Préparation Agrégation : Logique
Calcul Propositionnel

Christine Paulin
http://www.lri.fr/~paulin/Agreg

6 octobre 2015

Résumé

En lien avec la leçon : 916-Formules du calcul propositionnel : représentation, formes normales, satisfiabilité. Applications.

Programme
Calcul propositionnel : syntaxe et sémantique. Tables de vérité, tautologies, formes normales, forme clausale. Théorème de complétude du calcul propositionnel.

Table des matières

Références

[Carton, 2008] Carton, O. (2008). *Langages Formels Calculabilité et Complexité*. Vuibert.

[Cori and Lascar, 1993] Cori, R. and Lascar, D. (1993). *Logique Mathématique*. Axiomes. Masson.

[David et al., 2004] David, R., Nour, K., and Raffalli, C. (2004). *Introduction à la Logique. Théorie de la démonstration (2ème édition)*. Sciences Sup. Dunod.

[Devismes et al., 2012] Devismes, S., Lafourcade, P., and Lévy, M. (2012). *Informatique théorique : Logique et démonstration automati que, Introduction à la logique propositionnelle et à la logique du premier o rdre*. Ellipses.

[Goubault-Larrecq and Mackie, 1997] Goubault-Larrecq, J. and Mackie, I. (1997). *Proof Theory and Automated Deduction*, volume 6 of *Applied Logic Series*. Kluwer Academic Publishers.

1 Syntaxe

Définition 1.1 (Ensemble de formules propositionnelles) *noté* PROP*, plus petit ensemble qui contient*
- *ensemble des variables propositionnelles :* $\mathcal{X}$
- $\top, \bot$
- $\neg P$ *si* $P \in$ PROP
- $P \wedge Q, P \vee Q, P \Rightarrow Q$ *si* $P, Q \in$ PROP

On considère ici les formules comme des termes "abstraits". La *signature* associée est $\top, \bot, \neg, \wedge, \vee, \Rightarrow$ avec $\top, \bot$ des constantes (arité 0), $\neg$ un symbole d'arité 1 et $\wedge, \vee, \Rightarrow$, des symboles d'arité 2.

Un terme peut se représenter par exemple par un arbre dont les nœuds sont étiquetés par les connecteurs logiques $\top, \bot, \neg, \wedge, \vee, \Rightarrow$, avec un nombre de fils qui correspond à l'arité et dont les feuilles sont étiquetées par les variables propositionnelles.

Si on représente une formule de manière linéaire comme un mot, il faut ajouter des parenthèses pour éviter les ambiguités.

Règles de précédence : la négation a une précédence plus forte que celle de la conjonction, elle-même plus forte que la disjonction, elle-même plus forte que l'implication. L'implication associe à droite : la formule $P \Rightarrow Q \Rightarrow R$ se lie donc comme $P \Rightarrow (Q \Rightarrow R)$.

Remarque 1 *Dans le livre [Goubault-Larrecq and Mackie, 1997] les formules propositionnelles sont définies sans utiliser les constantes* $\top$ *et* $\bot$*. Si nécessaire, les formules* $x \vee \neg x$ *et* $x \wedge \neg x$ *peuvent être utilisées.*

Définitions récursives sur l'ensemble des formules

L'ensemble des propositions est l'algèbre dite *initiale* associée à la signature. Cela signifie qui si on se donne un ensemble A quelconque, des éléments $h^A_\top, h^A_\bot \in A$, des fonctions $h^A_X \in \mathcal{X} \to A$, $h^A_\neg \in A \to A$ et $h^A_\wedge, h^A_\vee, h^A_\Rightarrow \in A \times A \to A$, alors il existe une unique fonction $f \in \text{PROP} \to A$ telle que :

$$\begin{array}{ll} f(\top) = h^A_\top \quad f(\bot) = h^A_\bot & \\ f(x) = h^A_X(x) & (x \in \mathcal{X}) \\ f(\neg P) = h^A_\neg(f(P)) & (P \in \text{PROP}) \\ f(P \circ Q) = h^A_\circ(f(P), f(Q)) \ (\circ \in \{\vee, \wedge, \Rightarrow\} & P, Q \in \text{PROP}) \end{array}$$

Exercice 1.1 (Définitions itératives sur les formules) *Donner les définitions itératives des fonctions suivantes :*
- *Taille d'une formule* P *(nombre de symboles logiques, notée* $|P|$*)*
- *Ensemble des variables d'une formule* P *(notée* $\mathsf{var}(P)$*)*

Définition 1.2 (Substitution) *Une* substitution *est une application de* $\mathcal{X}$ *dans* PROP *qui est l'identité sauf sur un nombre fini de variables (le domaine). On notera* $Q_1/x_1, \ldots, Q_n/x_n$ *la substitution* ρ *telle que* $\forall i \in 1 \ldots n, \rho(x_i) = Q_i$ *et* $\rho(y) = y$ *si* $y \notin \{x_1, \ldots, x_n\}$*.*

Exercice 1.2 *Définir* $P[\sigma]$ *le résultat de la substitution* σ *appliquée à la formule* P*.*

Ce schéma est dit itératif. Il permet de dériver un schéma dit primitif récursif dans lequel les fonctions $h_\neg$, $h_\circ$, h_X peuvent également accéder aux sous-termes P et Q de la formule. Ce qui correspond au schéma suivant.

$$\begin{array}{ll} f(\top) = h^A_\top \quad f(\bot) = h^A_\bot & \\ f(x) = h^A_X(x) & (x \in \mathcal{X}) \\ f(\neg P) = h^A_\neg(P, f(P)) & (P \in \text{PROP}) \\ f(P \circ Q) = h^A_\circ(P, Q, f(P), f(Q)) \ (\circ \in \{\vee, \wedge, \Rightarrow\} & P, Q \in \text{PROP}) \end{array}$$

Exemple 1 (Définition récursive sur les formules) *Définir une fonction* `neg` *qui transforme une formule* P *en une formule équivalent à* $\neg P$ *en utilisant les lois de de Morgan (c'est-à-dire en transformant la négation d'une conjonction en disjonction des négations, etc.).*

Preuve par récurrence sur une formule. A la définition de l'ensemble des formules est associé une notion de preuve par récurrence structurelle sur la formule.

Proposition 1.1 (Récurrence structurelle sur les formules) *Soit $\Phi(P)$ une propriété sur les formules $P \in$ PROP. Si les conditions suivantes sont vérifiées*

— $\forall x \in \mathcal{X}, \Phi(x)$

— $\Phi(\top)$ *et* $\Phi(\bot)$

— *pour toute formule $P \in$ PROP, si $\Phi(P)$ alors $\Phi(\neg P)$*

— *pour toutes formules $P, Q \in$ PROP et tout connecteur $\circ \in \{\vee, \wedge, \Rightarrow\}$, si $\Phi(P)$ et $\Phi(Q)$ alors $\Phi(P \circ Q)$*

alors $\forall P \in$ PROP, $\Phi(P)$

Exercice 1.3 *Montrer par récurrence structurelle que pour deux substitutions σ et τ, et pour toute formule P, si $\forall x \in \mathsf{var}(P), \sigma(x) = \tau(x)$ alors $P[\sigma] = P[\tau]$.*

2 Sémantique

Définition 2.1 (Interprétation) *Une interprétation est une application de l'ensemble des variables propositionnelles dans l'ensemble des booléens $\mathbb{B} = \{F, V\}$ (on pourra aussi utiliser $0, 1$ comme valeurs booléennes).*

Une interprétation partielle est une fonction partielle de l'ensemble des variables propositionnelles dans l'ensemble des booléens $\mathbb{B}$.

