimisation

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

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

et c est alors également satisfaisable.

Démonstration du premier énoncé

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

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

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 $\llbracket d \rrbracket (v[x/1]) \geq \llbracket d \rrbracket v$ car x ne peut apparaître que positivement en d , et par conséquent que

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

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

Propriétés de cette règle

- ▶ Elle conserve la **satisfaisabilité**.

Propriétés de cette règle

- ▶ Elle conserve la **satisfaisabilité**.
- ▶ Elle n'est pas une équivalence.

Propriétés de cette règle

- ▶ Elle conserve la **satisfaisabilité**.
- ▶ Elle n'est pas une équivalence.
- ▶ Exemple $(x \vee y) \wedge (x \vee z)$ donne \top .
Affectation $[x \mapsto 0, y \mapsto 0, z \mapsto 0]$

Propriétés de cette règle

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

Exemple

$(\textcolor{red}{x} \vee \neg y) \wedge (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2)$ est satisfaisable

Exemple

ssi $(\textcolor{red}{x} \vee \neg y) \wedge (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2)$ est satisfaisable
 $(\textcolor{red}{y} \vee z_1) \wedge (\neg z_1 \vee \neg z_2)$ est satisfaisable

Exemple

| | | |
|-----|---|-------------------|
| | $(x \vee \neg y) \wedge (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2)$ | est satisfaisable |
| ssi | $(y \vee z_1) \wedge (\neg z_1 \vee \neg z_2)$ | est satisfaisable |
| ssi | $(\neg z_1 \vee \neg z_2)$ | est satisfaisable |

Exemple

| | | |
|-----|--|-------------------|
| | $(\textcolor{red}{x} \vee \neg y) \wedge (y \vee z_1) \wedge (\neg z_1 \vee \neg z_2)$ | est satisfaisable |
| ssi | $(\textcolor{red}{y} \vee z_1) \wedge (\neg z_1 \vee \neg z_2)$ | est satisfaisable |
| ssi | $(\neg \textcolor{red}{z}_1 \vee \neg z_2)$ | est satisfaisable |
| ssi | \top | est satisfaisable |

La formule de départ est donc satisfaisable.

c est une formule en forme CNF

function $dp(c) = \text{case}$

(1)

(2)

(3)

(4)

(5)

(6)

(7)

c est une formule en forme CNF

```
function  $dp(c)$  = case  
(1) if  $c = \top$  then true  
(2)  
(3)  
(4)  
(5)  
(6)  
(7)
```

c est une formule en forme CNF

```
function  $dp(c)$  = case  
(1) if  $c = \top$  then true  
(2) if  $c$  contient une clause  $\perp$  then false  
(3)  
(4)  
(5)  
(6)  
(7)
```

c est une formule en forme CNF

function $dp(c) = \text{case}$

- (1) *if* $c = \top$ *then true*
- (2) *if* c contient une clause \perp *then false*
- (3) *if* c contient une clause unitaire x *then* $dp(c[x/1])$
- (4)
- (5)
- (6)
- (7)

c est une formule en forme CNF

function $dp(c) = \text{case}$

- (1) *if* $c = \top$ *then* *true*
- (2) *if* c contient une clause \perp *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)
- (6)
- (7)

c est une formule en forme CNF

function $dp(c) = \text{case}$

- (1) *if* $c = \top$ *then* *true*
- (2) *if* c contient une clause \perp *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)
- (7)

c est une formule en forme CNF

function $dp(c) = \text{case}$

- (1) *if* $c = \top$ *then* *true*
- (2) *if* c contient une clause \perp *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)

c est une formule en forme CNF

function $dp(c) = \text{case}$

- (1) *if* $c = \top$ *then* *true*
- (2) *if* c contient une clause \perp *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.

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).

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.

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

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

$\underset{1}{\neg z_3} \quad \underset{0}{\neg z_4} \quad \underset{0}{z_3} \quad \underset{1}{z_4} \quad \underset{1}{\neg z_3} \quad \underset{1}{z_4}$

$$z_3 = 0$$

$$z_4 = 1$$

Formule de départ

Si x_1 est True \Rightarrow toute la clause est vraie

$$(\textcolor{red}{x}_1 \vee \neg x_2 \vee y_1 \vee \neg y_2 \vee \neg z_2 \vee \neg z_4)$$

