Leçon 916 : Formule du calcul propositionnel : représentation, forme normale, satisfiabilité. Applications.

Julie Parreaux

2018 - 2019

Références pour la leçon

[2] Duparc, *La logique pas à pas.*
[1] Cori et Lascar, *La logique mathématique, tome 1.*
[3] Stern, *Fondements mathématiques de l'informatique.*

Développements de la leçon

Transformation de Tseitin

2-SAT est NL-complet (donc dans P)

Plan de la leçon

Motivation

Défense

La logique propositionnelle est une logique très simple permettant d'exprimer des contraintes : c'est la première logique que nous utilisons.

Ce qu'en dit le jury

Le jury attend des candidats qu'ils abordent les questions de la complexité de la satisfiabilité. Pour autant, les applications ne sauraient se réduire à la réduction de problèmes NP-complets à SAT.

Une partie significative du plan doit être consacrée à la représentation des formules et à leurs formes normales.

1 Le langage de la logique propositionnelle

On formalise la logique grâce à une syntaxe, une sémantique et un lien entre les deux donné par un système de preuve (sur la syntaxe). Nous allons également donner plusieurs représentations possibles (dans un ordinateur) de la syntaxe.

1.1 Syntaxe [2, p.68]

- *Définition* : langage avec connecteurs minimaux (induction) + exemples
- *Notation* : Autres connecteurs
- *Remarque* : Ambiguïté et parenthèses + exemples

1.2 Représentations

Comment on représente les mots de ce langage dans un ordinateur ? Laquelle est la plus efficace (sous quels critères) ?

- Représentation linéaire
- Représentation arborescente (fait apparaître l'induction) + *Théorème* de la lecture unique [1, p.57] (unicité de la représentation par l'arbre, représentation fortement non ambiguë)
- Représentation DAG (direct acyclique graph) : moins de mémoire (utilisé pour la satisfiabilité)
- Représentation BD
- Représentation circuit (donne de nouvelles classes de complexités)

1.3 Sémantique [2, p.83]

Comment donner du sens à ce qu'on écrit ? Que signifie ces formules ?

- Valuation sur un modèle
- *Application* : traduction du langage naturel
- *Problèmes* : validité et satisfiabilité (deux notions duales)
- Une représentation : table de vérité (c'est une méthode pratique, ordonnée, pour tester la validité d'une formule)
- *Remarque* : complexité de calcul (Conséquence de Cook)
- *Théorème* Cook
- *Application* : réduction de problème pour NP-dureté (problème de séparation des automates)
- *Définition* : tautologie / contradictions

1.4 Théories et preuves

Vérifier que nous n'avons pas fait n'importe quoi avec ces définitions...

— *Définition* : théorie
— *Théorème* : compacité
— *Applications* : coloriage de sous-graphe // pavage // logique du premier ordre
— *Définition* : CNF (on voit après comment la mettre sous forme normale) + résolution
— *Théorème* : correction et complétude Les autres logiques ?
— Taille des arbres et certificats.

2 Équivalence de formules

Afin de trouver des moyens de résoudre le problème de satisfiabilité, on peut chercher à mettre nos formules sous différentes forme qui sont équivalentes : elles expriment les mêmes contraintes. Pour cela nous avons besoins de la notions d'équivalence et de formes normales.

2.1 Équivalence sémantique et équisatisfiabilité [2, p.106]

Comparer deux formules c'est connaître les modèles qui les discrimine (satisfait une des formule mais pas l'autre). Lorsque nous ne pouvons pas les différentier, nous parlons d'équivalence sémantique (c'est l'équivalence que l'on souhaite).

— *Définition* : équivalence sémantique + *Exemples* : loi de Morgan
— *Proposition* : caractérisation avec la validité
— *Conséquence* : Complexité du problème VALIDITE
— *Définition* : équisatisfiabilité (l'équivalence sémantique est trop contraignante mais pour la satisfiabilité, celle-ci suffit)
— *Proposition* : équivalent implique équisatisfiable.

2.2 Systèmes de connecteurs

L'ensemble des connecteurs logiques, appelé système de connecteurs, que nous choisissons pour écrire nos formules peuvent exprimer plus ou moins de choses. On en veut le moins de symbole possibles pour décrire notre logique.

— *Définition* : systèmes complets et minimaux + *Exemples*
— *Proposition* : $\{\neg, \vee\}$, $\{\neg, \wedge\}$, $\{\mathsf{nand}\}$ sont des systèmes complets minimaux.
— *Remarque* : $\{\neg, \wedge, \vee\}$ est un système complet mais pas minimal.

