

# Déduction automatique en logique propositionnelle classique

David Delahaye

Faculté des Sciences  
[David.Delahaye@lirmm.fr](mailto:David.Delahaye@lirmm.fr)

Master M2 2019-2020

# Méthode naïve

## Tester toutes les assignations

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

$A$	$B$	$A \wedge B$	$A \wedge B \Rightarrow A$
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	T

- 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

$$\neg\neg F \rightarrow F \quad \neg\top \rightarrow \perp \quad \neg\perp \rightarrow \top$$

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

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

$$F_1 \wedge \top \rightarrow F_1 \quad \top \wedge F_1 \rightarrow F_1 \quad F_1 \wedge \perp \rightarrow \perp \quad \perp \wedge F_1 \rightarrow \perp$$

$$F_1 \vee \top \rightarrow \top \quad \top \vee F_1 \rightarrow \top \quad F_1 \vee \perp \rightarrow F_1 \quad \perp \vee F_1 \rightarrow F_1$$

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

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

# 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 :  $\{\neg A \vee \neg B \vee A\}$ .

# Exercice

Nier et mettre en forme clausale les propositions suivantes

- ❶  $A \Rightarrow B \Rightarrow A$
- ❷  $(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C$
- ❸  $A \wedge B \Rightarrow B$
- ❹  $B \Rightarrow A \vee B$
- ❺  $(A \vee B) \Rightarrow (A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow C$
- ❻  $A \Rightarrow \perp \Rightarrow \neg A$
- ❼  $\perp \Rightarrow A$

# 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  $S[A := \perp]$  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  $S[A := \perp]$  l'est ;
- On notera que  $S$  est insatisfiable ssi  $S[A := \top]$  et  $S[A := \perp]$  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  $S[A := \perp]$  ;
- Si  $\top$  appartient à une clause alors cette clause est éliminée de  $S$  (du coup,  $S$  peut devenir vide et il sera satisfiable) ;
- Si  $\perp$  appartient à une clause alors on élimine  $\perp$  de la clause (du coup, on peut avoir une clause vide  $\square$  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  $S[A := \perp]$  ;
- En effet, si  $\neg A \in S$ ,  $S$  est satisfiable ssi  $S[A := \perp]$  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  $S \setminus P$  ;
- En effet,  $S$  est satisfiable ssi  $S \setminus P$  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  $A \notin S$ , 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  $S \setminus P$  ;
- En effet,  $S$  est satisfiable ssi  $S \setminus P$  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  $S \setminus T$  ;
- En effet,  $S$  est satisfiable ssi  $S \setminus T$  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( $S \setminus C$ ) ;
- sinon si  $S$  contient une clause unitaire avec  $A$  (resp.  $\neg A$ ) alors  
retourner DPLL( $S[A := \top]$ ) (resp. DPLL( $S[A := \perp]$ )) ;
- sinon si  $A$  (resp.  $\neg A$ ) est pur dans  $S$  alors  
retourner DPLL( $S[A := \top]$ ) (resp. DPLL( $S[A := \perp]$ )) ;
- sinon choisir une variable  $A$  de  $S$   
et retourner DPLL( $S[A := \top]$ ) ou DPLL( $S[A := \perp]$ ).

## 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  $S[A := \top]$  :  
 $S[A := \top] = \{\top, B, \neg\top\} = \{B, \perp\} = \{B, \square\}$   
On appelle  $DPLL(S[A := \top]) = DPLL(\{B, \square\})$
  - ▶ On a la clause vide  $\square$ , on retourne « insatisfiable ».

# Exercice

Appliquer DPLL sur les propositions suivantes

- ❶  $A \Rightarrow B \Rightarrow A$
- ❷  $(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C$
- ❸  $A \wedge B \Rightarrow B$
- ❹  $B \Rightarrow A \vee B$
- ❺  $(A \vee B) \Rightarrow (A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow C$
- ❻  $A \Rightarrow \perp \Rightarrow \neg A$
- ❼  $\perp \Rightarrow A$

