Résolution Propositionnelle

Deuxième Partie : Algorithmes

Benjamin Wack

Université Grenoble Alpes

Janvier 2025

Preuve par résolution de l'exemple du cours

- (H1) : $p \Rightarrow \neg j \equiv \neg p \vee \neg j$
- (H2) : $\neg p \Rightarrow j \equiv p \vee j$
- (H3) : $j \Rightarrow m \equiv \neg j \vee m$
- ($\neg$ C) : $\neg m \wedge \neg p$

Clauses : $\{\neg p \vee \neg j,\ p \vee j,\ \neg j \vee m,\ \neg m,\ \neg p\}$

$$\dfrac{\dfrac{\dfrac{p \vee j \qquad \neg j \vee m}{p \vee m} \qquad \neg m}{p} \qquad \neg p}{\bot} \quad \text{OU} \quad \dfrac{\dfrac{\dfrac{p \vee j \qquad \neg p}{j} \qquad \neg j \vee m}{m} \qquad \neg m}{\bot}$$

OU...

Au dernier cours

- Algèbre de Boole
- Fonctions booléennes
- Résolution

(1) $A \vdash B$

B se *déduit* de A : il existe une preuve par résolution de B à partir de A.

(2) $A \models B$

B est *conséquence* de A : tout modèle de A est aussi un modèle de B.

Aujourd'hui : Cohérence

(1) $\Rightarrow$ (2)

Aujourd'hui : Complétude

(2) $\Rightarrow$ (1)

Plan

Définition

La cohérence d'un système logique, c'est le fait que les preuves obtenues dans ce système ≪ ne prouvent que des choses vraies ≫.

Cohérence de la règle de résolution

Théorème 2.1.15

Si C est un résolvant de A et B alors $A, B \models C$.

Preuve.

Si C est un résolvant de A et B, alors il y a un littéral L tel que $A = A' + L$, $B = B' + L^c$, et $C = A' + B'$.

Soit v telle que $[A]_v = 1$ et $[B]_v = 1$: montrons que $[C]_v = 1$.

- Si $[L]_v = 1$. Alors $[L^c]_v = 0$.
  Comme $[B]_v = 1$, v est modèle de B'. Donc $[C]_v = 1$.
- Si $[L^c]_v = 1$. Alors $[L]_v = 0$.
  Comme $[A]_v = 1$, v est modèle de A'. Donc $[C]_v = 1$.

Dans tous les cas v est modèle de C.

□

Cohérence de la déduction

Théorème 2.1.16

Soit Γ un ensemble de clauses et C une clause. Si $\Gamma \vdash C$ alors $\Gamma \models C$.

Preuve.

Soit P une preuve de C à partir de Γ.
Supposons que pour toute preuve $\Gamma \vdash D$ plus courte que P, nous avons $\Gamma \models D$.
Montrons que $\Gamma \models C$. Nous avons deux cas possibles :

1. C est une hypothèse de Γ, dans ce cas évidemment $\Gamma \models C$.
2. $\Gamma \vdash A$ et $\Gamma \vdash B$ (avec une preuve plus courte) et

$$\frac{A \qquad B}{C}$$

Par hypothèse de récurrence : $\Gamma \models A$ et $\Gamma \models B$.
Par cohérence de la règle de résolution : $A, B \models C$. Donc $\Gamma \models C$.

□

Définition

Complétude pour la réfutation : Si $\Gamma \models \bot$ alors $\Gamma \vdash \bot$.

$\Gamma[L := 1]$

Définition 2.1.18

Soient Γ un ensemble de clauses et L un littéral.

$\Gamma[L := 1]$ est obtenu en :

- supprimant les clauses contenant L
- enlevant L^c des autres clauses

De même $\Gamma[L := 0]$ en échangeant les rôles de L et L^c.

Remarque : on a une variable de moins.