2.3 Formes normales conjonctives ou disjonctives

Les formes normales sont une manière de représenté une formule par une autre formule dont les connecteurs logiques sont restreints et ordonnés selon un certain ordre. On présente leur mise en forme, leurs points forts et leurs limites

Forme normale négative Elle est à l'origine de toutes les autres

— *Définition* : Forme normale négative (par sa grammaire)
— *Proposition* : Transformation en une forme normale négative

Forme normale conjonctive

— *Proposition* : Existence d'une formule CNF équivalente + *Remarque* : transformation exponentielle ($\varphi_n = (a_1 \wedge b_1) \vee \cdots \vee (a_n \wedge b_n)$) *Remarque* : C'est une forme normale négative

— *Proposition* : Existence d'une formule CNF équisatisfiable de taille linéaire (Transformation de Tseitin) DEV

— *Corollaire* : Le problème CNF-SAT est NP-complet

— *Remarque* : Le problème CNF-VALIDE est dans P.

Forme normale disjonctive Les formules conjonctives (et uniquement conjonctives) permettent de représenter un modèle.

— *Définitions* : forme normale disjonctive DNF

— *Proposition* : Existence d'une formule DNF équivalente

— *Théorème* : Le problème DNF-SAT est dans P

— *Conséquence* : Par Cook et $P \neq NP$, transformation exponentielle

— *Exemple* : $\varphi_n = (a_1 \vee b_1) \wedge \cdots \wedge (a_n \vee b_n)$

— *Remarque* : Le problème DNF-VALIDE est dans NP-complet

3 Résoudre la satisfiabilité en pratique

On a vu que savoir si une formule est satisfiable est difficile. Cependant, en pratique on est souvent amené à résoudre cette question : on pose des contraintes (emplois du temps) ou issue de problème NP-complet. On aimerai alors un moyen de résoudre la satisfiabilité de manière efficace.

3.1 Considérer des fragments de la logique

Limiter la logique (si le problème le permet) peut nous donner des algorithmes qui résolvent la satisfiabilité en temps linaire.

— *Problème* : 3SAT -> NP-complet (pas assez réduit)

— *Problème* : 2SAT -> NL-dur (donc dans P) DEV Attention expressivité n'est pas la même

— *Application* : problème d'ordonnancement multiprocesseurs où les processeurs n'ont que deux créneaux pour les tâches

— *Problème* : Horn -> P

— *Application* [3] : Prolog avec les clauses de Horn

— *Application* : Analyse syntaxique : calcul de premier, LL(1), problèmes sur grammaires algébriques

3.2 Essayer de résoudre astucieusement

Si on provient d'un problème NP-complet, on peut mettre la formule sous forme CNF et résoudre à l'aide d'un algorithme dont le pire cas s'exécute en temps exponentiel mais dont on espère que celui-ci soit meilleur dans la pratique.

— DPLL / CDLL

— *Application* : SAT-solver (Un des moyen de contrer la NP-complétude)

Quelques notions importantes

Équivalence sémantique et équisatisfiabilité

Comparer deux formules c'est connaître les modèles qui les discrimine (satisfait une des formule mais pas l'autre). Lorsque nous ne pouvons pas les différentier, nous parlons d'équivalence sémantique (c'est l'équivalence que l'on souhaite). Cependant être sémantiquement équivalent est une contrainte forte pour notre but de résoudre la satisfiabilité. On va alors définir une deuxième notion ; l'équisatisfiablité.

Formules sémantiquement équivalentes [2, p.106] On rappelle qu'une formule est satisfiable s'il existe un modèle (une valuation dans notre cas) qui rend la formule vrai. On dit que la formule est satisfaite dans ce modèle. Une formule est dite valide si pour tout modèle la formule est rendue vrai.

Définition. Deux formules sont équivalentes sémantiquement si et seulement si elles sont satisfaites dans les même modèles, on note $\varphi \equiv \psi$.

Proposition. *$\varphi \equiv \psi$ si et seulement si $\varphi \Leftrightarrow \psi$ est valide.*

Idée de la démonstration. $\Rightarrow$ Soit un modèle, par leur équivalence, elles prennent les mêmes valeurs sur ce modèle d'où la validité.

$\Leftarrow$ Soit un modèle, comme $\varphi \Leftrightarrow \psi$ est valide, φ et ψ prennent les mêmes valeurs sur ce modèle.

□

Exemple. Les lois de De Morgan.

Remarque : La relation $\equiv$ est bien une relation d'équivalence (elle est réflexive, symétrique et transitive).

