Cours 4

Induction

Calcul propositionnel : syntaxe et sémantique

Bibliographie

Induction

- A. Arnold et I. Guessarian, *Mathématiques pour l'informatique*

Logique

- R. Lassaigne et M. de Rougemont, *Logique et fondements de l'informatique*
- R. Cori et D. Lascar, *Logique mathématique*
- J. Stern, *Fondements mathématiques de l'informatique*
- M. Huth et M. Ryan, *Logic in Computer Science : modelling and reasoning about systems*

Induction

L'induction

Un outil formel élégant très utile en informatique

- elle permet de faire des définitions ”récursives”
  - d'ensembles
  - de fonctions
- elle propose une technique de preuve souvent plus élégante que la récursion sur les entiers
  - mais la récursion sur les entiers est toujours possible
- nous l'avons déjà (implicitement) utilisée pour présenter le $\lambda-$calcul

Définitions inductives

La définition inductive d'une partie X d'un ensemble consiste

- en la donnée explicite de certains éléments de X (*bases*),
- en la donnée de moyens de construire de nouveaux éléments de X à partir d'éléments déjà connus (*étapes inductives*).

Définition inductives

Définition

Soit E un ensemble. Une *définition inductive* d'une partie X de E consiste en la donnée

- d'un sous ensemble B de E,
- d'un ensemble K de fonctions (partielles) $\Phi : E^{a(\Phi)} \longrightarrow E$, où $a(\Phi) \in \mathbb{N}$ est *l'arité* de Φ.

X est défini comme étant **le plus petit** ensemble vérifiant les assertions (B) et (I) suivantes

(B) $B \subseteq X$

(I) $\forall \Phi \in K, \ \forall x_1, \ldots, x_{a(\Phi)} \in X, \ \Phi(x_1, \ldots, x_{a(\Phi)}) \in X.$

Définition inductives

L'ensemble ainsi défini est donc

$$X = \bigcap_{Y \in \mathcal{F}} Y$$

où $\mathcal{F} = \{Y \subseteq E \mid B \subseteq Y \text{ et } Y \text{ vérifie } (I)\}$.

- Cela justifie le terme "**le plus petit** ensemble".

Notation : nous pourrons noter une définition inductive sous la forme

(B) $x \in X \quad (\forall x \in B)$

(I) $x_1, \ldots, x_{a(\Phi)} \in X \Longrightarrow \Phi(x_1, \ldots, x_{a(\Phi)}) \in X \quad (\forall \Phi \in K)$.

Exemples

L'ensemble $P \subseteq \mathbb{N}$ des entiers pairs

(B) $0 \in P$

(I) $n \in P \Longrightarrow n+2 \in P$

L'ensemble $\mathrm{AB} \subseteq (A \cup \{\emptyset, (\ ,\ )\ ,\ ;\ \})^*$ des arbres binaires sur un alphabet A

(B) $\emptyset \in \mathrm{AB}$

(I) $\forall a \in A,\ g, d \in \mathrm{AB} \Longrightarrow (a; g; d) \in \mathrm{AB}$

Définition explicite

Théorème

Si X est défini inductivement par les conditions (B) et (I), tout élément de X peut s'obtenir à partir de la base en appliquant un nombre fini d'étapes inductives.

$$X = \bigcup_{n \in \mathbb{N}} X_n$$

où

$$\begin{array}{rcl} X_0 & = & B \\ X_{n+1} & = & X_n \cup \{\Phi(x_1, \ldots, x_{a(\Phi)}) \mid x_1, \ldots, x_{a(\Phi)} \in X_n \text{ et } \Phi \in K\} \end{array}$$

Remarque : tout élément de X peut être représenté graphiquement par une structure arborescente.

Les types récursifs cachent une définition inductive

En Caml

```
type list =
 | Nil
 | Cons of int*list
```

En Coq

```
Inductive list : Set :=
 | Nil
 | Cons (n:nat) (l:list).
```

ou

```
Inductive list : Set :=
 | Nil: list
 | Cons: nat→list→list.
```

Preuve par induction

Théorème

Soit X un ensemble défini inductivement par les conditions (B) et (I), et soit $\mathcal{P}(x)$ un prédicat exprimant une propriété de l'élément x de X. Si les conditions suivantes sont vérifiées :

- $\mathcal{P}(x)$ *est vraie pour chaque* $x \in B$,
- *pour tout* $x_1, \ldots, x_{a(\Phi)} \in X$, *si* $\mathcal{P}(x_1), \ldots, \mathcal{P}(x_{a(\Phi)})$ *sont vraies alors* $\mathcal{P}(\Phi(x_1, \ldots, x_{a(\Phi)}))$ *est vraie, pour tout* $\Phi \in K$,

alors $\mathcal{P}(x)$ *est vraie pour tout* $x \in X$.