Définition 2.2 (Valeur d'une formule) $[\![P]\!]_\rho$

$$[\![x]\!]_\rho = \rho(x)\ (x \in \mathcal{X}) \quad [\![\top]\!]_\rho = V \quad [\![\bot]\!]_\rho = F$$
$$[\![\neg P]\!]_\rho = \textbf{\textit{si}}\ [\![P]\!]_\rho = V\ \textbf{\textit{alors}}\ F\ \textbf{\textit{sinon}}\ V$$
$$[\![P \wedge Q]\!]_\rho = \textbf{\textit{si}}\ [\![P]\!]_\rho = V\ \textbf{\textit{alors}}\ [\![Q]\!]_\rho\ \textbf{\textit{sinon}}\ F$$
$$[\![P \vee Q]\!]_\rho = \textbf{\textit{si}}\ [\![P]\!]_\rho = V\ \textbf{\textit{alors}}\ V\ \textbf{\textit{sinon}}\ [\![Q]\!]_\rho$$
$$[\![P \Rightarrow Q]\!]_\rho = \textbf{\textit{si}}\ [\![P]\!]_\rho = V\ \textbf{\textit{alors}}\ [\![Q]\!]_\rho\ \textbf{\textit{sinon}}\ V$$

Définition 2.3 (Validité)

— *un modèle d'une formule est une interprétation ρ qui rend vraie la formule (on dit aussi que l'interprétation satisfait la formule)*

— *une formule est satisfiable si elle admet un modèle, un ensemble $\mathcal{E}$ de formules est satisfiable s'il existe une interprétation qui rend vraies toutes les formules de $\mathcal{E}$. Dans le cas contraire une formule ou un ensemble de formules est dit insatisfiable*

— *une formule est valide si elle est vraie dans toutes les interprétations, on dit aussi que c'est une tautologie*

— *une formule P est conséquence logique d'un ensemble $\mathcal{E}$ de formules (noté $\mathcal{E} \models P$) si toute interprétation qui satisfait les formules de $\mathcal{E}$ satisfait aussi la formule P.*

— *deux formules P et Q sont équivalentes (noté $P \equiv Q$) si chacune est conséquence logique de l'autre ce qui signifie aussi que ces deux formules sont satisfaites par les mêmes interprétations.*

Table de vérité. La table de vérité d'une formule consiste en un tableau dont chaque ligne correspond à une interprétation des variables propositionnelles pour laquelle on calcule la valeur de vérité de la formule.

Exemple :

p	q	$p \Rightarrow q$	$(p \Rightarrow q) \Rightarrow p$	$((p \Rightarrow q) \Rightarrow p) \Rightarrow p$
V	V	V	V	V
V	F	F	V	V
F	V	V	F	V
F	F	V	F	V

Proposition 2.1 *La valeur d'une proposition P ne dépend que de la valeur de l'interprétation sur l'ensemble $\mathsf{var}(P)$ des variables qui apparaissent dans la formule.*

Preuve: *Triviale par récurrence sur la structure de la formule.* □

Proposition 2.2 (Sémantique et substitution) *Soit deux formules P et Q, x une variable propositionnelle, b un booléen et ρ une interprétation. On note $\rho + (x \mapsto b)$ l'interprétation ρ' telle que $\rho'(x) = b$ et $\rho'(y) = \rho(y)$ pour tout $x \neq y$.*

On a $[\![P[Q/x]]\!]_\rho = [\![P]\!]_{\rho+(x\mapsto[\![Q]\!]_\rho)}$

Preuve: *Récurrence structurelle sur la formule P.* □

Exercice 2.1 *Déduire du résultat précédent la relation entre la validité de P et celle de $P[Q/x]$. Faire la même chose pour l'insatisfiabilité et la satisfiabilité.*

A une interprétation ρ sur un ensemble fini de variables $(x_i)_{i\in 1\dots n}$ on peut faire correspondre une formule dont ρ est un modèle, il suffit de prendre la formule $p_1 \wedge \dots \wedge p_n$ avec $p_i = x_i$ si $\rho(x_i) = V$ et $p_i = \neg x_i$ si $\rho(x_i) = F$.

Arbre sémantique. L'ensemble des interprétations sur un ensemble fini ou dénombrable de variables peut se représenter comme un arbre binaire dit *arbre sémantique* (fini ou infini).

On rappelle qu'un arbre est un graphe dont l'un des sommets est appelé racine et tel que pour tout sommet, il existe un unique chemin de la racine à ce sommet. Tout sommet de l'arbre qui n'est pas la racine admet un unique prédecessur que l'on appelle le père du sommet. Dans le cas des arbres, on utilise aussi la terminologie *nœud* comme synonyme de sommet dont le degré sortant est non nul et de *feuille* pour un sommet de degré sortant nul.

On considère un arbre binaire complet. L'ensemble des sommets de l'arbre est l'ensemble des mots finis sur l'alphabet $\{F, V\}$ de longueur inférieure au nombre de variables propositionnelles. On a une arête entre le sommet m et les sommets mF et mV. La racine correspond au mot vide.

On se donne une énumération des variables $x_1, \dots, x_n, \dots$. Chaque sommet correspondant au mot $a_1 \dots a_p$ correspond à une interprétation partielle $x_i \mapsto a_i$.

Dans le cas d'un nombre infini de variables, une branche infinie correspond à une interprétation.

Exercice 2.2 *Soit n variables propositionnelles, combien de formules différentes faut-il considérer pour en avoir 2 qui sont équivalentes ?*

Développement 1 (Compacité) *[Goubault-Larrecq and Mackie, 1997, thm 2.25 p32] Si un ensemble $\mathcal{E}$ est insatisfiable alors il existe un sous-ensemble fini de formules de $\mathcal{E}$ qui est insatisfiable.*

Exercice 2.3 (Preuve du théorème de compacité)
- *Montrer le résultat par un argument élémentaire dans le cas où l'ensemble des variables propositionnelles utilisées dans les formules est fini.*
- *On suppose qu'il y a un nombre dénombrable de variables propositionnelles.*
  - *montrer que sur chaque branche infinie de l'arbre sémantique, il existe un sommet correspondant à une interprétation partielle ρ et une formule de $P \in \mathcal{E}$ telle que $[\![P]\!]_\rho = F$*
  - *On appelle A l'ensemble des sommets qui correspondent à une interprétation ρ pour laquelle il existe une formule $P \in \mathcal{E}$ telle que $[\![P]\!]_\rho = F$. On considère le graphe G correspondant à la restriction de l'arbre sémantique aux sommets qui ne sont pas dans A ainsi que aux sommets qui sont dans A mais dont le père (s'il existe) n'est pas dans A.*
    - *montrer que G est un arbre*
    - *montrer que G est fini*
    - *montrer que les feuilles de G sont dans A*
    - *en déduire la propriété demandée*
- *On se place maintenant dans le cadre général d'un nombre quelconques de variables $\mathcal{X}$ L'ensemble $\mathcal{X} \to \mathbb{B}$ des interprétations est muni de la topologie produit. On prend sur $\mathbb{B}$ la topologie discrète ($\emptyset, \{F\}, \{V\}, \mathbb{B}$ sont des ouverts), l'espace est séparé (deux points différents sont couverts par deux ouverts disjoints) et trivialement compact (de tout recouvrement par des ouverts on peut extraire un*

recouvrement fini). La topologie produit est formée d'une union quelconque d'ouverts élémentaires de la forme $x \mapsto O_x$ avec $O_x = \mathbb{B}$ sauf sur un nombre fini de valeurs de x.
Le théorème de Tychonoff qui sera admis et dont la preuve requiert l'utilisation de l'axiome du choix établit que pour la topologie produit, $\mathcal{X} \to \mathbb{B}$ est également compact.
- *Montrer que pour toute formule P, l'ensemble $\phi(P)$ des interprétations qui rendent faux la formule P est à la fois ouvert et fermé.*
- *Montrer que $(\mathcal{X} \to \mathbb{B}) \subseteq \bigcup_{P \in \mathcal{E}} \phi(P)$*
- *En déduire le théorème de compacité*

