Déduction automatique en logique propositionnelle classique

David Delahaye

Faculté des Sciences
David.Delahaye@lirmm.fr

Master Informatique M2 2022-2023

Méthode naïve

Tester toutes les assignations

- On donne toutes les valeurs possibles aux variables propositionnelles ;
- Les valeurs d'une variable propositionnelle sont $\top$ et $\bot$;
- La proposition est valide si elle donne $\top$ dans tous les cas ;
- La proposition est insatisfiable si elle donne $\bot$ dans tous les cas ;
- La proposition est non valide si elle donne $\bot$ dans certains cas ;
- La proposition est satisfiable si elle donne $\top$ dans certains cas.

Remarques

- Méthode naïve car exponentielle (donc inefficace) ;
- Pour n variables, on a 2^n cas à tester.

Exemple

Tester la validité d'une formule

- $A \wedge B \Rightarrow A$;
- On fait une table de vérité :

A	B	$A \wedge B$	$A \wedge B \Rightarrow A$
$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\bot$	$\top$
$\bot$	$\bot$	$\bot$	$\top$

- On peut faire un arbre aussi, puis annoter les noeuds/feuilles par les valeurs de vérité, c'est plus visuel.

Méthode de Davis-Putnam-Logemann-Loveland

Principe des méthodes clausales par réfutation

- On n'utilise pas la formule initiale en entrée ;
- On nie la formule (on prend sa négation) ;
- On la met sous forme clausale (ensemble de clauses) ;
- Une clause est disjonction de littéraux ;
- Un littéral est un axiome ou la négation d'un axiome ;
- Un axiome est une variable (propositionnelle) ;
- Puis on cherche si la forme clausale est insatisfiable ;
- Si elle l'est alors sa négation (la formule initiale) est valide.

Mise en forme clausale

Règles de transformation

$$\neg\neg F \rightarrow F \qquad \neg\top \rightarrow \bot \qquad \neg\bot \rightarrow \top$$

$$\neg(F_1 \wedge F_2) \rightarrow \neg F_1 \vee \neg F_2 \qquad \neg(F_1 \vee F_2) \rightarrow \neg F_1 \wedge \neg F_2$$

$$F_1 \Rightarrow F_2 \rightarrow \neg F_1 \vee F_2$$

$$F_1 \wedge \top \rightarrow F_1 \qquad \top \wedge F_1 \rightarrow F_1 \qquad F_1 \wedge \bot \rightarrow \bot \qquad \bot \wedge F_1 \rightarrow \bot$$

$$F_1 \vee \top \rightarrow \top \qquad \top \vee F_1 \rightarrow \top \qquad F_1 \vee \bot \rightarrow F_1 \qquad \bot \vee F_1 \rightarrow F_1$$

$$(F_1 \wedge F_2) \vee F_3 \rightarrow (F_1 \vee F_3) \wedge (F_2 \vee F_3)$$

$$F_3 \vee (F_1 \wedge F_2) \rightarrow (F_3 \vee F_1) \wedge (F_3 \vee F_2)$$

Mise en forme clausale

Exemple

- Proposition : $A \wedge B \Rightarrow A$.
- Étapes de clausification :
  $A \wedge B \Rightarrow A \rightarrow \neg(A \wedge B) \vee A \rightarrow \neg A \vee \neg B \vee A$
- L'ensemble de clauses est : $\{\neg A \vee \neg B \vee A\}$.

La règle de base de DPLL : le « splitting »

Principe du « splitting »

- On considère un ensemble de clauses S et une variable A ;
- On note $S[A := \top]$ l'ensemble obtenu en enlevant de S toutes les clauses contenant le littéral A et en effaçant le littéral $\neg A$ dans les clauses restantes ;
- On note $S[A := \bot]$ l'ensemble obtenu en enlevant de S toutes les clauses contenant le littéral $\neg A$ et en effaçant le littéral A dans les clauses restantes ;
- On notera que S est satisfiable ssi $S[A := \top]$ ou $S[A := \bot]$ l'est ;
- On notera que S est insatisfiable ssi $S[A := \top]$ et $S[A := \bot]$ le sont ;
- On peut « splitter » selon toutes les variables de S, mais on retombe exactement sur la méthode naïve, qui est exponentielle ;
- Le but de DPLL est donc d'éviter à tout prix le « splitting ».

La règle de base de DPLL : le « splitting »

Simplifications implicites

- On considère des simplifications implicites lorsqu'on réalise les opérations $S[A := \top]$ et $S[A := \bot]$;
- Si $\top$ appartient à une clause alors cette clause est éliminée de S (du coup, S peut devenir vide et il sera satisfiable) ;
- Si $\bot$ appartient à une clause alors on élimine $\bot$ de la clause (du coup, on peut avoir une clause vide $\Box$ et S sera alors insatisfiable).