Exemples

Exemple 2.1.19

$\Gamma = \overline{p}+q, \overline{q}+r, p+q, p+r.$

- $\Gamma[p := 1] =$

  $\{q, \overline{q}+r\}.$

- $\Gamma[p := 0] =$

  $\{\overline{q}+r, q, r\}.$

Observons que :

- $(\overline{1}+q)(\overline{q}+r)(1+q)(1+r) = q(\overline{q}+r)$
- $(\overline{0}+q)(\overline{q}+r)(0+q)(0+r) = (\overline{q}+r)qr$

Propriété de $\Gamma[L := \ldots]$

Propriété 2.1.21

Γ a un modèle si et seulement si $\Gamma[L := 1]$ ou $\Gamma[L := 0]$ en a un.

Preuve.

$\Rightarrow$ Si v est un modèle de Γ alors c'est un modèle de l'un des $\Gamma[L := \ldots]$ (à choisir selon $[L]_v$).

$\Leftarrow$ Si v est un modèle de $\Gamma[L := i]$ alors on peut en déduire un modèle de Γ (en posant $[L]_{v'} = i$).

□

Lemme 2.1.22

Lemme 2.1.22

Soit Γ un ensemble de clauses, C une clause et L un littéral.
Si $\Gamma[L := 1] \vdash C$ alors $\Gamma \vdash C$ ou $\Gamma \vdash C + L^c$.

Preuve.

On rajoute le littéral L^c aux clauses où on l'avait enlevé dans Γ.

- Si $C \in \Gamma[L := 1]$:
  - soit C était dans Γ alors $\Gamma \vdash C$
  - soit C est obtenue en enlevant un L^c alors $\Gamma \vdash C + L^c$
- Si C est un résolvant de A et B :
  - soit $\Gamma \vdash A$ et $\Gamma \vdash B$ d'où $\Gamma \vdash C$
  - soit il faut rajouter L^c dans A ou B, donc dans C aussi

□

Complétude de la résolution

Théorème 2.1.24

Soit Γ un ensemble fini de clauses. Si $\Gamma \models \bot$ alors $\Gamma \vdash \bot$.

Preuve

Par récurrence sur le nombre de variables de Γ.

- Cas de base : Γ n'a aucune variable. Donc $\Gamma = \emptyset$ (impossible car $\emptyset$ est valide) ou $\Gamma = \{\bot\}$.
- Hérédité : on montre que $\Gamma \vdash \bot$ ou bien que $\Gamma \vdash \bar{x}$ et $\Gamma \vdash x$.

Corollaire 2.1.25

Γ est insatisfaisable si et seulement si $\Gamma \vdash \bot$.

Présentation de deux algorithmes

Comment décider ≪ systématiquement ≫ si Γ est contradictoire ?

- L'algorithme de Davis, Putnam, Logemann et Loveland
  Parcours ≪ intelligent ≫ des assignations possibles des variables propositionnelles de Γ
- La Stratégie complète
  Construction des TOUTES les clauses déductibles de Γ.

Remarque

Solutions exponentielles en temps dans le pire des cas.

Pourquoi une complexité exponentielle ?

Rappels :

- dans une clause, l'ordre ou la répétition de littéraux n'a pas d'importance ;
- une clause valide (contenant p et $\bar{p}$) est inutile ;
- dans une preuve, les clauses sont réutilisables.

Combien de clauses distinctes peut-on écrire avec n variables ?

Pour chaque variable p, une clause peut contenir :

- p
- $\bar{p}$
- ou aucun des deux

donc 3^n clauses distinctes.

Ainsin si Γ contient n littéraux, aucune preuve par résolution ne peut être de longueur supérieure à 3^n lignes.

Réduction d'un ensemble de clauses

Pour accélérer les algorithmes, on réduit l'ensemble de clauses.

Comment réduire ?

Enlever les clauses valides et les clauses contenant une autre clause de l'ensemble.

Exemple 2.1.27

La réduction de l'ensemble de clauses
$\{p+q+\overline{p},\ p+r,\ p+r+\overline{s},\ r+q\}$ donne l'ensemble réduit :

$\{p+r,\ r+q\}$.