Formules équisatisfiables Souvent, on cherche une formule sémantiquement équivalente pour répondre au problème de la satisfiabilité. L'équivalence pour tous les modèles n'est donc pas nécessaire. Dans ce contexte, deux formules telles que si l'une est satisfiable alors l'autre l'est également suffit.

Définition. Deux formules φ et ψ sont dites équisatisfiable si et seulement si (φ est satisfiable si et seulement si ψ est satisfiable).

Autrement dit, deux formules sont dites équisatisfiable si l'une des deux admet un modèle qui la satisfait l'autre aussi. Cependant ce modèle peut être distinct (on ne demande pas que se soit le même modèle qui satisfont les deux formules).

Proposition. *Deux formules équivalentes sont équisatisfiables.*

Démonstration. Soit φ et ψ deux formules telles que $\varphi \equiv \psi$. Comme $\varphi \Leftrightarrow \psi$ est valide, φ est satisfiable si et seulement si ψ l'est aussi (elles le sont pour le même modèle). D'où l'équisatisfiabilité. □

Les formes normales

Les formes normales sont une manière de représenté une formule par une autre formule dont les connecteurs logiques sont restreints et ordonnés selon un certain ordre [2, p.166]. Ces deux formules peuvent être équivalentes mais sont généralement équisatisfiable. Cette condition suffit puisque la mise sous forme normale est très souvent effectuée pour répondre au problème de la satisfiabilité d'une formule.

Forme normale négative La forme normale négative est à la base des deux formes suivante puisque les formes normales conjonctives et disjonctives sont négatives. Les négations ne peuvent porter que sur des variables propositionnelles et non sur des sous-formules.

Définition. Une forme normale négative est une forme normale engendrée par le grammaire

$$\varphi ::= \bot \mid \top \mid \ p \mid \neg p \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi)$$

où p est une variable propositionnelle.

Proposition. *Toute formule admet une forme normale équivalente.*

Idée de la démonstration. On effectue une traduction syntaxique inductive (sur la formule à traduire) où seul le cas de la négation est réécrit.

- $tr(\bot) = \bot$
- $tr(\top) = \top$
- $tr(p) = p$
- $tr(\neg p) = \neg p$
- $tr(\varphi \vee \psi) = tr(\varphi) \vee tr(\phi)$
- $tr(\varphi \wedge \psi) = tr(\varphi) \wedge tr(\phi)$
- $tr(\neg(\varphi \vee \psi)) = tr(\neg\varphi) \wedge tr(\neg\phi)$
- $tr(\neg(\varphi \wedge \psi)) = tr(\neg\varphi) \vee tr(\neg\phi)$

□

Forme normale conjonctive Les formes normales conjonctives sont les formes normales les plus utilisée : leur satisfiabilité restent NP-complet mais la traduction peut se faire en temps linéaire. Nous avons donc une représentation plus simple pour les SAT-solveur qui n'explose pas à la transformation. C'est pour cette raison que nous l'utilisons souvent.

Définition. Un littéral est une variable propositionnelle ou sa négation.

Définition. Une forme normale conjonctive est une formule de la forme $\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m} l_{i,j} \right)$ où les $l_{i,j}$ sont des littéraux.

Remarque : Une forme normale conjonctive est une forme normale négative (la réciproque est fausse).

Définition. Une disjonction de littéraux est appelée une clause. Cela explique le fait que nous parlons de forme clausale : on a une autre représentation.

Proposition. *Pour toute formule ψ, il existe une formule φ sous forme normale conjonctive telle que ψ et φ soient équivalentes.*

Idées de la démonstration. **Preuve existentielle**
- ψ n'admet pas de modèle où elle est fausse (c'est une tautologie) : elle est équivalente à $\top$.
- ψ admet un modèle où elle est fausse : on prend alors la négation de la formule que l'on transforme en DNF (c'est possible car sur ces modèles la négation est vraie et ce sont les seuls modèles). On remet la négation à la formule obtenue.

Preuve constructive On met la formule sous forme normale négative puis application de ces règles de distributivité.

□

Remarque : Cette transformation peut exploser.

Théorème. *Le problème de validité (*CNF-SAT*) est dans NP.*

Idée de la démonstration. On fait une réduction à SAT à l'aide de la transformation de Tseintin qui est linéaire. □

Théorème. *Le problème de validité (*CNF-SAT*) est dans P.*

Idée de la démonstration. Une forme normale conjonctive est valide si et seulement si pour toute clause, il existe une variable telle que cette variable et sa négation soient dans la clause. Cette propriété implique un algorithme linéaire testant la validité.