Une règle de simplification : la résolution unitaire

Principe de la résolution unitaire

- Une clause est unitaire si elle contient un unique littéral A ou $\neg A$;
- Si $A \in S$ alors on peut remplacer S par $S[A := \top]$;
- En effet, si $A \in S$, S est satisfiable ssi $S[A := \top]$ l'est ;
- Si $\neg A \in S$ alors on peut remplacer S par $S[A := \bot]$;
- En effet, si $\neg A \in S$, S est satisfiable ssi $S[A := \bot]$ l'est.

Une règle de simplification : les clauses pures

Principe des clauses pures

- Un atome A est pur dans S ssi il apparaît toujours avec le même signe ;
- Autrement dit, pour un atome A, soit $A \notin S$, soit $\neg A \notin S$;
- On dira que A ou $\neg A$ est un littéral pur dans S ;
- Une clause contenant un littéral pur sera dite pure ;
- Si P est le sous-ensemble des clauses pures de S, alors on peut remplacer S par $S \backslash P$;
- En effet, S est satisfiable ssi $S \backslash P$ l'est.

Une règle de simplification : les clauses pures

Principe des clauses pures

- Un atome A est pur dans S ssi il apparaît toujours avec le même signe ;
- Autrement dit, pour un atome A, soit $A \not\in S$, soit $\neg A \not\in S$;
- On dira que A ou $\neg A$ est un littéral pur dans S ;
- Une clause contenant un littéral pur sera dite pure ;
- Si P est le sous-ensemble des clauses pures de S, alors on peut remplacer S par $S \backslash P$;
- En effet, S est satisfiable ssi $S \backslash P$ l'est.

Une dernière simplification : les tautologies

- Une tautologie est une clause contenant à la fois A et $\neg A$;
- Si T est le sous-ensemble des tautologies de S, alors on peut remplacer S par $S \backslash T$;
- En effet, S est satisfiable ssi $S \backslash T$ l'est.

Procédure de DPLL

Algorithme

DPLL(S)=

si $S = \emptyset$ alors retourner « satisfiable » ;

sinon si $\square \in S$ alors retourner « insatisfiable » ;

sinon si S contient une tautologie C alors retourner DPLL($S \backslash C$) ;

sinon si S contient une clause unitaire avec A (resp. $\neg A$) alors
retourner DPLL($S[A := \top]$) (resp. DPLL($S[A := \bot]$)) ;

sinon si A (resp. $\neg A$) est pur dans S alors
retourner DPLL($S[A := \top]$) (resp. DPLL($S[A := \bot]$)) ;

sinon choisir une variable A de S
et retourner DPLL($S[A := \top]$) ou DPLL($S[A := \bot]$).

Exécution

Exemple

- Démontrer la validité de la formule : $A \wedge B \Rightarrow A$;
- On nie la formule : $\neg(A \wedge B \Rightarrow A)$;
- On la met sous forme clausale :
  $\neg(A \wedge B \Rightarrow A) \rightarrow \neg(\neg(A \wedge B) \vee A) \rightarrow \neg\neg(A \wedge B) \wedge \neg A \rightarrow A \wedge B \wedge \neg A$
- $S = \{A, B, \neg A\}$;
- On applique DPLL :
  - On a une clause unitaire A, on calcule $S[A := \top]$:
    $S[A := \top] = \{\top, B, \neg\top\} = \{B, \bot\} = \{B, \Box\}$
    On appelle DPLL($S[A := \top]$)=DPLL($\{B, \Box\}$)
  - On a la clause vide $\Box$, on retourne « insatisfiable ».

Résolution

Principe de la méthode

- Méthode clausale par réfutation (comme DPLL) :
  - On nie la proposition initiale ;
  - On la met ensuite en forme clausale.
- Règle de résolution entre deux clauses :

$$\frac{C \vee A \qquad \neg A \vee C'}{C \vee C'}$$

- Les clauses au-dessus de la barre sont les prémisses ;
- La clause en dessous est le résolvant entre les clauses prémisses.

Procédure de résolution

Algorithme

$Sat := \emptyset$;
tant que $S \neq \emptyset$ faire
  choisir $C \in S$;
  $S := S \backslash \{C\}$;
  si $C = \square$ alors retourner « insatisfiable » ;
  si C est une tautologie alors ; (* passer à la clause suivante *)
  sinon, si $C \in Sat$ alors ; (* idem *)
  sinon pour tout résolvant C_1 entre C
  et une clause de $Sat \cup \{C\}$ faire
    $S := S \cup \{C_1\}$;
  $Sat := Sat \cup \{C\}$;
retourner « satisfiable ».