— *Donner la formulation de la contraposée du théorème de compacité*
— *Déduire de la compacité que pour tout ensemble $\mathcal{E}$ et toute formule P, si $\mathcal{E} \models P$ alors il existe un sous ensemble fini Γ de $\mathcal{E}$ tel que $\Gamma \models P$.*

Modélisation en calcul propositionnel

Exercice 2.4 (Coloriage de graphe) *Le problème de coloriage de graphe consiste à se donner un ensemble fini de couleurs et à associer à chaque sommet d'un graphe une couleur de manière à ce que deux sommets adjacents n'aient pas la même couleur.*

Pour chaque couleur $c \in C$ on se donne un ensemble de variables propositionnelles (x_i^c) avec (x_i^c) qui sera vrai si le sommet i a la couleur c.
- *Proposer un ensemble de formules tel que toute interprétation qui rend vraies ces formules correspond à un coloriage du graphe.*
- *Soit un graphe infini, montrer que si tous les sous-graphes finis sont coloriables alors le graphe complet est coloriable*

3 Représentations

3.1 Ensembles de connecteurs

Si on s'intéresse aux formules modulo équivalence alors on peut restreindre le nombre de connecteurs

$$\top \equiv x \vee \neg x \qquad \bot \equiv x \wedge \neg x \qquad \neg p \equiv p \Rightarrow \bot$$
$$p \vee q \equiv \neg(\neg p \wedge \neg q) \quad p \wedge q \equiv \neg(\neg p \vee \neg q) \quad p \Rightarrow q \equiv (\neg p \vee q) \equiv \neg(p \wedge \neg q)$$

Traiter le cas des formules avec juste les variables propositionnelles, la conjonction (ou la disjonction) et la négation suffit à couvrir tous les cas. Cette remarque peut se révéler utile pour limiter le nombre de cas à traiter dans les démonstrations syntaxiques tout en gardant la généralité.

Définition 3.1 (Forme normale de négation) *une formule est en* forme normale de négation *si elle n'utilise que les connecteurs $\vee$, $\wedge$ (on élimine les implications $\Rightarrow$, $\top$ et $\bot$) et des symboles de négation uniquement sur des variables propositionnelles.*

Exercice 3.1 *Montrer que toute formule P est équivalente à une formule en forme normale de négation. Comparer les tailles des deux formules.*

Exercice 3.2 *Montrer que si on se restreint à des formules avec juste les connecteurs $\vee$ $\wedge$ alors la fonction booléeenne correspondant est croissante pour un ordre que l'on explicitera. En déduire qu'il n'est pas possible de représenter toutes les formules avec juste ces connecteurs.*

Ensembles complets de constructeurs. On peut se poser la question de quel sous-ensemble de connecteurs est suffisant. Voici quelque réponses
— Algèbre de Boole : $\vee, \wedge, \neg, 0, 1$

— Anneaux de Boole $\oplus, \wedge, 0, 1$ avec $\oplus$: ou exclusif
On a

$$p \oplus q \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$$

$$\neg p \equiv 1 \oplus p \quad p \vee q \equiv p \oplus q \oplus (p \wedge q)$$

— On peut aussi utiliser uniquement $\Rightarrow$ et $\bot$ (logique minimale), on introduit alors $\neg p \equiv p \Rightarrow \bot, p \vee q \equiv \neg p \Rightarrow q$ qui permet de tout reconstruire.
— On peut tout représenter avec un unique connecteur binaire la barre de Sheffer $(x|y) \equiv \neg x \vee \neg y$. On part de $x|x$ qui est équivalent à $\neg x$, on a alors $\neg x|\neg y \equiv x \vee y$ et on peut tout reconstruire.
— Un autre connecteur intéressant est un connecteur à trois arguments représentant la conditionnelle ainsi que les formules $\top$ et $\bot$

$$\texttt{if}(p, q, r) \equiv (p \wedge q) \vee (\neg p \wedge r) \equiv (p \Rightarrow q) \wedge (\neg p \Rightarrow r)$$

$$\neg p \equiv \texttt{if}(p, \bot, \top) \quad p \wedge q \equiv \texttt{if}(p, q, \bot)$$

On peut même se ramener à des formules dans lesquelles le connecteur `if` ne porte que sur des variables propositionnelles (ce qui revient à avoir une famille de connecteurs binaires $(\texttt{if}_x)_{x \in \mathcal{X}}$), voir ci-dessous la représentation sous-forme d'arbre binaires de décision.

3.2 Formes normales

Il peut être utile d'avoir des formes "canoniques" pour les formules.

Définition 3.2 (Littéral, clause) — *Un littéral est une variable propositionnelle ou sa négation*
— *Une clause est une disjonction de littéraux, on peut la représenter comme un ensemble de littéraux*
— *La clause vide correspond à une disjonction de zéro littéraux qui correspond à la formule $\bot$*

Définition 3.3 (Forme clausale) *Une formule est en* forme normale conjonctive *(FNC ou CNF en anglais) si elle s'écrit comme une conjonction de disjonction de littéraux.*

La forme clausale *d'une formule est un ensemble de clauses tel que la formule est équivalente à la conjonction des ces clauses.*
— *un ensemble vide de clauses représente la formule vraie*
— *on peut se limiter à des clauses où chaque variable apparaît au plus une fois (en effet une clause dans laquelle x et $\neg x$ apparaissent est équivalente à la formule vraie et donc peut disparaître de la forme clausale*

$$\top \equiv \emptyset \quad \bot \equiv \{\emptyset\}$$

Proposition 3.1 *Toute formule est équivalente à une formule en FNC.*

Preuve: *On peut réécrire la formule en utilisant les équivalences et la distributivité*

$$\neg\neg p \longrightarrow p \quad \neg(p \vee q) \longrightarrow \neg p \wedge \neg q \quad \neg(p \wedge q) \longrightarrow \neg p \vee \neg q \quad (p \wedge q) \vee r \longrightarrow (p \vee r) \wedge (q \vee r)$$

□

Exercice 3.3 (Mise en forme clausale équivalente)
Définir de manière mutuellement récursive deux fonctions $\texttt{clpos}(F)$ et $\texttt{clneg}(F)$ qui donnent des ensembles de clauses équivalentes à F et $\neg F$. On utilisera l'opération de combinaison $\texttt{sh}$ avec $\texttt{sh}(S, T) = \{C \vee C' | C \in S, C' \in T\}$.

Donner la forme normale conjonctive de la formule $\bigvee_{i=1\ldots n}(a_i \wedge b_i)$ que constatez vous ?

Si on a la table de vérité f d'une formule, il est facile de construire la forme normale conjonctive associée, chaque ligne de la formule peut se voir comme un ensemble de littéraux. La formule est vraie si elle ne se trouve sur aucune ligne où elle est fausse.

$$\bigwedge_{l_1,\ldots,l_n, f(l_1,\ldots,l_n)=F} \bar{l_1} \vee \ldots \vee \bar{l_n}$$

Il peut y avoir plusieurs formes normales conjonctives pour la même formule

$$p \equiv (p \vee q) \wedge (p \vee \neg q)$$

Définition 3.4 (Propositions equi-satisfiables) *Deux propositions P et Q sont équisatisfiables si la formule P est satisfiable si et seulement si Q est satisfiable.*

Deux formules équivalentes sont équisatisfiables mais le contraire est faux (prendre par exemple $\top$ et une variable x)

Exercice 3.4 (Mise en forme clausale équisatisfiable) *Si on s'intéresse uniquement à la satisfiabilité d'une formule, on peut trouver une forme equisatisfiable qui est linéaire en la taille de la formule initiale. Cela se fait en ajoutant des variables correspondant à des sous-termes de la formule initiale.*

- *montrer que $A \vee (P \wedge Q)$ est vraie pour I si et seulement si $\{A \vee x, \neg x \vee P, \neg x \vee Q\}$ est vrai pour une interprétation $I + \{x \mapsto b\}$ avec x une variable qui n'apparaît pas dans la formule initiale.*
- *en déduire une méthode qui calcule pour toute formule P un ensemble de clauses dont la taille est linéaire en la taille de P et qui est satisfiable exactement lorsque P est satisfiable.*
- *montrer que $A \vee B \vee C \vee D$ est vraie pour I si et seulement si $\{A \vee B \vee x, \neg x \vee C \vee D\}$ est vrai pour une interprétation $I + \{x \mapsto b\}$ avec x une variable qui n'apparaît pas dans la formule initiale. En déduire que la satisfiabilité d'une formule peut toujours se ramener à la satisfiabilité d'un ensemble de clauses ayant au plus trois littéraux. Estimer le nombre de clauses résultant de la transformation.*

Remarques.