Justification

Propriété 2.1.28

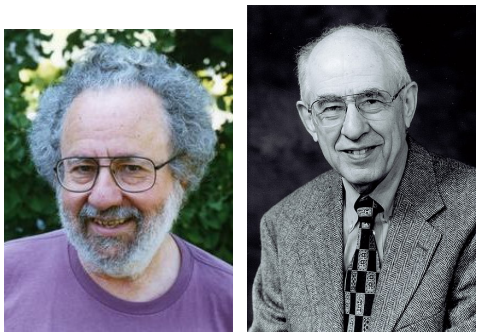
Un ensemble de clauses E est équivalent à l'ensemble de clauses réduit obtenu à partir de E.

Preuve.

- On peut enlever une clause valide : $x.1 \equiv x$
- On peut enlever une clause incluant une autre clause : $x.(x+y) \equiv x$

□

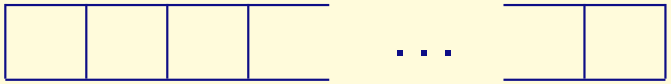
Historique



- Martin Davis (1928-2023), mathématicien américain
- Hilary Putnam (1926-2016), philosophe, mathématicien et informaticien américain
- règle de **résolution** (utilisation exhaustive dans l'algo initial)
- Algorithme de satisfaisabilité des formules booléennes (1960)
  - détermine un modèle d'un ensemble de clauses (si possible)
  - conçu initialement pour étudier les formules du premier ordre
  - raffinement par M. Davis, G. Logemann et D. Loveland en 1962
  - à la base des SAT-solveurs efficaces
- Preuve d'indécidabilité des équations Diophantiennes (avec Y. Matiyasevich et J. Robinson)

Démo SAT solveur

Problème



- Chaque case peut contenir un jeton ou pas.
- Deux jetons ne doivent jamais être voisins.
- Au moins deux cases doivent contenir un jeton.

Modélisation booléenne

La variable *i* est vraie si la case *i* contient un jeton.

Résolution par un SAT solveur

- Construction des clauses
- Traduction au format DIMACS
- Exécution et interprétation du résultat

Principes de simplification (Davis & Putnam)

Deux types de transformations pour simplifier les formules :

1. préservant le sens :
   - réduction
2. préservant seulement la satisfaisabilité :
   - suppression des clauses qui ont des littéraux isolés
   - résolution unitaire

DPLL est (en général) efficace car il utilise ces deux types de transformations.

Principe de séparation (Logemann & Loveland)

≪ Branchement/Retour-arrière ≫

- **Branchement** : Après toutes les simplifications, affecter **vrai** à une variable bien choisie.
- Continuer récursivement l'algorithme.
- **Retour-arrière** : Si on arrive à une contradiction, on retourne au dernier choix, et on ≪ branche ≫ en affectant **faux** à la variable choisie.

L'algorithme DPLL (figure 2.1)

bool fonction Algo_DPLL(Γ : ensemble de clauses)

0 Supprimer les clauses valides de Γ.
Si $\Gamma = \emptyset$, renvoyer (`vrai`).
Sinon renvoyer (`DPLL`(Γ))

bool fonction `DPLL`(Γ : ensemble de clauses non valides)
La fonction renvoie vrai si et seulement si Γ est satisfaisable.

1 **Si** $\bot \in \Gamma$, renvoyer(`faux`).
Si $\Gamma = \emptyset$, renvoyer (`vrai`).
2 Réduire Γ.
3 Enlever de Γ les clauses comportant des littéraux isolés.
Si l'ensemble Γ a été modifié, aller en 1.
4 Appliquer à Γ la résolution unitaire.
Si l'ensemble Γ a été modifié, aller en 1.
5 Choisir x une variable quelconque de Γ
renvoyer (`DPLL`($\Gamma[x := 0]$) ou alors `DPLL`($\Gamma[x := 1]$))

Suppression des clauses qui ont des littéraux isolés.

Définition 2.2.1

Le littéral L est **isolé** si aucune clause de Γ ne comporte L^c.