Coq : génération d'un principe d'induction

```
Check list_ind.
>list_ind : ∀ P : list → Prop,
       P Nil →
       (∀ (n : nat) (l : list), P l → P (Cons n l)) →
       ∀ l : list, P l
```

Exercice

On définit sur les arbres binaires AB

- le *nombre de feuille* $f(x)$ d'un arbre x comme le nombre d'occurrence du symbole $\emptyset$,
- le *nombre de nœud* $n(x)$ d'un arbre x comme le nombre d'occurrence des symboles de A.

Montrer que pour tout $x \in \mathrm{AB}$

$$n(x) \leqslant 2f(x) - 1$$

Définition non-ambiguë

Définition

La définition inductive d'un ensemble X par les conditions (B) et (I) est dite *non-ambiguë* si

- pour tout $x_1, \ldots, x_{a(\Phi)} \in X$ et $\Phi \in K$, $\Phi(x_1, \ldots, x_{a(\Phi)}) \notin B$,
- et pour tout $x_1, \ldots, x_{a(\Phi)}, x'_1, \ldots, x'_{a(\Phi')} \in X$ et $\Phi, \Phi' \in K$, $\Phi(x_1, \ldots, x_{a(\Phi)}) = \Phi'(x'_1, \ldots, x'_{a(\Phi')})$ implique $\Phi = \Phi'$ et $x_1 = x'_1, \ldots, x_{a(\Phi)} = x'_{a(\Phi)}$.

Exercice : donner des exemples de

- définition non-ambiguë
- définition ambiguë

Fonctions définies inductivement

Théorème

Soit $X \subseteq E$ un ensemble défini inductivement par les conditions (B) et (I) tel que X est non-ambiguë, soit F un ensemble quelconque, soit f_B une fonction de $B \longrightarrow F$ et une famille de fonctions $f_\Phi \in E^{2a(\Phi)} \longrightarrow F$, pour tout $\Phi \in K$.
Il existe une unique fonction $f \in X \longrightarrow F$ telle que

- *pour tout $x \in B$,*
$$f(x) = f_B(x)$$
- *pour tout $x_1, \ldots, x_{a(\Phi)} \in X$ et $\Phi \in K$,*
$$f(\Phi(x_1, \ldots, x_{a(\Phi)})) = f_\Phi(x_1, \ldots, x_{a(\Phi)}, f(x_1), \ldots, f(x_{a(\Phi)}))$$

Fonctions définies par induction Caml/Coq

En Caml

```
let rec length : list → int = function
 | Nil → 0
 | Cons (x,l) → 1 + length l
```

En Coq

```
Fixpoint length (l:list) : nat :=
 match l with
   | Nil ⇒ 0
   | Cons x q ⇒ 1 + length q
 end.
```

Calcul propositionnel

Logique : motivations

Mathématiques

- comprendre la nature du raisonnement
- formaliser le raisonnement : en faire une théorie mathématique, s'assurer la cohérence.
- mécaniser le raisonnement[1]

Informatique : faire raisonner les machines

- intelligence artificielle
- vérification des programmes

[1] Voir fin du cours pour les (més)aventures de Hilbert et Gödel.

Le calcul propositionnel

Objectifs

- Formaliser "et", "ou", "implique", "non",..
- Noyau minimal commun à tous les systèmes logiques

Les ingrédients d'un système logique (ou système formel)

- Qu'est-ce qu'une formule ? (syntaxe)
- Quel sens donner à une formule ? (sémantique)
- Comment démontrer qu'une formule est *vraie* ? (systèmes de déduction)

Attention à ne pas confondre les formules (syntaxe) et leurs interprétations (sémantique) !

Syntaxe

$\mathbb{V}$ un ensemble dénombrable de symboles, appelés *variables propositionnelles*.
C ensemble fini de connecteurs : $C = \{\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\}$

Définition (Formules propositionnelles)

L'ensemble $\mathfrak{F}$ des formules propositionnelles est un langage sur $\mathbb{V} \cup C \cup \{(,)\}$ défini inductivement par :

- $\mathbb{V} \subseteq \mathfrak{F}$
- si $F \in \mathfrak{F}$, alors $\neg F \in \mathfrak{F}$
- si $F, G \in \mathfrak{F}$, alors $(F \wedge G), (F \vee G), (F \Rightarrow G), (F \Leftrightarrow G) \in \mathfrak{F}$

Définition explicite

On pose

- $\mathfrak{F}_0 = \mathbb{V}$,
- $\mathfrak{F}_{n+1} = \mathfrak{F}_n \cup \{\neg F \mid F \in \mathfrak{F}_n\} \cup \{(F \, \alpha \, G) \mid F, G \in \mathfrak{F}_n, \ \alpha \in C\}$

Lemme

$$\mathfrak{F} = \bigcup_{n \in \mathbb{N}} \mathfrak{F}_n$$

Principe d'induction sur les formules

Comment prouver une propriété sur les formules ?

