Cours Logique et Calculabilité

L3 Informatique 2013/2014

Texte par Séverine Fratani, avec addenda par Luigi Santocanale
Version du 14 janvier 2014

Table des matières

Chapitre 1

Introduction

1.1 Pourquoi la logique ?

1.1.1 La formalisation du langage

Le mot *logique* provient du grec *logos* (raison, discours), et signifie "science de la raison". Cette science a pour objets d'étude le discours et le raisonnement. Ceux-ci dépendent bien entendu du langage utilisé. Si on prend le langage courant, on se rend compte facilement :

- qu'il contient de nombreuses ambiguïtés : nous ne sommes pas toujours sûrs de la sémantique d'un énoncé ou d'une phrase. Par exemple : « nous avons des jumelles à la maison ». (Deux filles ou des lunettes optiques ?) Ou « il a trouvé un avocat » (le professionnel ou le fruit ?)
- qu'il est difficile de connaître la véracité d'un énoncé : « il pleuvra demain », « Jean est laid », « la logique c'est dur ».
- qu'il permet d'énoncer des choses paradoxales :
  1. « Je mens » : comme pour tout paradoxe de ce type, on aboutit à la conclusion que si c'est vrai alors c'est faux... et inversement.
  2. « Je suis certain qu'il n'y a rien de certain »
  3. Un arrêté enjoint au barbier (masculin) d'un village de raser tous les hommes du village qui ne se rasent pas eux-mêmes et seulement ceux-ci. Le barbier n'a pas pu respecter cette règle car :
     - s'il se rase lui-même, il enfreint la règle, car le barbier ne peut raser que les hommes qui ne se rasent pas eux-mêmes ;
     - s'il ne se rase pas lui-même (qu'il se fasse raser ou qu'il conserve la barbe), il est en tort également, car il a la charge de raser les hommes qui ne se rasent pas eux-mêmes.
  4. Paradoxe de Russel : c'est la version mathématique du paradoxe du barbier : soit a l'ensemble des ensembles qui ne se contiennent pas eux-même. Cet ensemble n'existe pas car on peut vérifier que $a \in a$ ssi $a \notin a$.

Tout ceci fait que les langues naturelles ne sont pas adaptées au raisonnement formel. C'est pourquoi par exemple, on vous a appris un langage spécifique pour faire des preuves en mathématique. Une preuve mathématique ne peut être faite en utilisant tout le vocabulaire de la langue naturelle car les énoncés et les preuves deviendraient alors ambiguës. Le langage utilisé en mathématique est celui de la logique classique (en réalité, c'est un langage un peu plus souple, entre la logique classique et la langue naturelle).

1.1.2 La formalisation du raisonnement

Une fois le langage formalisé, ce qui intéresse les logiciens c'est le raisonnement, et en particulier, la définition de systèmes formels permettant de mécaniser le raisonnement. Au début du 20ième siècle, le rêve du logicien est de faire de la logique un calcul et de mécaniser le raisonnement et par suite toutes les mathématiques. En 1930, Kurt Gödel met fin à cette utopie en présentant son résultat d'incomplétude : il existe des énoncés d'arithmétique qui ne sont pas prouvable par un

système formel de preuve. Il n'existe donc pas d'algorithme qui permette de savoir si un énoncé mathématique est vrai.

1.2 Logique et informatique

Malgré ce résultat négatif, l'arrivée de l'informatique à partir des années 30 marque l'essor de la logique.

Elle est présente dans quasiment tous les domaines de l'informatique :

— vous verrez par exemple en cours d'architecture que votre ordinateur est formé de circuits logiques.

— la programmation n'est au fond que de la logique. Dans les année 60, la correspondance de Curry-Howard, établie une correspondance preuve/programme : une relation entre les démonstrations formelles d'un système logique et les programmes d'un modèle de calcul.

— le traitement automatique des langues,

— l'intelligence artificielle,

— la logique apparait également dans toutes les questions de sureté.
On demande maintenant de plus en plus de prouver la sureté des programmes et des protocoles. Pour cela on modélise les exécutions des programmes, on exprime les propriétés de sûreté par une formule logique, puis on vérifie que les modèles satisfont bien la formule.
A cette effet, d'innombrables logiques ont étés développées, comme les logiques temporelles qui permettent de raisonner sur l'évolution de certains systèmes au cours du temps. Il existe même des logiques pour formaliser les règles des pare-feu, afin d'éviter d'avoir des systèmes de règle incohérents.

— et plein d'autres que j'oublie.

Il existe également des logiciels qui permettent de prouver des formules logique (automatiquement ou semi-automatiquement). En particulier on a des logiciels permettant de générer du code vérifié : on entre une abstraction du programme a réaliser, on prouve sur cette abstraction de façon plus ou moins automatique mais sûre, les propriétés de sûreté souhaitées ; et le logiciel produit du code certifié.

L'informatique est donc indissociable de la logique. Heureusement tout bon informaticien n'est pas obligé d'être un bon théoricien de la logique, mais il doit être capable maitriser son utilisation.

1.3 Contenu du cours

On s'intéressera principalement à (des fragments de) la logique classique, qui est la logique utilisée pour les mathématiques, et forme la base de presque toutes les autres logiques.

Nous allons nous intéresser aux fragment suivants :

— la logique propositionnelle ;

— la logique du premier ordre.

Pour chacune de ces logiques, nous nous poserons principalement les questions suivantes :

— Quelle est sa syntaxe ? I.e., comment écrire une phrase dans le langage de la logique considéré.

— Quel est sa sémantique ? C'est-à-dire, étant une phrase, savoir lui attribuer un sens. Cette question ouvre à une autre qui est "de quel forme sont les modèles d'une formule", c'est à dire : mon langage parle d'objets, qui se placent dans un univers précis : quel est cet univers ?