Lemme 2.2.2

Supprimer des clauses avec un littéral isolé préserve la satisfaisabilité.

La preuve est demandée dans l'exercice 49.

Intuition : il n'y a rien à perdre à prendre $[L]_v = 1$.

Exemple 2.2.3

Soit Γ l'ensemble de clauses

(1) $p + q + r$

(2) $\bar{q} + \bar{r}$

(3) $q + s$

(4) $\bar{s} + t$

Simplifiez Γ en supprimant des clauses qui ont des littéraux isolés.

p et t sont isolés.

(2) $\bar{q} + \bar{r}$

(3) $q + s$

$\bar{r}$ et s sont maintenant isolés.
On obtient l'ensemble vide $\emptyset$.
Γ a donc un modèle (par exemple $p = 1, t = 1, r = 0, s = 1$).

Résolution unitaire

Définition 2.2.4

Une clause unitaire est une clause qui ne comporte qu'un littéral.

Soit L le littéral d'une clause unitaire de Γ.
On construit un ensemble Θ :

- enlever les clauses qui comportent L
- à **l'intérieur** des clauses restantes, enlever L^c
- si Γ comporte deux clauses unitaires complémentaires, alors $\Theta = \{\bot\}$.

On applique ce procédé pour chaque clause unitaire.

Lemme 2.2.5

Γ a un modèle si et seulement si Θ en a un.

La preuve est demandée dans l'exercice 50.

Exemple 2.2.6 Résolution unitaire

Simplifiez les ensembles de clauses suivants par résolution unitaire :

- $\Gamma = p + q,\ \bar{p},\ \bar{q}$

  $q,\ \bar{q}$ par résolution unitaire sur $\bar{p}$,
  puis $\perp$ par RU sur $\bar{q}$
  Donc Γ n'a pas de modèle.

- $\Gamma = a + b + \bar{d},\ \bar{a} + c + \bar{d},\ \bar{b},\ d,\ \bar{c}$

  1. $a,\ \bar{a}$
  2. $\perp$

  donc Γ n'a pas de modèle.

- $\Gamma = p,\ q,\ p + r,\ \bar{p} + r,\ q + \bar{r},\ \bar{q} + s$

  Par RU on obtient $r,\ s$.
  Donc Γ a un modèle puisque r, s en a un.

L'algorithme DPLL (figure 2.1)

bool fonction Algo_DPLL(Γ : ensemble de clauses)

