Logique

David Baelde, ENS Rennes, L3 SIF

11 janvier 2023 (en chantier)

Table des matières

Chapitre 1

Préliminaires

Un des objectifs de la formalisation de la logique est d'éviter les raisonnements incorrects. Un exemple célèbre est le paradoxe de Russell : si l'on peut former $R = \{x \mid x \notin x\}$ on déduit $R \in R \Leftrightarrow R \notin R$, puis $\bot$. Même si nous n'avons pas pour objectif de bien poser la théorie des ensembles, cet exemple doit nous inciter à questionner les raisonnements et les définitions que nous nous autorisons à faire.

1.1 Définitions inductives

Considérons la définition inductive d'élément accessible par rapport à l'ordre canonique dans $\mathbb{N}$, puis $\mathbb{Z}$. Comment comprendre et justifier ces définitions ?

Proposition 1.1.1. *Soit E un ensemble. Soit $f : 2^E \to 2^E$ une fonction monotone sur les parties de E, c'est à dire qu'on a, pour tous $X, Y \subseteq E$ tels que $X \subseteq Y$, $f(X) \subseteq f(Y)$. La fonction f admet un* plus petit point fixe *: il existe X tel que $X = f(X)$ et tout autre ensemble ayant cette propriété contient X.*

Démonstration. On définit X comme l'intersection des ensembles Y tels que $f(Y) \subseteq Y$.

- Montrons que $f(X) \subseteq X$. Soit Y tel que $f(Y) \subseteq Y$. On a $X \subseteq Y$ par définition de X, donc $f(X) \subseteq f(Y)$ puis $f(X) \subseteq Y$. On en déduit $f(X) \subseteq X$ par définition de X.
- On montre ensuite que $X \subseteq f(X)$. En fait, par le point précédent et la monotonicité de f on a $f(f(X)) \subseteq f(X)$, d'où $X \subseteq f(X)$ par définition de X.
- Il est enfin clair que $X \subseteq Y$ pour tout Y tel que $f(Y) \subseteq Y$. □

Ce plus petit point fixe contient les itérées de f à partir de l'ensemble vide : $\emptyset \subseteq X$, puis $f(\emptyset) \subseteq f(X) \subseteq X$, $f^2(\emptyset) \subseteq X$, etc. Mais ces inclusions sont en général strictes – considérer par exemple l'accessibilité sur $\mathbb{N} \cup \{\omega\}$ avec $i < \omega$ pour tout $i \in \mathbb{N}$.

Quand on définit inductivement un prédicat sur E, qu'on peut voir comme un sous-ensemble de E (le sous-ensembles de valeurs pour lesquelles p est vrai), on dit en fait que p est le plus petit point fixe de la fonction associée à la définition, qui doit être monotone. C'est le cas pour l'accessibilité :

$$f(X) = \{n \mid \forall m.\ m < n \Rightarrow m \in X\}$$

Exemple 1.1.1. *On peut tout à fait définir inductivement p par "p est vrai si p est vrai" : c'est une façon détournée de décrire le prédicat faux. Pour*

voir un prédicat sans argument comme un sous-ensemble, il faut considérer les sous-ensembles d'un singleton $S = \{\mathbf{1}\}$: l'ensemble vide est le prédicat faux, l'ensemble plein est le prédicat faux. Ici, notre prédicat p défini inductivement est le plus petit point fixe de la fonction identité sur 2^S, c'est à dire l'ensemble vide.

Il est important de comprendre ce qui n'est pas une définition inductive, i.e. ce que ne permet pas le théorème précédent.

Exemple 1.1.2. *On ne peut pas définir inductivement p par "p est vrai si p est faux" – ce qui nous ramènerait au paradoxe de Russell. La fonction associée est définie par $f(\emptyset) = S$ et $f(S) = \emptyset$, et n'est donc pas monotone.*

Chapitre 2

Logique propositionnelle

2.1 Syntaxe

On se donne un ensemble $\mathcal{P}$ de *variables propositionnelles*.

Définition 2.1.1. *L'ensemble des formules du calcul propositionnel est défini inductivement comme suit :*
- $\bot$ *et* $\top$ *sont des formules ;*
- *les variables propositionnelles sont des formules ;*
- *si* ϕ *est une formule, alors* $\neg\phi$ *aussi ;*
- *si* ϕ *et* ψ *sont des formules, alors* $\phi \wedge \psi$, $\phi \vee \psi$ *et* $\phi \Rightarrow \psi$ *aussi.*

Mais sur quel ensemble de départ se place-t-on quand on crée cette définition ? On peut considérer qu'on se place sur un ensemble d'arbres étiquetés, ou de graphes.