- On a de même la notion de forme normale disjonctive (disjonction de conjonctions de littéraux)
- Certains problèmes peuvent se modéliser directement sous forme clausale (exemple le Sudoku).

3.3 Diagrammes de décision binaires

Si on écrit les formules avec le constructeur `if` alors on peut se restreindre au cas de `if` dont la condition est une variable propositionnelle et les feuilles l'une des constantes $\top$ ou $\bot$.

$$\texttt{if}(\top, A, B) \equiv A \qquad \texttt{if}(\bot, A, B) \equiv B \qquad \texttt{if}(\texttt{if}(A, B, C), P, Q) \equiv \texttt{if}(A, \texttt{if}(B, P, Q), \texttt{if}(C, P, Q))$$

On remarque que la transformation n'est pas linéaire.

L'expression normalisée correspond à un arbre binaire dont les noeuds internes sont étiquetés par les variables propositionnelles, et dont les feuilles sont étiquetées $\top$ et $\bot$.

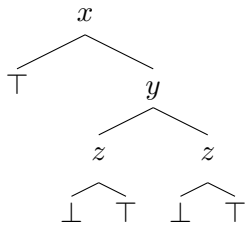
On va restreindre encore plus en demandant que les variables apparaissent au plus une fois une fois le long de chaque branche dans un ordre fixé a priori.

Pour construire l'arbre récursivement on peut utiliser la propriété suivante qui permet d'éliminer les variables les unes après les autres.

$$P \equiv \texttt{if}(x, P[\top/x], P[\bot/x])$$

Une formule sans variable propositionnelle peut ensuite se simplifier en $\bot$ ou $\top$.

Exemple 2 $((x \vee y \vee z) \wedge (y \Rightarrow (x \vee z))) \Rightarrow x$

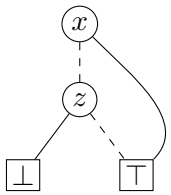


Exercice 3.5 *Construire l'arbre pour la formule :* $((x \Rightarrow y) \wedge z \Rightarrow x) \vee \neg z$

Définition 3.5 (OBDD) *Un diagramme de décision binaire ordonné est un graphe fini dirigé acyclique (DAG) dont les sommets sont étiquetés par les variables propositionnelles et les valeurs* $\bot, \top$ *et dont les arètes sont étiquetées par les valeurs* V, F.

- *les sommets étiquetés par des variables propositionnelles ont un degré sortant* 2*, une des arètes est étiquetée par la valeur* V *et l'autre par la valeur* F*, les sommets sont étiquetés par les formules* $\top, \bot$ *et ont un degré sortant nul. Cela permet de faire correspondre à tout sommet du graphe un arbre binaire de décision, c'est-à-dire une formule logique.*
- *les variables apparaissant le long d'un chemin sont ordonnées de manière croissante.*
- *aucun sommet n'a deux fils correspondant au même arbre binaire de décision*
- *deux sommets correspondant à un même arbre binaire de décision sont égaux*

Exemple 3 *On représente l'arète étiquetée* V *en trait plein et celle étiquetée* F *en pointillés. On repart de la formule de l'exemple 2 et on simplifie l'arbre obtenu.*



Construction d'un OBDD Afin de garantir le partage, on introduit une table t qui représente le graphe. On a deux sommets spéciaux pour les feuilles $\top$ et $\bot$. La table contient pour tous les autre sommets s du graphe une information comportant la variable x qui étiquette le sommet s et les sommets a et b des deux successeurs de s pour l'étiquette V et pour l'étiquette F. On note cette information $\texttt{if}(x, a, b)$ et on considère t comme une fonction des sommets du graphe différents de $\top$ et $\bot$ vers une information de la forme $\texttt{if}(x, a, b)$.

Un obdd est donné par la table t est un sommet s. On note $\texttt{form}(t, s)$ la formule logique associée au sommet s du graphe. On a

$$\texttt{form}(t, \bot) = \bot \quad \texttt{form}(t, \top) = \top$$
$$\texttt{form}(t, s) = \texttt{if}(x, \texttt{form}(t, a), \texttt{form}(t, b)) \quad \text{si } s \neq \top, \bot \quad t(s) = \texttt{if}(x, a, b)$$

Exercice 3.6 *Justifier pourquoi ces équations définissent bien une fonction.*

Opération de base de construction d'un OBDD. On se donne une opération de base pour enrichir un obdd t. Etant donné une variable x et deux sommets a et b tels que le sommet a représente la formule A et le sommet b représente la formule B, on cherche à représenter la formule $\texttt{if}(x, A, B)$. On notera le résultat de l'opération $\texttt{mkif}(x, a, b)$. Le résultat sera un obdd t' et un sommet s tel que $\texttt{form}(t', s) \equiv \texttt{if}(x, A, B)$ et $\texttt{form}(t', s') = \texttt{form}(t, s')$ pour tous les sommets s' de t différents de s.

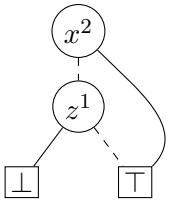
- si $a = b$ alors on ne change pas l'obdd et on renvoie a ;
- si le sommet existe déjà dans la table, c'est-à-dire qu'il existe un sommet s tel que $t(s) = \texttt{if}(x, a, b)$ alors on ne change pas l'obdd et on renvoie s ;

— dans les autres cas, on va ajouter à t un nouveau sommet s et une liaison $s \mapsto \texttt{if}(x, a, b)$. On renvoie la nouvelle table et le sommet s.

On remarque que cette opération ne préserve pas la condition d'ordre sur les variables, il ne faut donc l'utiliser que lorsque x est plus petit que les variables qui apparaissent dans A et B.

Exemple 4

$1 \mapsto if(z, \bot, \top)$
$2 \mapsto if(x, \top, 1)$



Opérations sur les OBDD Il est facile de faire opérer une fonction de transformation de formules qui respecte l'équivallence lorsque la formule est représentée à l'aide du connecteur `if`, en effet on aura $f(\texttt{if}(x, A, B)) \equiv \texttt{if}(x, f(A), f(B))$.

Il faut par contre utiliser la construction intelligente `mkif`

Exercice 3.7 — *Définir la fonction de négation d'une formule représentée comme un obdd.*
— *Définir la conjonction de deux formules représentées comme des obdds.*
— *Que pensez-vous de la complexité de ces opérations ? comment faire pour l'améliorer ?*

Propriétés des OBDD

Développement 2 (Canonicité de la représentation sous forme de OBDD) *[Goubault-Larrecq and Mackie, 1997, thm 2.48, p59] On fixe un ordre sur les variables, la représentation de deux formules équivalentes correspond syntaxiquement au même arbre, c'est-à-dire :*

$$(form(t, a) \equiv form(u, b)) \Leftrightarrow (form(t, a) = form(u, b)) \Leftrightarrow a = b$$

Application :
— Comment reconnait-on qu'une formule représentée par un obdd est valide ?
— Comment reconnait-on qu'une formule représentée par un obdd est insatisfiable ?
— Donner une méthode pour compter le nombre de modèles d'une formule.