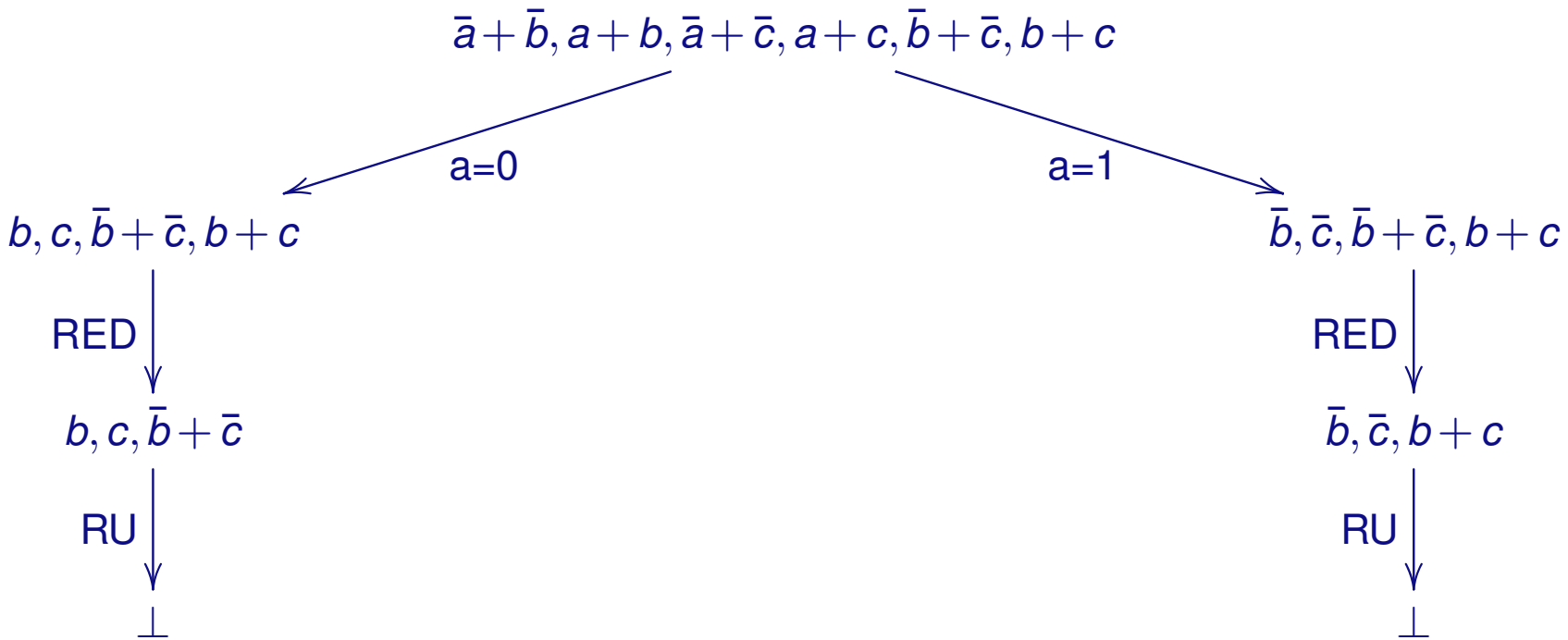
0 Supprimer les clauses valides de Γ.
Si $\Gamma = \emptyset$, renvoyer (`vrai`).
Sinon renvoyer (`DPLL`(Γ))

bool fonction `DPLL`(Γ : ensemble de clauses non valides)
La fonction renvoie vrai si et seulement si Γ est satisfaisable.

1 **Si** $\bot \in \Gamma$, renvoyer(`faux`).
Si $\Gamma = \emptyset$, renvoyer (`vrai`).
2 Réduire Γ.
3 Enlever de Γ les clauses comportant des littéraux isolés.
Si l'ensemble Γ a été modifié, aller en 1.
4 Appliquer à Γ la résolution unitaire.
Si l'ensemble Γ a été modifié, aller en 1.
5 Choisir x une variable quelconque de Γ
renvoyer (`DPLL`($\Gamma[x := 0]$) ou alors `DPLL`($\Gamma[x := 1]$))