Cette définition peut être vue comme la définition d'un type de données algébrique en OCaml :

```
type form = Bot | Top | Var of var | Not of form | ...
```

C'est une intuition utile, mais la variante OCaml permet des objets récursifs qui sont exclus ici : par exemple, `let rec x = Not x`.

Certains considèrent que les formules sont des suites de symboles, mais il faut alors ajouter des symboles parenthèses dans la définition ci-dessus pour assurer une lecture unique : on préfère une vision de plus haut niveau. Cela n'empêche pas d'écrire nos formules-arbres en utilisant une notation linéaire, avec des règles de priorité pour nos opérateurs et des parenthèses pour désambiguer :
- On considèrera que les trois connecteurs binaires sont associatifs à droite.
- On considèrera que le $\wedge$ lie plus fortement que $\vee$, qui lie plus fortement que $\Rightarrow$.

Ainsi, $A \vee B \wedge C \vee D$ correspond à $A \vee ((B \wedge C) \vee D)$.

2.2 Sémantique

Pour donner un sens à nos formules, on utilise la notion d'interpétation : une interprétation est une fonction $\mathcal{I} : \mathcal{P} \to \{0, 1\}$.

Définition 2.2.1. *La relation de satisfaction entre les interprétations et les formules, notée* $\mathcal{I} \models \phi$*, est définie par induction sur les formules :*
- $\mathcal{I} \models \top$ *et* $\mathcal{I} \not\models \bot$ *;*
- $\mathcal{I} \models A$ *ssi* $\mathcal{I}(A) = 1$ *;*
- $\mathcal{I} \models \phi \wedge \psi$ *ssi* ($\mathcal{I} \models \phi$ *et* $\mathcal{I} \models \psi$) *;*

— $\mathcal{I} \models \phi \vee \psi$ *ssi* ($\mathcal{I} \models \phi$ *ou* $\mathcal{I} \models \psi$) ;
— $\mathcal{I} \models \phi \vee \psi$ *ssi* ($\mathcal{I} \models \phi$ *implique* $\mathcal{I} \models \psi$) ;
— $\mathcal{I} \models \neg\phi$ *ssi* $\mathcal{I} \not\models \phi$.

Il faut bien noter ici que la relation de satisfaction n'est pas définie inductivement.

À partir de la relation de satisfaction on peut définir une fonction d'interprétation, qu'on peut noter sans confusion $\mathcal{I}(\phi)$ et définie par $\mathcal{I}(\phi) = 1$ si $\mathcal{I} \models \phi$ et 0 sinon. Ces deux définitions peuvent être posées dans l'autre sens : si l'on définit d'abord $\mathcal{I}$, on définit ensuite $\mathcal{I} \models \phi$ par $\mathcal{I}(\phi) = 1$.

Dans ce cours, on utilise en priorité la notation $\models$ qui est de toute façon standard, et en fait plus versatile en logique.

On dérive à partir de la notion de satisfaction de nombreuses notions incontournables.

Définition 2.2.2. *L'ensemble des modèles d'une formule ϕ est l'ensemble des interprétations $\mathcal{I}$ qui satisfont ϕ, i.e. $\mathcal{I} \models \phi$.*

Définition 2.2.3. *Une formule ϕ est* valide *si elle est satisfaite par toute interprétation : $\mathcal{I} \models \phi$ pour tout $\mathcal{I}$.*

Définition 2.2.4. *La formule ψ est* conséquence logique *de la formule ϕ, ce qu'on note $\phi \models \psi$, si tous les modèles de ϕ sont aussi des modèles de ψ :*

$$\mathcal{I} \models \phi \text{ implique } \mathcal{I} \models \psi \text{ pour tout } \mathcal{I}$$

Définition 2.2.5. *Deux formules sont* logiquement équivalentes *si chacune est conséquence logique de l'autre, i.e. elles ont les mêmes modèles. L'équivalence logique de ϕ et ψ est notée $\phi \equiv \psi$.*

Plusieurs équivalentes logiques remarquables, quelles que soient ϕ et ψ : $\neg\neg\phi \equiv \phi$, $\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$, etc.

2.3 Formes normales

Proposition 2.3.1. *Pour toute formule ϕ il existe une formule logiquement équivalente ψ n'utilisant pas le connecteur $\Rightarrow$ et dans laquelle la négation n'est utilisée que sur des variables.*

Exemple 2.3.1. *$\neg(\neg P \Rightarrow Q)$ sera transformé en $\neg P \wedge \neg Q$.*

Démonstration.