- par récurrence sur la hauteur
- par induction sur les formules

Théorème

Si une propriété $\mathcal{P}(F)$ vérifie

- $\mathcal{P}(F)$ *est vraie pour tout formule* $F \in \mathbb{V}$,
- *si* $\mathcal{P}(F)$ *est vraie pour* $F \in \mathcal{F}$ *alors* $\mathcal{P}(\neg F)$ *est vraie,*
- *et si* $\mathcal{P}(F)$ *et* $\mathcal{P}(G)$ *sont vraies pour* $F, G \in \mathcal{F}$ *alors* $\mathcal{P}((F \wedge G))$, $\mathcal{P}((F \vee G))$, $\mathcal{P}((F \Rightarrow G))$ *et* $P((F \Leftrightarrow G))$ *sont vraies,*

alors $\mathcal{P}(F)$ *vraie pour toute formule* F.

Hauteur d'une formule

Définition (hauteur)

La *hauteur* d'une formule $F \in \mathcal{F}$ est le plus petit entier n tel que $F \in \mathcal{F}_n$.

Exercice : Démontrer que pour tout formule F,

$$\text{hauteur}(F) < \text{longueur}(F)$$

Décomposition unique

Théorème (Décomposition unique)

Soit F une formule, un et un seul des 3 cas suivants se présente :

- $F \in \mathbb{V}$
- *il existe une unique formule G telle que $F = \neg G$*
- *il existe un unique $\alpha \in C \setminus \{\neg\}$, et deux uniques formules G et H telles que $F = (G \; \alpha \; H)$*

(cf TD ou Cori&Lascar)

Remarque : En d'autres termes, la définition inductive de $\mathcal{F}$ est non-ambiguë.

Substitution

Définition (Substitution)

La formule $F[G/p]$ (*substitution de G à p dans F*) est définie par induction sur la formule F :

- si $F = p$, $F[G/p] = G$
- si $F = q \in \mathbb{V} \setminus \{p\}$, $F[G/p] = F$
- si $F = \neg H$, $F[G/p] = \neg H[G/p]$
- si $F = (F_1 \propto F_2)$ avec $\propto \; \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$,
  $F[G/p] = (F_1[G/p] \propto F_2[G/p])$

Valuation

Nous interprétons maintenant les formules en termes de valeur de vérité.

Définition

Une distribution de valeurs de vérité (*valuation*) est une application $\varphi : \mathbb{V} \rightarrow \{\text{vrai,faux}\}$ (où $\{0, 1\}$, $\{T, F\}$).

Opérations booléennes

$[\![\neg]\!] : \{0,1\} \rightarrow \{0,1\}$

x	$[\![\neg]\!](x)$
0	1
1	0

$[\![\alpha]\!] : \{0,1\} \times \{0,1\} \rightarrow \{0,1\}, \forall \alpha \in C \setminus \{\neg\}$

x	y	$[\![\wedge]\!](x,y)$	$[\![\vee]\!](x,y)$	$[\![\Rightarrow]\!](x,y)$	$[\![\Leftrightarrow]\!](x,y)$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Valeur de vérité

Définition

La valeur de vérité $\overline{\varphi}(F)$ d'une formule F, par rapport à une valuation φ est définie inductivement par :

- $\overline{\varphi}(p) = \varphi(p),\ p \in \mathbb{V}$
- $\overline{\varphi}(\neg F) = [\![\neg]\!](\overline{\varphi}(F))$
- $\overline{\varphi}((F \alpha G)) = [\![\alpha]\!](\overline{\varphi}(F), \overline{\varphi}(G)), \forall \alpha \in C \setminus \{\neg\}$

Remarque : $\overline{\varphi}$ sera éventuellement noté φ.

Tautologies, formules équivalentes

Définition

- Une formule F est *satisfaite* pour une valuation φ si $\varphi(F) = 1$.
- Une *tautologie* est une formule satisfaite pour toute valuation.
- Deux formules F et G sont dites *équivalentes* si pour toute valuation φ, $\varphi(F) = \varphi(G)$ (c'est une relation d'équivalence notée $\equiv$).

Exemple : $((p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r)))$ est une tautologie.

Conséquence, satisfiabilité

Définition

Soit Σ un ensemble de formules, F une formule.

- F est *conséquence* de Σ (noté $\Sigma \models F$) si toute valuation qui satisfait toutes les formules de Σ satisfait aussi F.
- Σ est *satisfiable* s'il existe une valuation qui satisfait toutes les formules de Σ.

Proposition

$\Sigma \models F$ si et seulement si $\Sigma \cup \{\neg F\}$ est non satisfiable.

Plan