— La logique est elle-décidable ? Etant donné une phrase (formule) du langage, existe-il une procédure effective permettant d'évaluer cette formule.

— Existe-il un système formel de calcul permettant de "prouver" qu'un énoncé est vrai ou faux.

D'autres questions se posent évidement mais se sont principalement celles-ci qui intéressent l'informaticien.

Chapitre 2

Calcul propositionnel

2.1 Introduction

Les formules (ou phrases, ou enoncés) du calcul propositionnel sont de deux types : ou bien une formule est une *proposition atomique*, ou bien elle est composée à partir d'autres formules à l'aide des connecteurs logiques $\wedge, \neg, \vee, \Rightarrow$ (*et*, *non*, *ou*, *implique*, que l'on appelle *connecteurs propositionnels*).

Considérons, par exemple, l'énonce arithmétique « 2+2 = 4 *ou* 3+3 = 5 ». Cette formule peut se considérer comme construite des propositions atomiques « 2+2 = 4 » et « 3+3 = 5 », via le connecteur propositionnel *ou*. Une analyse similaire peut se faire pour les énoncés du langage naturel. On considère l'énonce « s'il pleut, alors le soleil se cache », que l'on reconnaîtra être équivalent à « il pleut implique que le soleil se cache », comme obtenu des deux propositions atomiques « s'il pleut » et « le soleil se cache » via le connecteur propositionnel *implique*.

Une proposition atomique est un énoncé simple, ne pouvant prendre que les valeurs "vrai" ou "faux ", et ce de façon non ambiguë ; elle donne donc une information sur un état de chose. De plus une proposition atomique est indécomposable : « le ciel est bleu et l'herbe est verte » n'est pas une proposition atomique mais la composition de deux propositions atomiques. Dans l'analyse du langage naturel, on ne peut pas considérer comme des propositions : les souhaits, les phrases impératives ou les interrogations.

Nous avons déjà vu des exemples de formules composées. Considérons maintenant l'énoncé « s'il neige, alors le soleil se cache et il fait froid ». C'est une formule composée, via le connecteur *implique*, depuis la formule atomique « il neige » et la formule composée « le soleil se cache et il fait froid ». On peut donc composer des formules à partir d'autres formules composées. La valeur de vérité d'une formule composée se calcule comme une fonction de formules dont elle est composée.

Le calcul des propositions est la première étape dans la définition de la logique et du raisonnement. Il définit les règles de déduction qui relient les phrases entre elles, sans en examiner le contenu ; il est ainsi une première étape dans la construction du calcul des prédicats, qui lui s'intéresse au contenu des propositions.

Nous partirons donc en général de faits : "p est vrai, q est faux" et essaierons de déterminer si une affirmation particulière est vraie.

2.2 Syntaxe du calcul propositionnel : les formules

Le *langage* du calcul propositionnel est formé de :
— symboles propositionnels $\textsc{Prop} = \{p_1, p_2, \ldots\}$;
— connecteurs logiques $\{\neg, \wedge, \vee, \Rightarrow\}$;
— symboles auxiliaires : parenthèses et espace.

Remarque 2.1. Dans la littérature logique on utilise plusieurs synonymes pour symbole propositionnel ; ainsi *variable propositionnelle*, *proposition atomique*, *formule atomique*, ou encore *atome* sont tous des synonymes de *symbole propositionnel*.

L'ensemble $\mathcal{F}_{\text{cp}}$ des *formules* du calcul propositionnel est le plus petit ensemble tel que :
— tout symbole propositionnel est une formule ;
— si φ est une formule alors $\neg\varphi$ est une formule ;
— si φ, ψ sont des formules alors $\varphi \vee \psi$, $\varphi \wedge \psi$ et $\varphi \Rightarrow \psi$ sont des formules.

Les symboles auxiliaires ne sont utilisés que pour lever les ambiguïtés possibles : par exemple, la formule $p \vee q \wedge r$ est ambiguë, car elle peut se lire de deux façons différentes, $((p \vee q) \wedge r)$ ou bien $(p \vee (q \wedge r))$.

Exemple. p, $p \Rightarrow (q \vee r)$ et $p \vee q$ sont des formules propositionnelles ; $\neg(\vee q)$ et $f(x) \Rightarrow g(x)$ n'en sont pas.

A cause de la structure inductive de la définition, une formule peut-être vue comme un arbre dont les feuilles sont étiquetés par des symboles propositionnels et les noeuds par des connecteurs. Par exemple, la formule $p \Rightarrow (\neg q \wedge r)$ correspond à l'arbre représenté Figure 2.2.

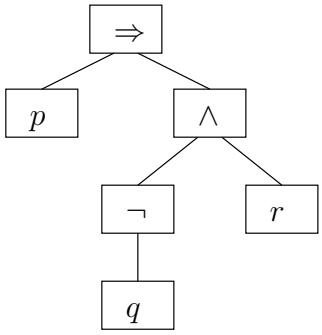

FIGURE 2.1 – Représentation arborescente de la formule $p \Rightarrow (\neg q \wedge r)$

Notation 2.2. On utilise souvent en plus le connecteur binaire $\Leftrightarrow$ comme abréviation : $\varphi \Leftrightarrow \psi$ est l'abréviation de $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$. De la même façon, on ajoute le symbole $\bot$ qui correspond à *Faux* et le symbole $\top$ qui correspond à *Vrai*. Ces deux symboles sont aussi des abréviations, ils ne sont pas indispensables au langage. (Par exemple $\bot$ peut être utilisé à la place de $p \wedge \neg p$ et $\top$ à la place de $p \vee \neg p$.)