Exercice 3.8 *[Goubault-Larrecq and Mackie, 1997, ex. 2.27] On se donne n variables propositionnelles $x_1, \ldots, x_n$ et on construit par récurrence sur n une formule Ψ_n.*

$$\Psi_1 \stackrel{\text{def}}{=} x_1 \qquad \Psi_{k+1} \stackrel{\text{def}}{=} x_{k+1} \Leftrightarrow \Psi_k$$

— *Construire l'OBDD correspondant à cette formule en prenant comme ordre $x_n < x_{n-1} < \ldots < x_1$.*
— *Donner le nombre de sommets et le nombre de chemins dans ce graphe.*

3.4 Complexité

Définition 3.6 (Problème NP) *Un problème est NP difficile s'il peut être résolu en temps polynomial en la taille des entrées sur une machine de Turing non déterministe.*

Développement 3 (Théorème de Cook-Levin) *[Carton, 2008, thm 4.19, p189] Le problème de savoir si une formule F est satisfiable est un problème NP.*

Le problème (SAT) de savoir si un ensemble de clauses C est satisfiable est un problème NP-complet : il est NP et tout problème NP se ramène en temps polynomial à un problème SAT.

C'est encore vrai si on se limite à des clauses qui ont au plus 3 littéraux.

4 Systèmes de preuve

4.1 Système de déduction

Les règles d'inférence sont une manière de *définir un ensemble* R (ou de manière équivalente un prédicat $R(x)$) qui est très utile en informatique théorique.

Une règle d'inférence se présente sous la forme de fraction :

$$\frac{P_1 \dots P_k}{R(t)}$$

Le dénominateur (aussi appelé la *conclusion* de la règle) est forcément un cas particulier de la relation que l'on cherche à définir. Le numérateur contient un nombre quelconque de conditions (que l'on appelera aussi les *prémisses*). Si ces conditions sont vérifiées alors la règle permet d'établir $R(t)$. S'il n'y a aucune condition $k = 0$, cela veut dire que $R(t)$ est toujours vrai.

Par exemple on peut définir le fait qu'un nombre entier est pair avec les règles d'inférence suivantes :

$$\frac{}{\texttt{pair}(0)} \qquad \frac{}{\texttt{pair}(2)} \qquad \frac{\texttt{pair}(x) \quad \texttt{pair}(y)}{\texttt{pair}(x+y)}$$

Dans la deuxième règle, x et y sont des variables qui sont implicitement quantifiées universellement, c'est-à-dire que la règle est de fait un schéma de règle.

Il y a plusieurs ensembles d'entiers qui vérifient ces propriétés (par exemple l'ensemble des entiers $\mathbb{N}$). Les règles d'inférence définissent un ensemble qui est *le plus petit* qui convient. Il suffit de prendre l'intersection de tous les ensembles qui vérifient les règles d'inférence. Le fait que cet ensemble vérifie également les règles d'inférence est un théorème (théorème de point-fixe de Tarski) qui nécessite des conditions supplémentaires sur P_i (monotonie en R). Nous nous intéresserons au cas simple dans lequel chaque P_i est soit un cas particulier $R(u)$ de la définition en cours, soit une condition qui ne parle pas de R.

Dérivation. Lorsque l'on introduit un système d'inférence pour définir une propriété, alors on utilise ces mêmes règles d'inférence pour *justifier* que la propriété est vraie pour des cas particuliers.

Les dérivations peuvent se présenter de manière linéaire comme une suite S d'instances $R(t)$. Pour ajouter une nouvelle instance il faut identifier une instance de règle dont les prémisses qui parlent de R sont dans S et telle que les autres prémisses indépendantes de R sont vérifiées, on peut alors ajouter la conclusion de la règle à la suite S. On se donne parfois un ensemble de faits initial S_0 que l'on cherche ensuite à étendre en appliquant les règles d'inférence. On dira alors que l'on a une dérivation sous les hypothèses S_0.

On peut ainsi dériver

$$\begin{array}{c} \texttt{pair}(0) \\ \texttt{pair}(2) \\ \texttt{pair}(4) \\ \texttt{pair}(8) \end{array}$$

On peut aussi dériver $\texttt{pair}(9)$ sous l'hypothèse $\texttt{pair}(3)$.

On préfère en général une représentation sous forme d'arbre dont chaque nœud est étiqueté par une instance $R(t)$ et si un nœud étiqueté par la formule $R(t)$ a des fils étiquetés par les formules $R(u_1) \dots R(u_n)$ alors $\dfrac{R(u_1) \dots R(u_n)}{R(t)}$ est une instance de règle.

$$\dfrac{\dfrac{\vdots}{\texttt{pair}(4)} \qquad \dfrac{\dfrac{}{\texttt{pair}(2)} \quad \dfrac{}{\texttt{pair}(2)}}{\texttt{pair}(4)}}{\texttt{pair}(8)}$$

Induction. On peut évidemment raisonner par récurrence sur la longueur de la dérivation. On utilise en général un principe plus direct. Le fait que l'ensemble défini est le *plus petit* qui satisfait les règles d'inférence, nous permet de dériver un schéma de preuve par induction spécialisé qui dit que si on se donne un autre ensemble S qui vérifie les mêmes propriétés que R alors pour tout x, si $R(x)$ on a aussi $S(x)$.

Dans le cas de la définition de `pair`, cela se traduit par le principe suivant :

Proposition 4.1 (Principe d'induction sur les entiers pairs.)

— *soit $\phi(n)$ une propriété sur les entiers que l'on cherche à montrer pour tous les entiers qui vérifient $\texttt{pair}(n)$*

— *si on peut démontrer les trois propositions suivantes :*

1. *$\phi(0)$ est vérifié ;*
2. *$\phi(2)$ est vérifié ;*
3. *pour tous n, m arbitraires tels que $\phi(n)$ et $\phi(m)$ sont vérifiés, on peut établir $\phi(n+m)$;*

— *alors on peut en déduire que pour tout n tel que $\texttt{pair}(n)$ on a $\phi(n)$.*

Extensions

— On peut définir plusieurs prédicats de manière mutuellement récursive. L'ensemble de faits déduits contient alors des instances de chacune des prédicats (peut se ramener au cas de base en ajoutant un paramètre supplémentaire à la relation qui permet d'identifier la relation)

4.2 Système à la Hilbert

Ces systèmes sont organisés à partir d'un ensemble de formules $\mathcal{A}$ appelés axiomes (règles sans prémisses) et des règles de déduction de la forme $\dfrac{P_1 \dots P_n}{P}$.

On écrit juste une formule P (par abus de notation) mais on est en train de définir la relation *P est prouvable*.