— Première possibilité : par induction sur la taille (ou la hauteur de) la formule. Si la formule n'est pas construite (à toplevel) avec une négation ou une implication, on conclut aisément par hypothèse d'induction. Sinon, on se ramène à une formule équivalente avant d'invoquer l'hypothèse d'induction : par exemple pour ϕ de la forme $\neg(\phi_1 \Rightarrow \phi_2)$ on appliquera l'hypothèse d'induction sur ϕ_1 et $\neg\phi_2$ pour obtenir ϕ'_1 et ϕ'_2 et on conclut avec $\phi'_1 \wedge \phi'_2$.
— Deuxième possibilité, équivalente : on définit une fonction / un algorithme qui calcule récursivement la transformation, on constate sa correction partielle et on vérifie aisément qu'il termine.
— Troisième possibilité : on oriente les équivalences logiques utiles comme des règles de réécriture, qu'on applique autant que possible sans stratégie particulière ; on vérifie qu'une quantité décroit à chaque réécriture et qu'on a la forme normale attendue quand on ne peut plus réécrire.
Ici, une quantité naturelle est la somme des tailles des sous-formules commençant par une négation ou une implication. □

Dans tous les cas il y a unicité [1] de la forme normale obtenue, qui a une taille linéaire en la taille de la formule de départ, et peut être calculée en temps polynomial.

Définition 2.3.1. *Un litéral est une variable ou la négation d'une variable.*

Proposition 2.3.2 (Forme normale conjonctive)**.** *Toute formule est logiquement équivalente à une conjonction de disjonctions de litéraux.*

Proposition 2.3.3 (Forme normale disjonctive)**.** *Toute formule est logiquement équivalente à une disjonction de conjonctions de litéraux.*

2.4 Le problème de la satisfaisabilité

On a déja vu que SAT, le problème général de satisfaisabilité, est NP-complet : c'est le théorème de Cook. Si on rentrait dans les détails de la preuve, on verrait qu'on a en fait (modulo ajustements mineurs) une preuve que CNF-SAT est NP-complet. Il n'est pas utile de le détailler car on va faire mieux plus bas.

Il n'est pas intéressant de restreindre SAT en fixant l'ensemble des variables utilisables — pourquoi ? Par contre, il est intéressant de voir ce qu'il se passe si on restreint la forme syntaxique des formules :

— Il est clair que DNF-SAT se résout en temps polynomial.
— On verra plus tard que 2-SAT est polynomial et même linéaire.
— En fait, 3-SAT est encore NP-complet. Cela dérive du résultat suivant.

Définition 2.4.1. *On dit que deux formules sont equi-satisfaisables quand la satisfaisabilité de l'une est équivalente à la satisfaisabilité de l'autre. Autrement dit, soit les deux sont satisfaisables, soit aucune ne l'est.*

Proposition 2.4.1 (Transformation de Tseitin)**.** *Pour toute formule ϕ on peut calculer en temps polynomial une formule équi-satisfaible ψ (de taille linéaire en $|\phi|$) en forme 3-CNF.*

Démonstration. Soit ϕ une formule de départ. On se donne des nouvelles variables P_ψ pour toute sous-formule de ϕ, y compris ϕ.

On considère la formule $T(\phi) = P_\phi \wedge \bigwedge_{\psi \text{ sous-formule}} t(\psi)$ où les formules $t(\psi)$ sont définies comme suit :

— ...

On vérifie qu'on a bien construit une 3-CNF. On constate de plus qu'il existe une borne k sur la taille de tous les $t(\psi)$, donc la taille de notre formule est linéaire en la taille de ϕ.

Vérifions que la formule construite est équi-satisfaisable à ϕ :

— Si on a un modèle $\mathcal{I} \models \phi$ on l'étend en $\mathcal{I}'$ tel que $\mathcal{I}'(P_\psi) = 1$ ssi $\mathcal{I} \models \psi$. On vérifie que $\mathcal{I}' \models T(\phi)$.
— Si on a un modèle $\mathcal{J} \models T(\phi)$ on prend $\mathcal{J}'$ sa restriction aux variables de ϕ et on vérifie $\mathcal{J}' \models \phi$.

□

1. Il n'y a pas unicité dans l'absolu : l'énoncé de notre proposition permet de donner $\bot$ comme forme normale pour $\neg\neg P \wedge \neg P$, mais nos transformations donnent $P \wedge \neg P$.