Exemple 2.2.8

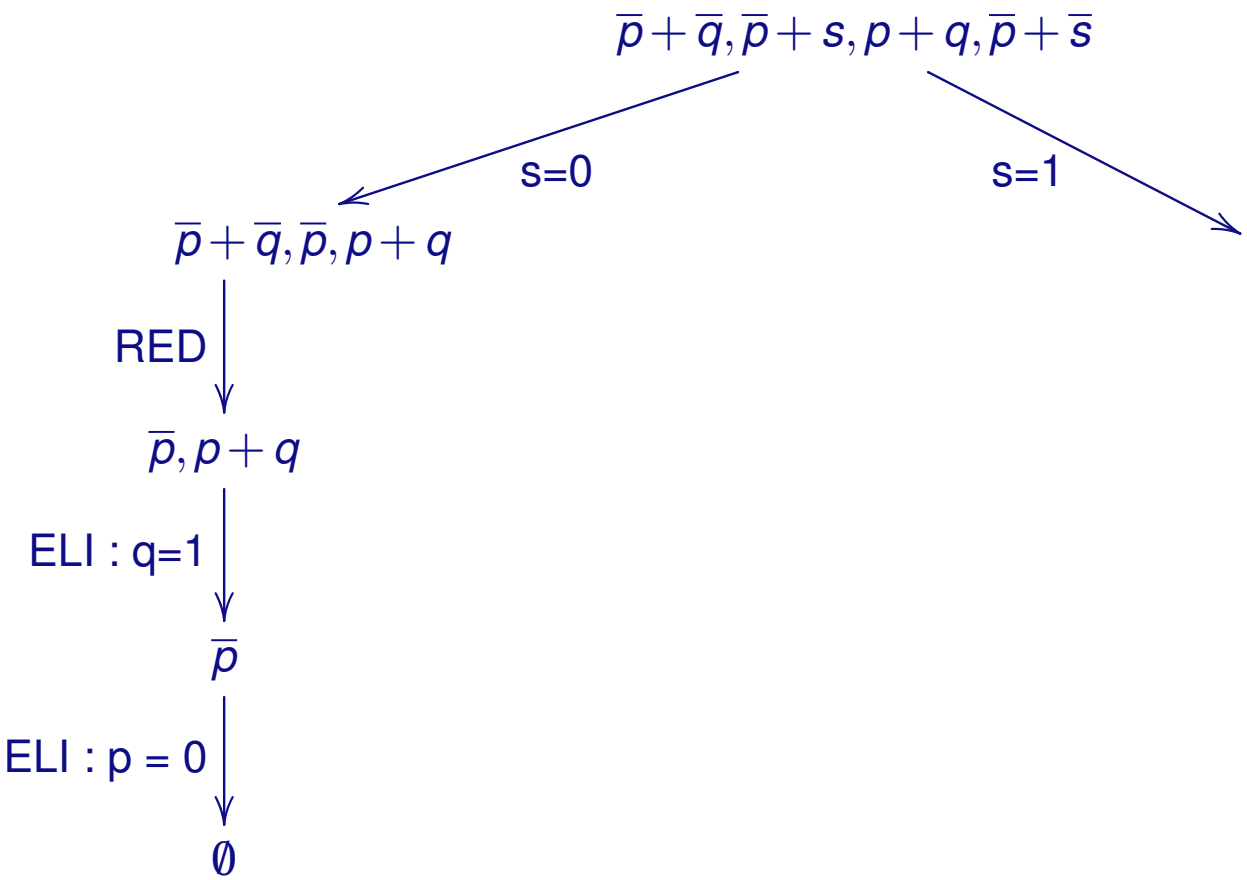
Soit Γ l'ensemble de clauses : $\bar{a}+\bar{b}, a+b, \bar{a}+\bar{c}, a+c, \bar{b}+\bar{c}, b+c$.



Puisque toutes les feuilles portent la clause vide, l'ensemble Γ est insatisfaisable.

Exemple 2.2.8

Soit Γ l'ensemble de clauses : $\overline{p}+\overline{q}, \overline{p}+s, p+q, \overline{p}+\overline{s}$.



Puisqu'une feuille porte l'ensemble vide, l'ensemble Γ est satisfaisable.
Il est inutile de poursuivre la construction de la branche droite.

Théorèmes 2.2.9 et 2.2.10

L'algorithme Algo_DPLL est correct et se termine.

Preuve de terminaison

- La suppression des clauses valides n'est exécutée qu'une seule fois.
- Boucle de simplifications : le nombre de clauses diminue strictement
- Appels récursifs : le nombre de variables diminue strictement

D'où la terminaison.

Preuve de correction

Il s'agit de démontrer que la satisfaisabilité est préservée tout au long de l'algorithme.

- Dans la boucle principale :
  cf lemme pour chaque simplification.
- Correction des appels récursifs :
  Rappel de la propriété 2.1.21 :
  Γ *a un modèle ssi* $\Gamma[x := 0]$ *ou* $\Gamma[x := 1]$ *est satisfaisable.*
  Donc si les appels récursifs sont corrects, l'appel courant l'est aussi.

Puisque l'algorithme est correct pour un ensemble Γ sans littéraux (cas d'arrêt), il est correct pour tout ensemble de clauses Γ.