Un exemple de règle de déduction est modus-ponens (MP) : $\dfrac{P \quad P \Rightarrow Q}{Q}$

Si on prend un fragment de la logique avec juste $\Rightarrow$ et $\bot$ (avec $\neg P \stackrel{\text{def}}{=} P \Rightarrow \bot$) il suffit de prendre comme axiomes

$$P \Rightarrow Q \Rightarrow P \qquad (P \Rightarrow Q \Rightarrow R) \Rightarrow (P \Rightarrow Q) \Rightarrow (P \Rightarrow R) \qquad \neg\neg P \Rightarrow P$$

et la règle de déduction MP. Une alternative à l'axiome $\neg\neg P \Rightarrow P$ est $(\neg Q \Rightarrow \neg P) \Rightarrow P \Rightarrow Q$

Faire une preuve de $P \Rightarrow P$ est très indirect.

$$\dfrac{\dfrac{(P \Rightarrow (P \Rightarrow P) \Rightarrow P) \Rightarrow (P \Rightarrow P \Rightarrow P) \Rightarrow (P \Rightarrow P) \qquad P \Rightarrow (P \Rightarrow P) \Rightarrow P}{(P \Rightarrow P \Rightarrow P) \Rightarrow (P \Rightarrow P)} \qquad P \Rightarrow P \Rightarrow P}{P \Rightarrow P}$$

Proposition 4.2 (Théorème de déduction) *Une formule B est pouvable à partir d'un ensemble de formules Γ, A si et seulement la formule $A \Rightarrow B$ est pouvable à partir de l'ensemble de formules Γ*

Preuve: *Si on peut déduire $A \Rightarrow B$ à partir de Γ on peut aussi le faire à partir de Γ, A (affaiblissement) et en utilisant MP on en déduit B.*

Dans l'autre sens, il faut faire une récurrence sur la dérivation qui permet de dériver B à partir de Γ, A. □

4.3 Déduction naturelle

Présentation avec des arbres. La déduction naturelle peut se présenter avec des arbres de preuve dont chaque sommet est étiqueté par une formule et qui comporte en plus un mécanisme pour rendre certaines feuilles passives.

$$\frac{\begin{array}{c}[P]\\ \vdots\\ Q\end{array}}{P \Rightarrow Q} \qquad \frac{P \Rightarrow Q \qquad P}{Q} \qquad \frac{\begin{array}{c}[\neg P]\\ \vdots\\ \bot\end{array}}{P}$$

Présentation avec des séquents Si on veut rester dans le cadre standard des systèmes de déduction, on peut utiliser plutôt une notion de *séquent* $\Gamma \vdash A$ avec Γ un ensemble (ou multi-ensemble) fini de formules représentant les hypothèses actives.

Définition 4.1 (Séquent valide) *Un séquent $\Gamma \vdash A$ est valide si $\Gamma \models A$.*

On écrit alors les règles précédentes sous la forme

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q} \qquad \frac{\Gamma \vdash P \Rightarrow Q \qquad \Gamma \vdash P}{\Gamma \vdash Q}$$

Définition 4.2 (Déduction naturelle, NK0) *Les règles se décomposent pour chaque connecteur entre des règles d'introduction pour prouver une formule avec ce connecteur et des règles d'élimination qui expliquent comment utiliser une preuve sur une formule avec ce connecteur.*

	Introduction(I)	*Élimination*(E)
	Hypothèse $\dfrac{A \in \Gamma}{\Gamma \vdash A}$	
$\top$	$\dfrac{}{\Gamma \vdash \top}$	
$\bot$		$\dfrac{\Gamma, \neg C \vdash \bot}{\Gamma \vdash C}$
$\neg$	$\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}$	$\dfrac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash C}$
$\wedge$	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \quad \dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$
$\vee$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$	$\dfrac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$
$\Rightarrow$	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$	$\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$

Exemple 5 (Preuve de $A \vee \neg A$)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A \vee \neg A), A \vdash \neg(A \vee \neg A)}\,hyp \qquad \dfrac{\dfrac{}{\neg(A \vee \neg A), A \vdash A}\,hyp}{\neg(A \vee \neg A), A \vdash A \vee \neg A}\,\vee I}{\neg A \vee \neg A, A \vdash \bot}\,\neg E}{\neg(A \vee \neg A) \vdash \neg A}\,\neg I \qquad \dfrac{}{\neg(A \vee \neg A) \vdash \neg(A \vee \neg A)}\,hyp}{\neg(A \vee \neg A) \vdash A \vee \neg A}\,\vee I}{\neg(A \vee \neg A) \vdash \bot}\,\neg E}{\vdash A \vee \neg A}\,\bot E$$

Exercice 4.1 (Preuve de $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$)

Proposition 4.3 (Affaiblissement) *Si* $\Gamma \vdash A$ *et* $\Gamma \subseteq \Delta$ *alors* $\Delta \vdash A$

Preuve: *Par récurrence sur la structure de la dérivation sachant que l'affaiblissement est trivial au niveau de la règle hypothèse et se propage ensuite.* □

Développement 4 (Correction et complétude) *[Goubault-Larrecq and Mackie, 1997, thm 2.26, p33]*

Un séquent $\Gamma \vdash A$ *est valide si et seulement si il est prouvable en déduction naturelle.*

Preuve: *Pour montrer qu'un séquent prouvable est valide, il suffit de montrer que chaque règle préserve la validité ce qui est trivial.*

L'autre sens est plus intéressant.

- *Pour une interprétation* ρ*, on note* A^ρ *la formule qui est* A *si la valeur de* A *est vraie dans l'interprétation* ρ *et* $\neg A$ *sinon. Montrer que si* $\mathsf{var}(P) = \{x_1, \ldots, x_n\}$ *alors* $x_1^\rho, \ldots, x_n^\rho \vdash P^\rho$ *est dérivable en NK0.*
- *Montrer que si* $\Gamma, P \vdash Q$ *et* $\Gamma, \neg P \vdash Q$ *sont dérivables dans NK0, alors il en est de même pour* $\Gamma \vdash Q$*.*
- *En déduire que si* P *est une formule valide alors* $\vdash P$ *est dérivable en déduction naturelle.*
- *En déduire la complétude dans le cas où* Γ *est fini. Que peut-on dire dans le cas où* Γ *est infini ?*

□

4.4 Calcul des séquents

On remarque que si chaque règle de la déduction naturelle est valide, par contre beaucoup de ces règles ne sont pas inversibles, c'est-à-dire que le séquent de conclusion peut être valide sans que les prémisses le soient. Le calcul des séquents à conclusion multiple permet d'éviter ce problème en retardant les choix à faire.

Définition 4.3 (Calcul des séquents, LK0) *Un séquent (à conclusion multiple) s'écrit* $\Gamma \vdash \Delta$ *avec des ensembles finis de formule* Γ *et* Δ*. Il est satisfait par une interprétation* ρ *s'il existe une formule* A *de* Γ *telle* ρ *ne valide pas* A *ou une formule* A *de* Δ *telle que* ρ *valide* A*, donc si (dans le cas fini)* ρ *valide* $\bigwedge \Gamma \Rightarrow \bigvee \Delta$

hypothèse	$(\textsc{Hyp})\dfrac{}{A, \Gamma \vdash \Delta, A}$	
	gauche	*droite*
$\bot$	$\dfrac{}{\bot, \Gamma \vdash \Delta}$	$\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \bot}$
$\top$	$\dfrac{\Gamma \vdash \Delta}{\top, \Gamma \vdash \Delta}$	$(\textsc{Triv})\dfrac{}{\Gamma \vdash \Delta, \top}$
$\neg$	$\dfrac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta}$	$\dfrac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}$
$\wedge$	$\dfrac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}$	$\dfrac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}$
$\vee$	$\dfrac{A, \Gamma \vdash \Delta \quad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta}$	$\dfrac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B}$
$\Rightarrow$	$\dfrac{\Gamma \vdash \Delta, A \quad B, \Gamma \vdash \Delta}{A \Rightarrow B, \Gamma \vdash \Delta}$	$\dfrac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B}$

Exercice 4.2 *Faire une preuve de* $A \vee \neg A$ *de* $((A \Rightarrow B) \Rightarrow A) \Rightarrow A)$

Remarque *La preuve se fait de manière "directe" malgré le caractère "classique" des formules (contrairement à la déduction naturelle). La clé est de pouvoir "jouer" sur deux formules à la fois dans la conclusion et de se servir des hypothèses d'une des deux formules à prouver pour conclure sur la seconde.*

Proposition 4.4 (Affaiblissement) *Si* $\Gamma \vdash \Delta$*,* $\Gamma \subseteq \Gamma'$ *et* $\Delta \subseteq \Delta'$ *alors* $\Gamma' \vdash \Delta'$

Proposition 4.5 (Propriété de la sous-formule) *Toutes les formules intervenant dans la preuve d'un séquent* $\Gamma \vdash \Delta$ *sont des sous-formules de formules de* Γ, Δ*.*

Preuve: *Il suffit de remarquer que les règles* simplifient *les formules, sans en ajouter de nouvelles.* □

Exercice 4.3 *Chercher une dérivation de la règle de modus-ponens, que constatez-vous ?*

Définition 4.4 (Règle de coupure) *On peut ajouter au système la règle dite de coupure.*

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}$$

Une présentation alternative de la coupure inclut l'affaiblissement

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

Exercice 4.4 — *Montrer que la règle de coupure est valide. Est-elle inversible ?*
— *La propriété de sous-formule reste-t-elle vraie ?*
— *Dérivier la règle de modus-ponens.*

Développement 5 (Complétude) *[Goubault-Larrecq and Mackie, 1997, thm 2.26, p33]*
Le système G est complet : tout séquent valide est dérivable sans la règle de coupure.