Définition 2.3 (Sous-formule)**.** L'ensemble $SF(\varphi)$ des sous-formules d'une formules φ est défini par induction de la façon suivante.
— $SF(p) = \{p\}$;
— $SF(\neg\varphi) = \{\neg\varphi\} \cup SF(\varphi)$;
— $SF(\varphi \circ \psi) = \{\varphi \circ \psi\} \cup SF(\varphi) \cup SF(\psi)$ (où $\circ$ désigne un des symboles $\wedge$, $\vee$, $\Rightarrow$).

Par exemple, $SF(p \Rightarrow (\neg q \wedge r)) = \{p, q, r, \neg q, \neg q \wedge r, p \Rightarrow (\neg q \wedge r)\}$. Quand on voit une formule comme un arbre, une sous-formule est simplement un sous-arbre (voir Figure 2.2).

Définition 2.4 (Sous-formule stricte)**.** ψ est une sous-formule stricte de φ si ψ est une sous-formule de φ qui n'est pas φ.

2.3 Sémantique du calcul propositionnel

Il faut maintenant un moyen de déterminer si une formule est vraie ou fausse. La première étape est de donner une valeur de vérité aux propositions atomiques. L'évaluation d'une formule, dépend donc des valeurs choisies pour les symboles propositionnels. Ces valeurs sont données par une **valuation**.

Définition 2.5 (Valuation)**.** Une valuation est une application de PROP dans $\{0, 1\}$. La valeur 0 désigne le "faux" et la valeur 1 désigne le "vrai".

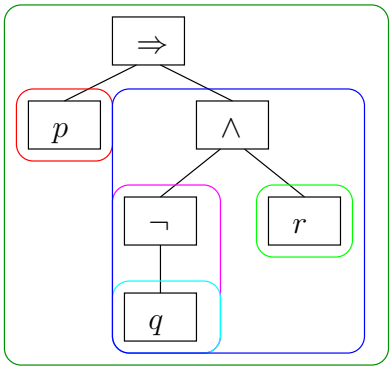

FIGURE 2.2 – Représentation arborescente des sous-formules de $p \Rightarrow (\neg q \wedge r)$

Une valuation sera souvent donnée sous forme d'un tableau. Par exemple, si PROP $= \{p, q\}$ alors la valuation $v : p \mapsto 1,\ q \mapsto 0$ s'écrit plus simplement v :

p	q
1	0

Une fois la valuation v choisie, la valeur de la formule se détermine de façon naturelle, par extension de la valuation v aux formules de la façon suivante :

Définition 2.6 (Valeur d'une formule)**.**
- $v(\neg\varphi) = 1$ ssi $v(\varphi) = 0$;
- $v(\varphi \vee \psi) = 1$ ssi $v(\varphi) = 1$ ou $v(\psi) = 1$;
- $v(\varphi \wedge \psi) = 1$ ssi $v(\varphi) = 1$ et $v(\psi) = 1$;
- $v(\varphi \Rightarrow \psi) = 0$ ssi $v(\varphi) = 1$ et $v(\psi) = 0$.

La définition précédente peut apparaître trompeuse car circulaire : afin d'expliquer la logique, nous somme en train de l'utiliser (*ssi*, *ou*, *et* ...). Par ailleurs, on peut se servir de la définition suivante qui est en effet équivalente à la Définition 2.6 :

Définition 2.7 (Valeur d'une formule (bis))**.**
- $v(\neg\varphi) = 1 - v(\varphi)$;
- $v(\varphi \vee \psi) = \max(v(\varphi), v(\psi))$;
- $v(\varphi \wedge \psi) = \min(v(\varphi), v(\psi))$;
- $v(\varphi \Rightarrow \psi) = v(\neg\varphi \vee \psi)$.

Cette dernière définition est purement combinatoire car elle repose sur la structure de l'ensemble ordonné *fini* $\{\, 0 < 1 \,\}$; nous supposons en fait que cette structure est tellement évidente et claire per se qu'il n'y a pas besoin de la justifier autrement.

Exercice 2.8. Proposez un algorithme qui, étant donné une formule φ du calcul propositionnel et une valuation v, calcule $v(\varphi)$. Quel type de structure de données utiliser pour coder les formules ? Quel type de structure de données utiliser pour coder les valuations ?

Notez que la définition 2.7 correspond aux tables de vérité des connecteurs logiques (dont vous avez sûrement entendu parler) :

p	$\neg p$
0	1
1	0

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

p	q	$p \Rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

Remarque 2.9. Remarquez la définition particulière de l'implication : on l'entend en général comme un "si ..., alors ...", on voit ici que l'énoncé "si 1+1=1, alors la capitale de la France est Marseille" est vrai, puisque toute phrase $\varphi \Rightarrow \psi$ est vraie dès lors que φ est évaluée à faux. Ceci est

peu naturel, car dans le langage courant, on ne s'intéresse à la vérité d'un tel énoncé que lorsque la condition est vraie : "s'il fait beau je vais à la pêche" n'a d'intérêt pratique que s'il fait beau... Attribuer la valeur vrai dans le cas ou la prémisse est fausse correspond a peu près à l'usage du si .. alors dans la phrase suivante : "Si Pierre obtient sa Licence, alors je suis Einstein" : c'est à dire que partant d'une hypothèse fausse, alors je peux démontrer des choses fausses (ou vraies). Par contre, il n'est pas possible de démontrer quelque chose de faux partant d'une hypothèse vraie.