⇒ Raisonnons par l'absurde et supposons qu'il existe une clause ne contenant pas la négation d'aucun de ces littéraux. Alors le modèle qui rend faux chacun de ces littéraux ne satisfait pas la formule (la clause n'est pas satisfaite car elle ne contient aucune négation des littéraux que nous avons mis à faux). Contradiction avec la validité de la formule

⇐ Soit une telle formule et soit un modèle pour φ. Comme dans chacune des clauses il existe une variable de cette clause telle que sa négation apparaissent alors la clause est vraie (car la négation ou la variable est vraie).

□

Forme normale disjonctive Les formules conjonctives (et uniquement conjonctives) permettent de représenter un modèle. En effet, elle est vrai si et seulement si les valeurs de vérités du modèles sont transcrite dans les littéraux de la formule. Un forme normale disjonctive est donc satisfiable si et seulement si une de ces clauses décrit un modèle.

Définition. Une forme normale disjonctive est une formule de la forme $\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m} l_{i,j} \right)$ où les $l_{i,j}$ sont des littéraux.

Remarque : Une forme normale disjonctive est une forme normale négative (la réciproque est fausse).

Proposition. *Pour toute formule ψ, il existe une formule φ sous forme normale disjonctive telle que ψ et φ soient équivalentes.*

Idées de la démonstration. **Preuve existentielle**
— ψ n'admet pas de modèle : elle est équivalente à n'importe quelle contradiction : $x \wedge \neg x$
— ψ admet un modèle : alors pour chacun de ces modèles on construit la formule conjonctive qui décrit sa valuation et on regroupe dans une disjonction la totalité de ces modèles.

Preuve constructive On procède de la même manière que pour les formes normales conjonctives en inversant les règles de distributivité (mise sous forme normale négative puis application de ces règles de distributivité). On obtient les même effet que pour la forme normale conjonctive : on a une explosion de la taille de la formule.

□

Théorème. *Le problème de satsifiabilité (*DNF-SAT*) est dans P.*

Idée de la preuve. Pour chacune des clause (tant qu'on n'a pas réussi à en rendre une vraie) : réinitialiser la valuation et affecter la valeur vrai à chacun des littéraux. □

Conséquence : Par le théorème de Cook et si $P \neq NP$, on sait qu'il n'existe pas de transformation polynomiale d'une formule en une formule sous forme normale disjonctive équisatisfiable (ou équivalente).

Contre-exemple. Soit $(\varphi_n)_{n \in \mathbb{N}}$ une famille de formule du calcul propositionnel telle que sa mise sous forme disjonctive est exponentielle (même pour l'équisatisfiabilité). On pose, pour tout $n \in \mathbb{N}$, $\varphi_n = (a_1 \vee b_1) \wedge \cdots \wedge (a_n \vee b_n)$ qui a $2n$ variables et $2n - 1$ connecteurs. La forme normale disjonctive de φ_n s'écrit $\bigvee_{x_i \in \{a_i, b_i\}} (x_1 \wedge \cdots \wedge x_n)$ avec $2n$ disjonctions.

Théorème. *Le problème de validité (*DNF-SAT*) est NP-complet.*

Remarque : La dualité entre les eux formes normales pour les problèmes de validité et de satisfiabilité provient de la dualité entre ces deux problèmes et du fait que la négation d'une forme normale conjonctive donne une forme normale disjonctive (sous forme normale négative) et réciproquement.

Systèmes de connecteurs

L'ensemble des connecteurs logiques, appelé système de connecteurs, que nous choisissons pour écrire nos formules peuvent exprimer plus ou moins de choses [2, p.175]. Lorsqu'ils permettent d'exprimer toute la logique propositionnelle, on parle de système de connecteurs

complets. La recherche de tels systèmes nous permet de nous assurer de la cohérence de la définition de notre logique (on exprime tous ce qu'on veut exprimer). On peut également chercher à exprimer la logique avec le moins de connecteurs possibles : on cherche donc des systèmes de connecteurs complets et minimaux.

Cette notion de système de connecteurs complets et minimaux existe dans toutes les logiques (modale, premier ordre, ...) mais ne sont pas présenté sous cette forme.

Définition. Un système de connecteur est dit complet si toute formule est équivalente à une formule qui ne contient que ces connecteurs là. Si, de plus, tous sous-ensembles de connecteurs issus de ce système ne forme un système de connecteur complet, on parlet de système complet minimal.

Proposition. 1. *$\{\neg, \vee\}$ est un système complet minimal.*

2. *$\{\neg, \wedge\}$ est un système complet minimal.*
3. *$\{\neg, \wedge, \vee\}$ est un système complet mais pas minimal.*
4. *$\{\mathsf{nand}\}$ est un système complet minimal où P nand $Q \equiv \neg(P \wedge Q)$.*

Idée de la démonstration.