Preuve: *Toutes les règles sont inversibles, une interprétation satisfait la conclusion si et seulement si elle satisfait l'ensemble des prémisses.*

On peut toujours construire un arbre de dérivation jusqu'aux feuilles qui sont de la forme $\Gamma \vdash \Gamma, \Delta$ *avec* Γ, Δ *deux ensembles de variables propositionnelles. Soit* $\Gamma \cap \Delta \neq \emptyset$ *et le séquent est un axiome, soit* $\Gamma \cap \Delta = \emptyset$ *et alors le séquent est faux pour une interprétation qui rend vrai les variables de* Γ *et fausses celles de* Δ*. Si une des feuilles n'est pas un axiome alors, en propageant dans l'arbre de preuve on en déduit que l'interprétation qui invalide la feuille rend également fausse la racine de l'arbre qui n'est donc pas valide. On en déduit qu'une formule valide admet un arbre de preuve.* □

Proposition 4.6 (Elimination des coupures) *Toute preuve avec coupure peut être transformée en une preuve sans coupure.*

Preuve: *On peut passer par la correction et la complétude : toute formule prouvable avec coupure est valide et donc admet une preuve sans coupure.*

On peut aussi faire une transformation directe. On élimine les coupures les plus hautes entre deux séquents prouvés sans coupure. La preuve se fait par récurrence sur (n, m) *en utilisant un ordre lexicographique avec* n *la taille de la formule de coupure et* m *la taille de la preuve qui mène à la coupure.*

— *Si la dernière règle appliquée avant la coupure ne porte pas sur la formule de coupure alors on peut faire "remonter" la coupure.*
— *On regarde ensuite les cas où la dernière règle utilisée avant la coupure dans chacune des prémisses porte sur cette formule.*

$$\begin{array}{l} hyp\ \dfrac{A \in \Gamma}{\Gamma \vdash \Delta, A} \quad hyp\ \dfrac{A \in \Delta}{\Gamma, A \vdash \Delta} \\ cut\ \dfrac{\qquad\qquad}{\Gamma \vdash \Delta} \end{array} \longrightarrow \quad hyp\ \frac{A \in \Gamma \cap \Delta}{\Gamma \vdash \Delta}$$

$$\begin{array}{l} \neg d\ \dfrac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A} \quad \neg g\ \dfrac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta} \\ cut\ \dfrac{\qquad\qquad}{\Gamma \vdash \Delta} \end{array} \longrightarrow \quad cut\ \frac{\Gamma, A \vdash \Delta \qquad \Gamma \vdash \Delta, A}{\Gamma \vdash \Delta}$$

$$\begin{array}{l} \Rightarrow d\ \dfrac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B} \quad \Rightarrow g\ \dfrac{\Gamma, B \vdash \Delta \qquad \Gamma \vdash \Delta, A}{\Gamma, A \Rightarrow B \vdash \Delta} \\ cut\ \dfrac{\qquad\qquad}{\Gamma \vdash \Delta} \end{array} \longrightarrow \quad \begin{array}{l} cut\ \dfrac{\Gamma, A \vdash \Delta, B \qquad \Gamma, A, B \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad \Gamma \vdash \Delta, A \\ cut\ \dfrac{\qquad\qquad\qquad\qquad\qquad\qquad}{\Gamma \vdash \Delta} \end{array}$$

On utilise un affaiblissement de $\Gamma, B \vdash \Delta$ *à* $\Gamma, A, B \vdash \Delta$.

□

Développement 6 (Preuve de taille exponentielle)
[Goubault-Larrecq and Mackie, 1997, exercice 2.21, p47 et p364]

On se donne n variables propositionnelles $x_1, \ldots, x_n$ et on construit par récurrence sur n une formule Ψ_n et une formule Φ_n^m.

$$\Psi_1 \stackrel{\text{def}}{=} x_1 \qquad \Psi_{n+1} \stackrel{\text{def}}{=} x_{n+1} \Leftrightarrow \Psi_n$$
$$\Phi_0^m \stackrel{\text{def}}{=} \Psi_m \qquad \Phi_{n+1}^m \stackrel{\text{def}}{=} x_{n+1} \Leftrightarrow \Phi_n^m$$

avec $A \Leftrightarrow B \stackrel{\text{def}}{=} (A \Rightarrow B) \wedge (B \Rightarrow A)$.

— *quelle est la taille des formules Ψ_n et Φ_n en partageant le maximum de sous-termes ?*
— *proposer des règles gauches et droites pour le connecteur $\Leftrightarrow$.*
— *montrer que la formule Φ_n^n est prouvable mais que son arbre de preuve a une taille exponentielle en n.*

4.5 Résolution

La résolution est une méthode de *réfutation* c'est-à-dire que l'on cherche à montrer qu'un ensemble de formules est insatisfiable. On s'intéresse uniquement à un ensemble de clauses.

Pour montrer la validité d'un séquent $\Gamma \vdash F$, on prend l'ensemble de formules $\Gamma, \neg F$ que l'on met en forme clausale et on cherche à montrer que cet ensemble de formules est insatisfiable.

On remarque qu'une clause correspond à un séquent dont toutes les formules sont atomiques (les littéraux positifs sont à droite du séquent et les littéraux négatifs à gauche).

Définition 4.5 (Preuve par résolution) *Soit C et C' deux clauses et p une variable propositionnelle, la règle de résolution a la forme suivante :*

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

La notation $C \vee C'$ construit la clause qui contient tous les littéraux de C et C' en supprimant les doublons. On présente parfois le système sans cette convention mais en intégrant une règle de factorisation *:* $\dfrac{C \vee l \vee l}{C \vee l}$

Soit $\mathcal{E}$ un ensemble de clauses :

— *une* déduction par résolution *à partir de $\mathcal{E}$ est une suite de clauses $C_1, \ldots, C_n$ telle que pour toute clause C_i dans cette suite soit $C_i \in \mathcal{E}$, soit il existe deux clauses C_j et C_k telles que $j, k < i$ et telle que C_i soit le résultat de la règle de résolution appliquée à C_j et C_k ;*
— *une* réfutation *de $\mathcal{E}$ est une* déduction par résolution *à partir de $\mathcal{E}$ qui contient la clause vide $\bot$;*
— *une* preuve par résolution *(on dit aussi une preuve par réfutation) d'une formule quelconque A à partir d'un ensemble de clauses $\mathcal{E}$ est une réfutation de l'ensemble de clauses $\mathcal{E} \cup C_{\neg A}$ avec $C_{\neg A}$ une forme clausale de la formule $\neg A$.*

Si on réécrit les clauses sous la forme de séquent, la règle de résolution est la règle de coupure (sous la forme incluant l'affaiblissement) sur une formule atomique $\dfrac{\Gamma \vdash \Delta, p \qquad \Gamma', p \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$

Exercice 4.5 *Soit $E \stackrel{\text{def}}{=} \{p \Leftrightarrow q, \neg(p \Rightarrow q) \vee \neg(q \Rightarrow p)\}$ un ensemble de formules. Construire l'ensemble de clauses associé et chercher une réfutation pour cet ensemble.*

Développement 7 (Correction et complétude de la méthode de résolution)

[Goubault-Larrecq and Mackie, 1997, thm 2.39 p49]
La méthode de résolution est correcte et complète.
***Preuve:** La **correction** se déduit de la validité de la règle de coupure : si une interprétation satisfait les prémisses alors elle satisfait la conclusion de la règle, donc si la clause vide est dérivable, aucune interprétation*

ne peut satisfaire l'ensemble initial de formules qui est donc insatisfiable. Une réfutation de $\Gamma, \neg F$ garantit donc que $\Gamma \models F$.