Exécution

Exemple

- Démontrer la validité de la formule : $A \wedge B \Rightarrow A$;
- $S = \{A, B, \neg A\}$;
- On applique la résolution :
  - $Sat = \emptyset$, $S = \{A, B, \neg A\}$;
  - On choisit la clause A : $Sat = \{A\}$, $S = \{B, \neg A\}$;
  - On choisit la clause B : $Sat = \{A, B\}$, $S = \{\neg A\}$;
  - On choisit la clause $\neg A$:
    - Résolution entre $\neg A$ et A : résolvant $\square$;
    - $Sat = \{A, B, \neg A\}$, $S = \{\square\}$;
  - On choisit la clause $\square$, on retourne « insatisfiable ».

Méthode des tableaux

Un peu d'histoire

- Méthode plus ancienne que la résolution ;
- Introduite par les pionniers Hintikka et Beth (années 50) ;
- Perfectionnée ensuite par Smullyan et Fitting ;
- À partir du calcul des séquents de Gentzen sans coupure.

Principe

- Par réfutation sur la proposition initiale et par cas.

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- α-règles (pas de branchement) ;
- β-règles (branchement).

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- α-règles (pas de branchement)

$$\frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B} \Rightarrow_{\text{right}}$$

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- α-règles (pas de branchement)

$$\frac{\Gamma \vdash \Delta, A \Rightarrow B}{\Gamma, A \vdash \Delta, B} \Rightarrow_{\text{right}}$$

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- α-règles (pas de branchement)

$$\frac{\vdash A \Rightarrow B}{A \vdash B} \Rightarrow_{\text{right}}$$

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- α-règles (pas de branchement)

$$\frac{\neg(A \Rightarrow B) \vdash \bot}{A, \neg B \vdash \bot} \Rightarrow_{\mathsf{right}}$$

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- α-règles (pas de branchement)

$$\frac{\neg(A \Rightarrow B)}{A, \neg B}\, \alpha_{\neg\Rightarrow}$$

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- β-règles (branchement)

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta} \Rightarrow_{\text{left}}$$

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- β-règles (branchement)

$$\frac{\Gamma, A \Rightarrow B \vdash \Delta}{\Gamma \vdash \Delta, A \qquad \Gamma, B \vdash \Delta} \Rightarrow_{\mathsf{left}}$$

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- β-règles (branchement)

$$\frac{A \Rightarrow B \vdash \bot}{\vdash A \qquad B \vdash \bot}\Rightarrow_{\mathsf{left}}$$

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- β-règles (branchement)

$$\frac{A \Rightarrow B \vdash \bot}{\neg A \vdash \bot \qquad B \vdash \bot}\Rightarrow_{\text{left}}$$

Méthode des tableaux

De LK0 aux règles de la méthode des tableaux

- β-règles (branchement)

$$\frac{A \Rightarrow B}{\neg A \mid B}\,\beta_{\Rightarrow}$$

Méthode des tableaux

Règles de clôture et règles analytiques

$$\frac{\bot}{\odot}\ \odot_{\bot} \qquad \frac{\neg\top}{\odot}\ \odot_{\neg\top} \qquad \frac{P \qquad \neg P}{\odot}\ \odot$$

$$\frac{\neg\neg P}{P}\ \alpha_{\neg\neg} \qquad \frac{P \Leftrightarrow Q}{\neg P, \neg Q \mid P, Q}\ \beta_{\Leftrightarrow} \qquad \frac{\neg(P \Leftrightarrow Q)}{\neg P, Q \mid P, \neg Q}\ \beta_{\neg\Leftrightarrow}$$

$$\frac{P \wedge Q}{P, Q}\ \alpha_{\wedge} \qquad \frac{\neg(P \vee Q)}{\neg P, \neg Q}\ \alpha_{\neg\vee} \qquad \frac{\neg(P \Rightarrow Q)}{P, \neg Q}\ \alpha_{\neg\Rightarrow}$$

$$\frac{P \vee Q}{P \mid Q}\ \beta_{\vee} \qquad \frac{\neg(P \wedge Q)}{\neg P \mid \neg Q}\ \beta_{\neg\wedge} \qquad \frac{P \Rightarrow Q}{\neg P \mid Q}\ \beta_{\Rightarrow}$$

Méthode des tableaux

Exemple

$$\dfrac{\dfrac{\dfrac{\neg(A \wedge B \Rightarrow A)}{A \wedge B, \neg A}\ \alpha_{\neg\Rightarrow}}{A, B, \neg A}\ \alpha_{\wedge}}{\odot}\ \odot$$