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

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

$$\wedge (\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1)$$

$$\wedge (\neg x_2 \vee \neg z_1 \vee z_2)$$

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

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

Formule de départ

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

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

Après suppression de la première clause

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

Après suppression de la première clause

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

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

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

Cas 1 : $x_2 = 0$

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

Cas 1 : $x_2 = 0$

$$\begin{aligned} & (\textcolor{red}{x}_2 \vee y_1) \wedge (\textcolor{red}{x}_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (\textcolor{red}{x}_2 \vee \neg y_2 \vee z_1 \vee \neg z_2) \\ & \wedge (\textcolor{red}{x}_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (\textcolor{red}{x}_2 \vee \neg y_2 \vee z_2 \vee \neg z_3) \\ & \wedge (\neg \textcolor{red}{x}_2 \vee \neg y_1) \wedge (\neg \textcolor{red}{x}_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg \textcolor{red}{x}_2 \vee y_1 \vee y_2) \wedge (\neg \textcolor{red}{x}_2 \vee \neg y_2 \vee z_1) \\ & \quad \wedge (\neg \textcolor{red}{x}_2 \vee \neg z_1 \vee z_2) \\ & \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4) \end{aligned}$$

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

Cas 1 : $x_2 = 0$

$$\begin{aligned}
& \overset{\text{Vrai}}{(x_2 \vee y_1)} \wedge \overset{T}{(x_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4)} \wedge \overset{T}{(x_2 \vee \neg y_2 \vee z_1 \vee \neg z_2)} \\
& \wedge \overset{T}{(x_2 \vee \neg y_1 \vee z_3 \vee \neg z_4)} \wedge \overset{T}{(x_2 \vee \neg y_2 \vee z_2 \vee \neg z_3)} \\
& \wedge (\neg x_2 \vee \neg y_1) \wedge (\neg x_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg x_2 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_2 \vee z_1) \\
& \quad \wedge (\neg x_2 \vee \neg z_1 \vee z_2) \overset{T}{} \\
& \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4)
\end{aligned}$$

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

Cas 1 : Après avoir mis $x_2 = 0$

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

Cas 1 : Après avoir mis $x_2 = 0$

$$\begin{aligned} & \textcolor{red}{y_1} \wedge (y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (\neg y_2 \vee z_1 \vee \neg z_2) \\ & \wedge (\neg y_1 \vee z_3 \vee \neg z_4) \wedge (\neg y_2 \vee z_2 \vee \neg z_3) \\ & \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4) \end{aligned}$$

Clause unitaire: $y_1 \Rightarrow$ résolution unitaire

Cas 1 : Après avoir mis $x_2 = 0$

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

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'apparaît 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

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

y_2 n'apparaît 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: $z_3 = 0$

$$\begin{aligned} & (\textcolor{red}{z}_3 \vee \neg z_4) \\ & \wedge (\neg \textcolor{red}{z}_3 \vee \neg z_4) \wedge (\textcolor{red}{z}_3 \vee z_4) \wedge (\neg \textcolor{red}{z}_3 \vee z_4) \end{aligned}$$

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

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 contiennent $\neg z_3$

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: $z_3 = 1$

$$\begin{aligned} & (z_3 \vee \neg z_4) \\ & \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4) \end{aligned}$$

- ▶ Supprimer $\neg z_3$ dans toutes les clause
- ▶ Supprimer toutes les clauses qui contiennent 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 contiennent 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 : $x_2 = 1$

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

Cas 2 : $x_2 = 1$

$$\begin{aligned} & (\textcolor{red}{x}_2 \vee y_1) \wedge (\textcolor{red}{x}_2 \vee y_1 \vee y_2 \vee z_1 \vee z_4) \wedge (\textcolor{red}{x}_2 \vee \neg y_2 \vee z_1 \vee \neg z_2) \\ & \wedge (\textcolor{red}{x}_2 \vee \neg y_1 \vee z_3 \vee \neg z_4) \wedge (\textcolor{red}{x}_2 \vee \neg y_2 \vee z_2 \vee \neg z_3) \\ & \wedge (\neg \textcolor{red}{x}_2 \vee \neg y_1) \wedge (\neg \textcolor{red}{x}_2 \vee \neg y_1 \vee \neg y_2) \wedge (\neg \textcolor{red}{x}_2 \vee y_1 \vee y_2) \wedge (\neg \textcolor{red}{x}_2 \vee \neg y_2 \vee z_1) \\ & \quad \wedge (\neg \textcolor{red}{x}_2 \vee \neg z_1 \vee z_2) \\ & \wedge (\neg z_3 \vee \neg z_4) \wedge (z_3 \vee z_4) \wedge (\neg z_3 \vee z_4) \end{aligned}$$

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

Cas 2 : $x_2 = 1$

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

- 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 et DPLL dual

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

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.

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 raisons d'échec quand une branche mène vers une formule non-satisfaisable.

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 raisons d'échec quand une branche mène vers une formule non-satisfaisable.
- ▶ Quelques outils : Chaff, GRASP, MiniSat.