On peut ajouter la définition de la valeur de l'abréviation $\Leftrightarrow$: $v(\varphi \Leftrightarrow \psi) = 1$ ssi $v(\varphi) = v(\psi)$. Ce qui correspond à la table de vérité suivante :

p	q	$p \Leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

Exercice 2.10. Définissons l'ensemble $\textsc{Prop}(\varphi)$, des variables propositionnelles contenues dans $\varphi \in \mathcal{F}_{\text{cp}}$, par induction comme suit :

$$\begin{aligned} \textsc{Prop}(p) &= \{\, p \,\}, \\ \textsc{Prop}(\neg\varphi) &= \textsc{Prop}(\varphi), \\ \textsc{Prop}(\varphi \circ \psi) &= \textsc{Prop}(\varphi) \cup \textsc{Prop}(\psi), \qquad\qquad \circ \in \{\, \wedge, \vee, \Rightarrow \,\}. \end{aligned}$$

Montrez que :
- $\textsc{Prop}(\varphi) = \textsc{Prop} \cap SF(\varphi)$, pour tout $\varphi \in \mathcal{F}_{\text{cp}}$;
- si $v(p) = v'(p)$ pour tout $p \in \textsc{Prop}(\varphi)$, alors $v(\varphi) = v'(\varphi)$.

2.3.1 Modèles d'une formule

Définition 2.11. L'ensemble des **valuations** d'un ensemble de variables propositionnelles Prop est noté $\mathsf{Val}(\textsc{Prop})$ (ou juste Val lorsqu'il n'y a pas d'ambiguité sur Prop). $\mathsf{Val}(\textsc{Prop})$ est donc l'ensemble des fonctions de Prop dans $\{0, 1\}$.

Par exemple, si $\textsc{Prop} = \{p, q, r\}$, alors Val est représenté par la Table 2.3.1, dans lequel chaque ligne est une valuation de Prop :

p	q	r
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Table 2.1 – L'ensemble $\mathsf{Val}(\textsc{Prop})$ des valuations de $\textsc{Prop} = \{p, q, r\}$

Définition 2.12 (Modèle d'une formule)**.** Un **modèle** de φ est une valuation v telle que $v(\varphi) = 1$. On note $\mathsf{mod}(\varphi)$ l'ensemble des modèles de φ.

Exemple. Si $\textsc{Prop} = \{p, q, r\}$ et $\varphi = (p \vee q) \wedge (p \vee \neg r)$ alors l'ensemble des modèles de φ est

$\mathsf{mod}(\varphi) =$

p	q	r
0	1	0
1	0	0
1	1	1
1	1	0
1	1	1

Définition 2.13 (Satisfaisabilité)**.** Une formule φ est **satisfaisable** (ou **consistante**, ou encore **cohérente**) si elle admet un modèle (*i.e.*, si il existe une valuation v telle que $v(\varphi) = 1$, *i.e.* si $\mathsf{mod}(\varphi) \neq \varnothing$).

Définition 2.14 (Insatisfaisabilité)**.** Une formule φ est **insatisfaisable** (ou **inconsistante**, ou **incohérente**) si elle n'admet aucun modèle (*i.e.*, si pour toute valuation v, $v(\varphi) = 0$, *i.e.*, si $\mathsf{mod}(\varphi) = \varnothing$)

Définition 2.15 (Tautologie)**.** Une formule φ est une **tautologie** (ou **valide**) si $v(\varphi) = 1$ pour toute valuation v (*i.e.*, si $\mathsf{mod}(\varphi) = \mathsf{Val}$). On note $\models \varphi$ pour dire que φ est une tautologie.

Un exemple de tautologie est $\varphi \vee \neg\varphi$, c'est à dire le *tiers exclus*.

Exercice 2.16. Montrez que les formules suivantes sont des tautologies :

$$p \Rightarrow p\,, \qquad p \Rightarrow (q \Rightarrow p)\,, \qquad (p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))\,, \qquad ((p \Rightarrow q) \Rightarrow p) \Rightarrow p\,.$$

Définition 2.17 (Equivalence)**.** On dit que φ **est équivalente à** ψ si les deux formules ont les mêmes modèles (*i.e.* si $\mathsf{mod}(\varphi) = \mathsf{mod}(\psi)$). On note alors $\varphi \equiv \psi$.

Exemple. Les opérateurs $\wedge$, $\vee$ sont associatifs-commutatifs. Deux formules identiques à associativité-commutativité près sont équivalentes. Remplacer une sous-formule ψ d'une formule φ par une formule équivalente ψ' donne une formule notée $\varphi[\psi \leftarrow \psi']$. Cette substitution préserve les modèles, *i.e.*, $\mathsf{mod}(\varphi) = \mathsf{mod}(\varphi[\psi \leftarrow \psi'])$.

Exercice 2.18. Écrivons $\varphi \equiv \psi$ si $v(\varphi) = v(\psi)$ pour toute valuation v (voir définition 2.17). Montrez :

$$\begin{array}{lll}
\varphi \vee \bot \equiv \varphi & \varphi \wedge \bot \equiv \bot & (\varphi \wedge \psi) \wedge \theta \equiv \varphi \wedge (\psi \wedge \theta) \\
\varphi \vee \top \equiv \top & \varphi \wedge \top \equiv \varphi & (\varphi \wedge \psi) \vee \theta \equiv (\varphi \wedge \theta) \vee (\psi \wedge \theta) \\
\varphi \vee \psi \equiv \psi \vee \varphi & \varphi \wedge \psi \equiv \psi \wedge \varphi & (\varphi \vee \psi) \wedge \theta \equiv (\varphi \wedge \theta) \vee (\varphi \wedge \theta) \\
\varphi \vee \varphi \equiv \varphi & \neg(\varphi \vee \psi) \equiv (\neg\varphi) \wedge (\neg\psi) & (\varphi \vee \psi) \vee \theta \equiv \varphi \vee (\psi \vee \theta) \\
\neg\neg\varphi \equiv \varphi & \varphi \wedge \varphi \equiv \varphi & \neg(\varphi \wedge \psi) \equiv (\neg\varphi) \vee (\neg\psi)\,.
\end{array}$$

Proposition 2.19. *Soient φ et ψ deux formules, on a :*

1. $\mathsf{mod}(\neg\varphi) = \mathsf{Val} - \mathsf{mod}(\varphi)$ *;*
2. $\mathsf{mod}(\varphi \vee \psi) = \mathsf{mod}(\psi) \cup \mathsf{mod}(\psi)$ *;*
3. $\mathsf{mod}(\varphi \wedge \psi) = \mathsf{mod}(\psi) \cap \mathsf{mod}(\psi)$ *;*
4. $\models \varphi \Rightarrow \psi$ *ssi* $\mathsf{mod}(\varphi) \subseteq \mathsf{mod}(\psi)$.