1. Le système $\{\neg, \vee\}$ est un système complet par les règles de réécriture $a \wedge b \equiv \neg(\neg a \vee \neg b)$, $a \Rightarrow b \equiv \neg a \vee b$ et $a \Leftrightarrow b \equiv \neg(\neg(\neg a \vee b) \vee \neg(\neg b \vee a))$ remplaçant les autres connecteurs. Montrons qu'il est minimal.
   - $\emptyset$ Dans ce cas, on n'a que les variables propositionnelles. Alors, on ne peut trouver une formule qui ne soit qu'une variable équivalente à $p \vee q$.
   - $\neg$ On se ramène au cas précédent en remarquant que toutes les formules sont de cette forme : $r = \underbrace{\neg \dots \neg}_{n \text{ fois}} x$. Donc r est une variable positive si n est paire et sa négation sinon. On ne peut trouver une formule r équivalente à $p \vee q$.
   - $\vee$ On veut une infinité de formule non équivalente deux à deux. On commence par chercher une formule équivalente à $\neg P$ ce qui est impossible (raisonnement par l'absurde).
2. On ne fait que $\{\neg, \vee\}$, $\{\neg, \vee\}$ est analogue ($\vee$ est le dual de $\wedge$).
3. Le système $\{\neg, \vee, \wedge\}$ est un système complet par les lois de Morgan et non minimal car $\{\neg, \vee\} \subset \{\neg, \vee, \wedge\}$ est minimal.
4. Le système $\{\mathsf{nand}\}$ est minimal (pour la même raison que le cas vide précédent). Montrons qu'il est complet. Pour cela, on va montrer que toute formule exprimer dans le système $\{\neg, \vee\}$ est équivalente à une de notre système. On raisonne donc par induction sur la hauteur de notre formule.
   - $h = 0$ Dans ce cas $\varphi = p$ avec $p \in VAR$ et $\varphi \equiv (p \mathsf{nand} p) \mathsf{nand} (p \mathsf{nand} p)$.
   - $h > 0$ On distingue deux cas :
     - $\varphi = \neg\psi$ L'hypothèse d'induction nous donne ψ' et on pose $\varphi \equiv (\psi' \mathsf{nand} \psi')$.
     - $\varphi = \psi_1 \vee \psi_2$ L'hypothèse d'induction nous donne ψ_1' et ψ_2' et on pose $\varphi \equiv (\psi_1' \mathsf{nand} \psi_2') \mathsf{nand} (\psi_1' \mathsf{nand} \psi_2')$.

□

Remarque : Le système de connecteur que nous avons défini suffit pour décrire la logique propositionnelle car il est complet.

Problèmes de décisions en logique

Nous allons maintenant présenter trois problème de décision légitime lorsque nous étudions une logique : le problème de la satisfiabilité, de la validité et du model checking. Nous rappelons également leur complexité (si elle existe).

Définition. Une formule (close) φ est satisfiable s'il existe un modèle $\mathcal{M}$ tel que $\mathcal{M}, v \models \varphi$.

Définition. Une formule (close) φ est valide (ou tautologique) si pour tout modèle $\mathcal{M}$, $\mathcal{M}, v \models \varphi$.

Définition. On définit le problème SAT sur les formules d'une logique.

Problème : SAT
entrée : ϕ une formule (close)
sortie : Oui si ϕ est satisfiable ; non sinon

Définition. On définit le problème VALIDE sur les formules d'une logique.

Problème : VALIDE
entrée : ϕ une formule (close)
sortie : Oui si ϕ est valide ; non sinon

Définition. On définit le problème MODEL-CHECKING sur les formules d'une logique.

Problème : MODEL-CHECKING
entrée : ϕ une formule (close), $\mathcal{M}$ un modèle fini
sortie : Oui si $\mathcal{M} \models \phi$; non sinon

Présentons quelques résultats sur ces problèmes de décisions.

Problème	Propositionnelle	FO
SAT	NP-complet (Cook)	Indécidable
VALIDE	NP-complet	Indécidable
MODEL-CHECKING	P	PSPACE-complet

Références

[1] R. Cori and D. Lascard. *Logique mathématique, tome 1*. Masson, 1994.

[2] J. Duparc. *La logique pas à pas*. Presse polytechnicienne et université romandes, 2015.

[3] J. Stern. *Fondements mathématiques de l'informatique*. Ediscience internationnal, 1990.