Remarques 2.2.11 et 2.2.12

- **Oubli de simplifications :** DPLL reste correct si on oublie (une ou plusieurs fois) la réduction (2), l'élimination des littéraux isolés (3) et/ou la réduction unitaire (4).
- **Choix de la variable :**
  - Un bon choix pour la variable x de l'étape (5), consiste à choisir la variable qui apparait le plus souvent.
  - Un meilleur choix consiste à choisir la variable qui va entraîner par la suite le plus de simplifications.

Cf. section 2.2.5 pour les principales heuristiques de branchement.

Principe de l'algorithme : construire toutes les clauses déduites de Γ

Suivant la hauteur des arbres de preuves.

Algorithme

Pour tout entier i
Tant qu'il est possible de construire de nouvelles clauses
Construire l'ensemble réduit de toutes les clauses ayant une preuve de hauteur $\leq i$.

En pratique :
Maintenir deux suites d'ensembles de clauses, $\Delta_{i(i\geq 0)}$ et $\Theta_{i(i\geq 0)}$

Deux suites d'ensembles de clauses

Δ_i = les nouvelles clauses utiles

Clauses déduites par une preuve de hauteur i, après élimination :

- des clauses valides
- des clauses incluant une clause de preuve de hauteur $< i$.

Δ_0 est obtenu en réduisant Γ.

Θ_i = les anciennes clauses encore utiles

Clauses déduites par une preuve de hauteur $< i$ après élimination :

- des clauses valides
- des clauses incluant une autre clause de preuve de hauteur $\leq i$.

Θ_0 est l'ensemble vide.

Exemple ??

Soit $\Gamma = \{a+b+\bar{a}, a+b, a+b+c, a+\bar{b}, \bar{a}+b, \bar{a}+\bar{b}\}$

i	Δ_i	Θ_i	$\Delta_i \cup \Theta_i$	Résolvants de Δ_i et $\Delta_i \cup \Theta_i$
0	~~$a+b+\bar{a}$~~, $a+b$ ~~$a+b+c$~~, $a+\bar{b}$ $\bar{a}+b, \bar{a}+\bar{b}$	$\emptyset$	~~$a+b$~~, ~~$a+\bar{b}$~~, ~~$\bar{a}+b$~~, ~~$\bar{a}+\bar{b}$~~	a, b, ~~$b+\bar{b}$~~, ~~$a+\bar{a}$~~, $\bar{b}$, $\bar{a}$
1	$a, b, \bar{b}, \bar{a}$	$\emptyset$	$a, b, \bar{b}, \bar{a}$	$\bot$
2	$\bot$	$\emptyset$	$\bot$	$\emptyset$
3	$\emptyset$	$\bot$		

Rappel :

- $\Delta_{i+1} =$
  - Tous les résolvants de Δ_i et de $\Delta_i \cup \Theta_i$
  - Réduire
  - Enlever les résolvants qui incluent une clause de $\Delta_i \cup \Theta_i$
- $\Theta_{i+1} =$
  Enlever de $\Delta_i \cup \Theta_i$ les clauses qui incluent une clause de Δ_{i+1}

Aujourd'hui

- La résolution est un système de déduction correct et complet : il permet de **caractériser** toutes les formules insatisfaisables.
- L'**algorithme** DPLL utilise les idées de la résolution pour :
  - trouver un **modèle**
  - sinon, prouver l'**insatisfaisabilité** par une exploration efficace des assignations.
- La Stratégie Complète est un **algorithme** qui donne **toutes** les clauses déductibles d'un ensemble initial

La prochaine fois

- Déduction naturelle

À chercher : **Hypothèses** :

- (H1) : $p \Rightarrow \neg j \equiv \bar{p} + \bar{j}$
- (H2) : $\neg p \Rightarrow j \equiv p + j$
- (H3) : $j \Rightarrow m \equiv \bar{j} + m$
- ($\neg$ C) : $\neg m \wedge \neg p$ (deux clauses $\bar{m}$ et $\bar{p}$)

Construire la preuve de $H1, H2, H3, \neg C \vdash \bot$ obtenue par l'algorithme DPLL (ici, peu importe le choix de la variable)