Démonstration. 1. pour toute valuation $v \in \mathsf{Val}$,

$$\begin{array}{rll}
v \in \mathsf{mod}(\neg\varphi) & \text{ssi} & v(\neg\varphi) = 1 \\
& \text{ssi} & v(\varphi) = 0 \\
& \text{ssi} & v \notin \mathsf{mod}(\varphi) \\
& \text{ssi} & \mathsf{Val} - \mathsf{mod}(\varphi)
\end{array}$$

2. pour toute valuation $v \in \mathsf{Val}$,

$$\begin{array}{rll}
v \in \mathsf{mod}(\varphi \vee \psi) & \text{ssi} & v(\varphi) = 1 \text{ ou } v(\psi) = 1 \\
& \text{ssi} & v \in \mathsf{mod}(\varphi) \text{ ou } v \in \mathsf{mod}(\psi) \\
& \text{ssi} & \mathsf{mod}(\varphi) \cup \mathsf{mod}(\psi)
\end{array}$$

3. pour toute valuation $v \in \mathsf{Val}$,

$$\begin{array}{rll}
v \in \mathsf{mod}(\varphi \wedge \psi) & \text{ssi} & v(\varphi) = 1 \text{ et } v(\psi) = 1 \\
& \text{ssi} & v \in \mathsf{mod}(\varphi) \text{ et } v \in \mathsf{mod}(\psi) \\
& \text{ssi} & \mathsf{mod}(\varphi) \cap \mathsf{mod}(\psi)
\end{array}$$

4.

$$\begin{aligned}
\models \varphi \Rightarrow \psi \quad & \text{ssi} \quad \text{pour toute valuation } v \in \mathsf{Val}, v(\varphi \Rightarrow \psi) = 1 \\
& \text{ssi} \quad \text{pour toute valuation } v \in \mathsf{Val}, v(\neg\varphi \vee \psi) = 1 \\
& \text{ssi} \quad \text{pour toute valuation } v \in \mathsf{Val}, v(\varphi) = 0 \text{ ou } v(\psi) = 1 \\
& \text{ssi} \quad \text{pour toute valuation } v \in \mathsf{Val}, v(\varphi) \leq v(\psi) \\
& \text{ssi} \quad \mathsf{mod}(\varphi) \subseteq \mathsf{mod}(\psi) \quad \square
\end{aligned}$$

Définition 2.20 (Conséquence logique)**.** Une formule ψ est **conséquence logique** d'une formule φ si tout modèle de φ est un modèle de ψ (*i.e.*, si $\mathsf{mod}(\varphi) \subseteq \mathsf{mod}(\psi)$). On note alors $\varphi \models \psi$.

Remarque 2.21. Attention à la confusion dans les deux notations !
— $v \models \varphi$ où v est une valuation, i.e., l'assignation d'une valeur aux propositions atomiques de la formule ; c'est un raccourcis assez fréquent pour $v(\varphi) = 1$;
— $\psi \models \varphi$ où ψ est une formule.

Proposition 2.22. *Soient φ et ψ deux formules propositionnelles.*

1. *$\varphi \models \psi$ si et seulement si $\models \varphi \Rightarrow \psi$.*
2. *$\varphi \equiv \psi$ si et seulement si $\models \varphi \Leftrightarrow \psi$.*

Démonstration. 1. Conséquence directe du point 4 de la Proposition 2.19

2. $\models \varphi \Leftrightarrow \psi$ ssi $\forall v \in \mathsf{Val}, v(\varphi \Leftrightarrow \psi) = 1$ (par la définition de tautologie) ssi $\forall v \in \mathsf{Val}, v(\varphi) = v(\psi)$ (par la table de verité de $\Leftrightarrow$) ssi $\forall v \in \mathsf{Val}$, $v \in \mathsf{mod}(\varphi)$ ssi $v \in \mathsf{mod}(\psi)$ ssi $\mathsf{mod}(\varphi) = \mathsf{mod}(\psi)$ ssi $\psi \equiv \varphi$. $\square$

2.3.2 La conséquence logique (d'un ensemble de formules)

Les formules propositionnelles peuvent être vues comme un ensemble de contraintes sur les propositions. Ainsi, $p \wedge q$ contraint p et q à être vraies. Il est donc très courant de considérer des ensembles de formules propositionnelles pour modéliser des problèmes de satisfaction de contraintes.

On étend les définitions vues précédemment aux ensembles de formules.

Définition 2.23 (Modèle)**.** Un **modèle** d'un ensemble de formules Γ est une valuation v telle que $v(\varphi) = 1$ pour tout $\varphi \in \Gamma$. On note $\mathsf{mod}(\Gamma)$ l'ensemble des modèles de Γ.

Cet ensemble de modèles est donc l'ensemble des valuations qu'on peut attribuer aux variables si on veut respecter toutes les contraintes de Γ.

Définition 2.24 (Satisfaisabilité/Consistance)**.** Un ensemble de formules Γ est **satisfaisable** (ou **consistant**, ou **cohérent**) si il admet au moins un modèle (*i.e.*, si $\mathsf{mod}(\Gamma) \neq \varnothing$)

Définition 2.25 (Insatisfaisabilité/Contradiction)**.** Un ensemble de formules Γ est **insatisfaisable** (ou **contradictoire**, ou **inconsistant**, ou encore **incohérent**) si il n'admet aucun modèle (*i.e.*, si $\mathsf{mod}(\Gamma) = \varnothing$), on note alors $\Gamma \models \bot$.