*Pour la **complétude**, on montre que si un ensemble S de clauses est insatisfiable alors on peut faire à partir de S une dérivation qui contient la clause vide.*

On construit l'arbre sémantique pour l'ensemble des clauses et on coupe aux sommets d'échec (ceux pour lesquels une des formules de S est fausse, comme dans le cas de la preuve de compacité du développement 1).

On raisonne par récurrence sur la taille de l'arbre pour un ensemble quelconque de clauses S.

- *Dans quel cas l'arbre est-il réduit à la racine ?*
- *Montrer que si l'arbre n'est pas réduit à la racine alors il existe un sommet qui n'est pas un sommet d'échec mais dont les deux fils sont des nœuds d'échec.*
- *Si ce nœud est étiqueté par la variable x, montrer que les deux sommets fils correspondent à deux clauses qui contiennent l'une le littéral x et l'autre le littéral $\neg x$.*
- *Conclure.*

□

Algorithme. On peut en déduire un algorithme pour chercher la clause vide à partir d'un ensemble fini de clauses. On travaille avec un ensemble SAT de clauses qui ont déjà été traitées et l'ensemble S des clauses à examiner

On choisit C dans S, on retire C de S

- si C est la clause vide alors on a fini
- si C est une tautologie ou a été déjà traité (dans SAT) alors on recommence
- sinon on calcule les résolvants de C et des clauses de SAT et C que l'on ajoute à S, on ajoute ensuite la clause C à SAT.

Proposition 4.7 *[Carton, 2008, proposition 4.43, p 212]*

Si on part d'un ensemble fini de 2-clauses (au plus deux variables par clause), on peut décider la satisfiabilité en temps polynomial (même linéaire en le nombre de clauses).

***Preuve:** On peut toujours se ramener à avoir exactement deux littéraux par clause, en effet si on a la clause vide c'est insatisfiable, si on a un littéral seul alors on peut propager la valeur de la variable correspondante dans les autres clauses de manière linéaire en le nombre de clauses sans changer la satisfiabilité.*

La résolution entre deux 2-clauses donne encore une 2-clause. S'il y a n variables, alors il y a $2n(n-1)$ 2-clauses (on choisit un des $\frac{n(n-1)}{2}$ couple de variables et pour chaque variable si elle doit être positive ou négative).

Algorithme linéaire *On se ramène à un problème de recherche de composantes fortement connexes dans un graphe.*

On construit un graphe dont les sommets sont les littéraux (x et $\neg x$). On note $\bar{l}$ le littéral correspondant à la négation du littéral l. Une clause $u \vee v$ se traduit en deux arcs $\bar{u} \to v$ et $\bar{v} \to u$. On remarque que tout chemin de u à v induit un chemin de $\bar{v}$ à $\bar{u}$.

- *Construire le graphe sur les deux exemples suivants*

$$E \stackrel{\text{def}}{=} \{r \vee q, \neg q \vee p, \neg p \vee r\} \qquad E' = E \cup \{\neg r \vee q\}$$

1. *identifiez les composantes fortement connexes dans les deux cas*
2. *dire si les ensembles de clauses sont satisfiables*

- *Montrer que si l'ensemble de clauses E est satisfait par une interprétation ρ alors sur toutes les arêtes $s \to s'$ on si $\rho(s) = V$ alors $\rho(s') = V$.*
- *En déduire que si l'ensemble de clauses E est satisfiable alors x et $\neg x$ ne sont pas dans la même composante fortement connexe. S'il existe un chemin de l à $\bar{l}$ et que F est satisfaite par une interprétation ρ, que peut-on en déduire pour $\rho(l)$ et $\rho(\bar{l})$?*

— *On suppose maintenant que dans le graphe, x et $\neg x$ ne sont pas dans la même composante fortement connexe (c'est-à-dire qu'il peut y avoir un chemin de l'un vers l'autre mais pas dans les deux sens). Construire une interprétation qui valide l'ensemble des clauses.*
— *le calcul des composantes fortement connexes par l'algorithme de Tarjan est linéaire en le nombre de sommets + le nombre d'arêtes qui correspond ici à deux fois le nombre de variables + le nombre de clauses.*

□

Autres stratégies Il y a d'autres stratégies pour trouver la clause vide qui restent complètes. Par exemple la *résolution ordonnée* consiste à donner un ordre strict aux atomes (comme dans la représentation des obdds). Si dans la preuve de complétude, on construit un arbre sémantique dans lequel les variables sont ordonnées par ordre croissant alors on peut remarquer que lorsqu'on fait une résolution entre deux clauses aux feuilles alors le littéral sur lequel se fait la résolution est maximal parmi les littéraux des deux clauses.

La *résolution linéaire* consiste en une suite de clauses $C_1, \ldots, C_n$ telle que chaque nouvelle clause C_{i+1} est une résolution entre la clause C_i et une clause C_j avec $j \leq i$.

Proposition 4.8 (résolution linéaire ordonnée) *[Goubault-Larrecq and Mackie, 1997, thm 7.33,p 261]*
La résolution est encore complète si on se limite à une résolution linéaire ordonnée.

5 Applications

La logique propositionnelle a de nombreuses applications en informatique.
— Les outils de preuves les plus développés pour le calcul propositionnel sont les SAT-solvers qui cherchent à établir la satisfiabilité d'un problème présenté comme un ensemble de contraintes sur des variables propositionnelles :
  — tout problème NP se ramène à un problème de satisfiabilité ;
  — l'inférence de solutions permet des résolutions de problèmes : jeux, coloriage (allocation de registres dans les compilateurs), planification ...
  — l'insatisfiabilité au contraire va être recherchée pour montrer que des formules sont valides, dans le cadre de la vérification de circuits ou de programmes ;
  — l'équivalence de formules propositionnelles (représentée par des BDD) est aussi utilisée pour faire de la preuve de circuit en comparant la formule qui correspond au circuit et celle qui correspond à la spécification ;
  — les BDD sont utilisés pour représenter des ensembles d'états dans des systèemes de transition pour faire de la vérification de propriétes temporelles (techniques de model-checking).
— La base propositionnelle est également essentielle dans de nombreuses extensions tant sur le plan théorique que pratique :
  — une extension utile des SAT-solver est les SMT-solver qui font de la satisfiabilité dans une théorie donnée (égalité, arithmétique) :
    — les formules atomiques comportent des variables et des prédicats définis dans la théorie comme l'égalité ou l'inégalité ;
    — on commence par remplacer ces prédicats par des variables propositionnelles et vérifier si la formule est satisfiable ;
    — si elle ne l'est pas de manière propositionnelle, elle ne l'est pas non plus dans la théorie ;
    — si elle est satisfiable, on regarde les interprétations possibles et on vérifie si elles sont satisfiables dans la théorie ;
    — par exemple si on aboutit sur les contraintes $x < 5$ et $8 < x$ on peut conclure que le modèle propositionnel n'est pas un modèle dans la théorie.
  — la prouvabilité au premier ordre pourra se ramener à la résolution de problèmes de calcul propositionnel (via le théorème de Herbrand qui permet de se placer dans un modèle "syntaxique").