## 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 :

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

- 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  $C \in S$  ;  
   $S := S \setminus \{C\}$  ;  
  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 ».
```

## 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 ».

# Exercice

Appliquer la résolution sur les propositions suivantes

- ❶  $A \Rightarrow B \Rightarrow A$
- ❷  $(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C$
- ❸  $A \wedge B \Rightarrow B$
- ❹  $B \Rightarrow A \vee B$
- ❺  $(A \vee B) \Rightarrow (A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow C$
- ❻  $A \Rightarrow \perp \Rightarrow \neg A$
- ❼  $\perp \Rightarrow A$

# 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

- $\alpha$ -règles (pas de branchement) ;
- $\beta$ -règles (branchement).

# Méthode des tableaux

## De LK0 aux règles de la méthode des tableaux

- $\alpha$ -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

- $\alpha$ -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

- $\alpha$ -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

- $\alpha$ -règles (pas de branchement)

$$\frac{\neg(A \Rightarrow B) \vdash \perp}{A, \neg B \vdash \perp} \Rightarrow_{\text{right}}$$

# Méthode des tableaux

## De LK0 aux règles de la méthode des tableaux

- $\alpha$ -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

- $\beta$ -règles (branchement)

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

# Méthode des tableaux

## De LK0 aux règles de la méthode des tableaux

- $\beta$ -règles (branchement)

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

# Méthode des tableaux

## De LK0 aux règles de la méthode des tableaux

- $\beta$ -règles (branchement)

$$\frac{A \Rightarrow B \vdash \perp}{\vdash A \quad B \vdash \perp} \Rightarrow_{\text{left}}$$

# Méthode des tableaux

## De LK0 aux règles de la méthode des tableaux

- $\beta$ -règles (branchement)

$$\frac{A \Rightarrow B \vdash \perp}{\neg A \vdash \perp \quad B \vdash \perp} \Rightarrow_{\text{left}}$$

# Méthode des tableaux

## De LK0 aux règles de la méthode des tableaux

- $\beta$ -règles (branchement)

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

# Méthode des tableaux

## Règles de clôture et règles analytiques

$$\frac{\perp}{\odot} \odot \perp$$

$$\frac{\neg \top}{\odot} \odot \neg \top$$

$$\frac{P \quad \neg P}{\odot} \odot$$

$$\frac{\neg \neg P}{P} \alpha_{\neg \neg}$$

$$\frac{P \Leftrightarrow Q}{\neg P, \neg Q \mid P, Q} \beta_{\Leftrightarrow}$$

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

$$\frac{P \wedge Q}{P, Q} \alpha_{\wedge}$$

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

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

$$\frac{P \vee Q}{P \mid Q} \beta_{\vee}$$

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

$$\frac{P \Rightarrow Q}{\neg P \mid Q} \beta_{\Rightarrow}$$



# Méthode des tableaux

## Exemple

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

# Exercice

Appliquer la méthode des tableaux sur les propositions suivantes

- ❶  $A \Rightarrow B \Rightarrow A$
- ❷  $(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C$
- ❸  $A \wedge B \Rightarrow B$
- ❹  $B \Rightarrow A \vee B$
- ❺  $(A \vee B) \Rightarrow (A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow C$
- ❻  $A \Rightarrow \perp \Rightarrow \neg A$
- ❼  $\perp \Rightarrow A$

## Un club très sélectif

- Un club privé a les règles suivantes :
  - ① Tout membre non écossais porte des chaussettes rouges ;
  - ② Tout membre porte un kilt ou ne porte pas de chaussettes rouges ;
  - ③ Les membres mariés ne sortent pas le dimanche ;
  - ④ Un membre sort le dimanche si et seulement s'il est écossais ;
  - ⑤ Tout membre qui porte un kilt est écossais et marié ;
  - ⑥ Tout membre écossais porte un kilt.
- Démontrer qu'en réalité, personne ne peut être accepté !
- Pour ce faire, on encodera le problème comme un problème de logique propositionnelle et on utilisera MiniSat pour le résoudre.
- MiniSat est utilisable en ligne (voir Moodle).
- MiniSat prend des formules sous forme CNF uniquement.
- Le format d'entrée est le format DIMACS (voir Moodle).

## Une princesse ou un tigre ? (R. Smullyan)

- Un prisonnier doit choisir entre deux cellules dont l'une cache une princesse et l'autre un tigre.
- S'il choisit la princesse, il doit l'épouser, mais s'il tombe sur le tigre, il est dévoré.
- Toutes les combinaisons sont possibles : il peut y avoir deux tigres, deux princesses, ou un tigre et une princesse.
- La cellule ne peut pas être vide, et il ne peut pas y avoir deux tigres, ou deux princesses, ou un tigre et une princesse dans une même cellule.

## Une princesse ou un tigre ? (R. Smullyan)

- Il y a plusieurs épreuves, on va s'intéresser à la première épreuve.
- Il y a deux affiches collées sur les portes :
  - ❶ Il y a une princesse dans cette cellule et un tigre dans l'autre.
  - ❷ Il y a une princesse dans une cellule et il y a un tigre dans une cellule.
- Le roi dit (et il dit toujours la vérité) : « Une des affiches dit la vérité, et l'autre ment. ».
- Quelle cellule doit choisir le prisonnier ?

## Une princesse ou un tigre ? (R. Smullyan)

- Modéliser ce problème et sa solution en utilisant Coq, et démontrer (en faisant une preuve) que la solution proposée est correcte.
- Modéliser ce problème et sa solution, puis le donner à Limboole pour qu'il vérifie que la solution proposée est correcte.
- Limboole doit être téléchargé (voir Moodle).
- Limboole prend des formules quelconques (pas forcément sous forme CNF) en entrée (voir le format dans le README).
- Limboole utilise PicoSAT par défaut comme solveur SAT.

## Implantation de DPLL

- Planter DPLL dans le langage de votre choix ;
- Plusieurs étapes dans cette implantation :
  - ▶ Type de données pour les propositions (pas de « parser ») ;
  - ▶ Fonction de clausification ;
  - ▶ Fonction de substitution d'une variable par une valeur ( $\top/\perp$ ) ;
  - ▶ Fonction pour les tautologies ;
  - ▶ Fonction pour la résolution unitaire ;
  - ▶ Fonction pour les clauses pures ;
  - ▶ Algorithme lui-même.