Un ensemble Γ contradictoire ne peut être satisfait : par exemple l'ensemble $\Gamma = \{p, \neg p\}$ est insatisfaisable.

Définition 2.26 (Conséquence logique)**.** Une formule φ est conséquence logique de Γ si et seulement si toute valuation qui donne 1 à toutes les formules de Γ donne 1 à φ (*i.e.*, si $\mathsf{mod}(\Gamma) \subseteq \mathsf{mod}(\varphi)$), on note alors $\Gamma \models \varphi$. On note $\mathrm{cons}(\Gamma)$ l'ensemble des conséquences logiques de Γ.

Remarque 2.27. Attention à la confusion dans les notations !
— $v \models \varphi$ où v est une valuation : cette notation est souvent utilisée pour dire que v rend vraie la formule φ ; elle est donc une autre notation pour $v(\varphi) = 1$ [1] ;
— $\psi \models \varphi$ où ψ est une formule ;
— $\Gamma \models \varphi$ où Γ est un ensemble de formule.

1. Dans ce texte, nous essayerons d'éviter cette notation.

Remarquez, par ailleurs que $\varphi \models \psi$ si, et seulement si, $\{\varphi\} \models \psi$; les derrières notations sont donc cohérentes entre elles.

Une conséquence logique φ d'un ensemble Γ est une nouvelle contrainte déduite directement de Γ. Puisqu'elle découle de Γ, elle ne peut pas apporter de contraintes supplémentaires que celles apportées par Γ. En particulier, les modèles de Γ et ceux de $\Gamma \cup \{\varphi\}$ sont exactement les mêmes (cette intuition sera prouvée formellement avec la Proposition 2.32).

Par exemple, si $\Gamma = \{(p \Rightarrow s) \vee q, \neg q\}$, on peut en déduire que $p \Rightarrow s$. Bien entendu, les modèles de Γ sont exactement les modèles de $\Gamma \cup \{p \Rightarrow s\}$.

Ceci implique donc également une méthode de simplification d'un ensemble de formule, si Γ contient une formule φ conséquence logique de $\Gamma - \{\varphi\}$, alors φ peut être retirée de l'ensemble de contraintes Γ sans en modifier la sémantique : $\mathsf{mod}(\Gamma) = \mathsf{mod}(\Gamma - \{\varphi\})$.

Proposition 2.28. *$\Gamma \models \varphi$ ssi $\Gamma \cup \{\neg\varphi\}$ est contradictoire.*

Démonstration. $\Gamma \models \varphi$ ssi
pour toute valuation v
- soit v est un modèle de Γ et $v(\varphi) = 1$
- soit v n'est pas un modèle de Γ

ssi pour toute valuation v
- soit v est un modèle de Γ et $v(\neg\varphi) = 0$
- soit v n'est pas un modèle de Γ

ssi pour toute valuation v, v n'est pas un modèle de $\Gamma \cup \{\neg\varphi\}$
ssi $\Gamma \cup \{\neg\varphi\}$ est contradictoire. □

Proposition 2.29. *Pour tous ensembles de formules Σ, Γ,*

$$\mathit{mod}(\Sigma \cup \Gamma) = \mathit{mod}(\Sigma) \cap \mathit{mod}(\Gamma).$$

En particulier, si $\Sigma \subseteq \Gamma$ alors $\mathit{mod}(\Gamma) \subseteq \mathit{mod}(\Sigma)$.

Cette proposition se comprend bien si on voit un ensemble de formules comme un ensemble de contraintes sur les variables propositionnelles. Plus on ajoute de contraintes, et moins il reste de possibilités pour résoudre ces contraintes.

Démonstration. Pour toute valuation v :

$$\begin{aligned} v \in \mathsf{mod}(\Sigma \cup \Gamma) \text{ ssi } & \text{ pour tout } \varphi \in \Sigma \cup \Gamma, v(\varphi) = 1 \\ \text{ssi } & \text{ pour tout } \varphi \in \Sigma, v(\varphi) = 1 \text{ et pour tout } \psi \in \Gamma,\ v(\psi) = 1 \\ \text{ssi } & v \in \mathsf{mod}(\Sigma) \text{ et } v \in \mathsf{mod}(\Gamma) \\ \text{ssi } & v \in \mathsf{mod}(\Sigma) \cap \mathsf{mod}(\Gamma). \end{aligned}$$

□

Les preuves des propositions suivantes sont laissées en exercice.

Proposition 2.30. *Si $\Gamma' \subseteq \Gamma$ et $\Gamma' \models \varphi$, alors $\Gamma \models \varphi$.*

Proposition 2.31. *$\{\varphi_1, \ldots \varphi_n\} \models \psi$ ssi $\models (\varphi_1 \wedge \ldots \wedge \varphi_n) \Rightarrow \psi$*

Cette proposition exprime le fait qu'un ensemble **fini** de formules propositionnelles peut toujours être vu comme une seule formule formée de la conjonction des formules de l'ensemble. Une formule étant un objet fini, ce résultat ne se généralise pas au cas des ensembles de taille non bornée. Dans le cas où Γ est infini, il faudra utiliser le théorème de compacité (Théorème 2.35) qui permet de ramener les problèmes de satisfaisabilité et de contradiction d'un ensemble de taille quelconque à celle d'ensemble finis.

Démonstration. Remarquons que

$$\begin{aligned} \mathsf{mod}(\{\varphi_1, \ldots, \varphi_n\}) &= \mathsf{mod}(\{\varphi_1\} \cup \ldots \cup \{\varphi_n\}) \\ &= \mathsf{mod}(\{\varphi_1\}) \cap \ldots \cap \mathsf{mod}(\{\varphi_n\}) \\ &= \mathsf{mod}(\varphi_1) \cap \ldots \cap \mathsf{mod}(\varphi_n) = \mathsf{mod}(\varphi_1 \wedge \ldots \wedge \varphi_n)\,. \end{aligned}$$

Donc, on a que $\mathsf{mod}(\{\varphi_1, \ldots, \varphi_n\}) \subseteq \mathsf{mod}(\psi)$ si et seulement si $\mathsf{mod}(\varphi_1 \wedge \ldots \wedge \varphi_n) \subseteq \mathsf{mod}(\psi)$ et, par la Proposition 2.19, la dernière relation est vraie ssi $(\varphi_1 \wedge \ldots \wedge \varphi_n) \Rightarrow \psi$ est une tautologie. □

Proposition 2.32. $\Gamma \models \varphi$ *si, et seulement si,* $\mathit{mod}(\Gamma) = \mathit{mod}(\Gamma \cup \{\varphi\})$.

Démonstration. La proposition decoule du fait que $\mathsf{mod}(\Gamma \cup \{\varphi\}) = \mathsf{mod}(\Gamma) \cap \mathsf{mod}(\{\varphi\})$, et que la relation $\mathsf{mod}(\Gamma) \subseteq \mathsf{mod}(\varphi)$ est equivalenete à $\mathsf{mod}(\Gamma) \cap \mathsf{mod}(\{\varphi\}) = \mathsf{mod}(\Gamma)$. □

Exemple. Avec cet exemple, nous allons tirer avantage des propositions et remarques précédentes pour résoudre un ensemble de contraintes ayant une certaine complexité.

On dispose de 4 variables propositionnelles, p_A, p_B, p_C, P_D, qui obéissent aux contraintes suivantes :

$\varphi_1 : p_B \wedge \neg p_C$
$\varphi_2 : p_A \Rightarrow (p_C \vee p_D)$
$\varphi_3 : \neg p_C \wedge (p_B \vee p_A)$

Soit $\Gamma_1 = \{\varphi_1, \varphi_2, \varphi_3\}$, cet ensemble forme l'ensemble des prémisses à partir desquelles nous allons essayer de déduire les valeurs que les variables propositionnelles peuvent prendre.

On se pose les questions suivantes :

1. *Peut-on simplifier l'ensemble* Γ_1 *de façon à ne pas changer l'ensemble de ses modèles, et donc de ses conséquences ?*
   (a) On remarque que la contrainte φ_3 est une **conséquence logique** de la contrainte φ_1.
   En effet, pour toute valuation v,
   — v satisfait φ_1 ssi $v(p_C) = 0$ et $v(p_B) = 1$,
   — v satisfait φ_3 ssi $v(p_C) = 0$ et ($v(p_A) = 1$ ou $v(p_B) = 1$).
   D'où, $\mathsf{mod}(\varphi_1) \subseteq \mathsf{mod}(\varphi_3)$.
   (b) Soit $\Gamma_2 = \{\varphi_1, \varphi_2\}$, on a $\mathsf{mod}(\Gamma_2) = \mathsf{mod}(\Gamma_1)$. En effet, par définition,
   $\mathsf{mod}(\Gamma_1) = \mathsf{mod}(\varphi_1) \cap \mathsf{mod}(\varphi_2) \cap \mathsf{mod}(\varphi_3) = \mathsf{mod}(\varphi_1) \cap \mathsf{mod}(\varphi_2) = \mathsf{mod}(\Gamma_2)$.
   Donc les conséquences de Γ_1 et Γ_2 sont les mêmes et on peut alors simplifier Γ_1 par Γ_2.
2. *Quel est l'ensemble des modèles de* Γ_2 *?*

Modèles de φ_1 :

p_A	p_B	p_C	p_D
0	1	0	0
0	1	0	1
1	1	0	0
1	1	0	1

Modèles de φ_2 :

p_A	p_B	p_C	p_D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	1
1	1	1	0
1	1	1	1

L'ensemble M des modèles de Γ_2 est l'intersections des modèles de φ_1 et φ_2 :

p_A	p_B	p_C	p_D
0	1	0	1
1	1	0	1

3. Γ_2 *est-il consistant ? contradictoire ?*
   Γ_2 admet un modèle, il est donc consistant et non contradictoire.
4. *Quelles conséquences logiques pouvons nous tirer de l'ensemble* Γ_2 *?*
   Les conséquences logiques de Γ_2 sont toutes les formules dont l'ensemble des modèles contient M. On a donc entre autres :
   $\neg p_C$, p_B, $p_B \wedge \neg p_C \in \mathrm{cons}(\Gamma)$
5. Ajoutons maintenant une nouvelle contrainte : $\Gamma_3 = \{\neg p_B \wedge \neg p_D\} \cup \Gamma_2$. On a $\mathsf{mod}(\Gamma_3) = \mathsf{mod}(\Gamma_2) \cap \mathsf{mod}(\neg p_B \wedge \neg p_D) = \varnothing$. Donc $\mathsf{mod}(\Gamma_3) = \varnothing$ et Γ_3 est contradictoire.

2.3.2.1 Compacité

Le théorème de compacité sert à caractériser la conséquence logique dans les cas où l'ensemble des formules est infini en ne considérant que des sous-ensembles finis. Par ailleurs, ce théorème jouera un rôle cruciale plus tard, dans la preuve du théorème de complètude pour la calcul de la résolution (Théorème **??**).

Le théorème de compacité Pour commencer, nous avons besoin du lemme suivant appelé Lemme de König.

Lemme 2.33. *Tout arbre infini à branchement fini possède une branche infinie.*

Démonstration. Supposons que T, qui est à branchement fini, soit infini. On définit une branche infinie dans T, ce qui mènera à la conclusion.

Pour cela, on montre par induction la propriété suivante :

P(n) : := il existe une branche $e_0, \dots e_n$ telle que le sous arbre issu de e_n est infini.

— Pour $n = 0$, on choisit $e_0 = r$ (la racine de l'arbre) qui par hypothèse est racine d'un arbre infini.

— On suppose P(n) et on montre P(n+1) : Par hypothèse, il existe une branche $e_0, \dots e_n$ telle que le sous arbre issu de e_n est infini. Considérons les successeurs immédiats de e_n, disons $e_{n_1}, \dots, e_{n_k}$: si tous étaient racines de sous-arbres finis, disons de cardinaux $p_1, \dots, p_k$, alors il en serait de même de e_n (avec un cardinal d'au plus $p_1 + ... + p_{k+1}$), contradiction. Donc l'un d'entre eux est le e_{n+1} recherché.

□

Nous aurons besoin du Lemme dans la forme suivante :

Lemme 2.34. *Tout arbre a branchement fini et dont toutes les branches sont finies, est fini.*

Théorème 2.35 (Compacité)**.** *Un ensemble de formules propositionnelles* Γ *est satisfaisable ssi tout sous-ensemble fini de* Γ *est satisfaisable.*

Par contraposée, le théorème de compacité peut s'énoncer de la façon suivante :

Théorème 2.36 (Compacité)**.** *Un ensemble de formules propositionnelles* Γ *est contradictoire ssi il existe un sous-ensemble fini de* Γ *contradictoire.*

Remarquons que l'implication « si un sous-ensemble fini de Γ est contradictoire, alors Γ est contradictoire » est trivialement vraie. Nous nous limiterons à prouver l'implication inverse.

Démonstration. On fait la preuve dans le cas dénombrable.

Nous avons besoin d'une construction importante appelée "arbre sémantique" ou "arbre de Herbrand". Considérons un ensemble dénombrable d'atomes $\{p_0, p_1, p_2, \dots, p_n, \dots\}$. L'arbre sémantique associé est un arbre binaire infini dont toutes les arêtes à gauches sont étiquetées "faux" et celles à droites sont étiquetées "vrai". Chaque niveau de l'arbre est associé à une proposition. La racine (le niveau 0) est associé à p_0 : chaque fois que l'on descend d'un noeud de niveau p_i, ceci revient à poser p_i faux si l'on descend à gauche, et p_i vrai si l'on descend à droite.

— Chaque chemin infini partant de la racine correspond à une valuation de l'ensemble des propositions.

— Chaque noeud e à profondeur n correspond à une valuation v_e des variables $\{p_0, \dots p_{n-1}\}$.

— Nous appelons un noeud e *noeud d'échec* si sa profondeur est n et il existe une formule $\varphi_e \in \Gamma$ telle que $\textsc{Prop}(\varphi_e) \subseteq \{ p_0, \dots p_{n-1} \}$ et $v_e(\varphi_e) = 0$.

On suppose que Γ est inconsistante et on montre qu'il existe un sous-ensemble $\Gamma_0 \subseteq \Gamma$ fini et inconsistant.

On commence par remarquer que chaque branche contient un noeud d'échec. En effet, si une branche n'en contient pas, elle définit un modèle de Γ, ce qui est contradictoire à l'hypothèse.

On peut donc faire la construction suivante : Prenons le premier noeud d'échec de chaque branche et étiquetons le par une formule de Γ fausse sur ce noeud, puis coupons l'arbre au niveau du noeud d'echec.

L'arbre obtenu en tronquant ainsi toutes les branches est un arbre à branchement fini, ses branches sont finies donc il est fini. Le sous-ensemble Γ_0 des formules de Γ étiquetant les feuilles de l'arbre est donc fini. Or toutes les feuilles de l'arbre sont des noeuds d'échec et donc chacune des valuations rend fausse au moins une des formules de Γ_0. L'ensemble Γ_0 est fini et inconsistant □

Corollaire du théorème de compacité :

Corollaire 2.37. *Une formule φ est conséquence d'un ensemble de formules Γ si et seulement si il existe un sous-ensemble fini Γ_{fini} de Γ tel que $\Gamma_{fini} \models \varphi$.*

Démonstration.

$\Gamma \models \varphi$ ssi $\Gamma \cup \{\neg\varphi\}$ est contradictoire
ssi il existe $\Gamma_f \subseteq \Gamma \cup \{\neg\varphi\}$ fini et contradictoire
ssi il existe $\Gamma_f \subseteq \Gamma$ fini tel que $\Gamma_f \cup \{\neg\varphi\}$ est contradictoire
ssi il existe un sous-ensemble fini $\Gamma_f \subseteq \Gamma$ tel que $\Gamma_f \models \varphi$. □

2.3.3 Décidabilité du calcul propositionnel

Une logique est décidable s'il existe un algorithme (calcul réalisable sur un ordinateur qui termine toujours pour toute donnée) qui permet de savoir pour chaque formule si elle est une tautologie (i.e. si $\models \varphi$) ou pas.

Théorème 2.38. *Le calcul propositionnel est décidable.*

Démonstration. Méthode des tables de vérité : calculer la table de vérité prenant en argument les symboles propositionnels de φ et calculer pour chaque valuation possible la valeur de φ.
Coût : $O(2^n)$ avec n la taille de φ (nombre de propositions). □

Nous verrons par la suite qu'il y a de meilleurs algorithmes, mais qu'il ont tous un coût exponentiel. La plupart de ces algorithmes débutent par une première phase de normalisation de la formule, c'est à dire qu'on modifie la syntaxe de la formule de manière à la mettre sous une forme normalisée, tout en conservant la sémantique de la formule, c'est-à-dire l'ensemble de ses modèles.