Logique et Preuve

Philippe Duchon – duchon@labri.fr

Année 2019-20

Qu'est-ce que la logique?

La logique est une science dont l'objet est l'étude du raisonnement, abstraction faite du domaine auquel ce raisonnement s'applique:

- philosophie
- mathématiques
- sûreté et sécurité des logiciels et matériels informatiques
- ...

Historique

- Cette discipline est très ancienne, les premiers ouvrages sur la logique ayant été écrits par Aristote (-384 -322).
- Renouveau fin XIX-ème – début XX-ème siècle : crise des fondements des mathématiques.
    - **Hilbert, 1900:** peut-on trouver une méthode systématique pour prouver toutes les vérités mathématiques? (on ne parle pas encore d'algorithme...)
    - **Gödel, 1930:** non seulement on ne peut pas, mais dans *n'importe quel système de preuve*, il existe des assertions qui sont "vraies" mais qu'il est impossible de prouver.
- Devient une part importante du géniel logiciel : *certification* du matériel et du logiciel
- Recherches mathématiques utilisant des *assistants de preuve*.

Problèmes constatés (licence, master (GL, MA))

- Difficultés avec le langage logique
  - Confusion entre $\Rightarrow$ et $\wedge$ (le fameux “donc”)
  - Erreurs dans la *spécification* des programmes
- Modes de raisonnement :
  - Confusion entre *axiomes* et *hypothèses*
  - Difficultés avec le *raisonnement par l’absurde*
  - etc.

Contenu de l'UE

Fondements de la logique

- Introduction *tranquille* des notions importantes. On s'appuie sur votre familiarité avec la notion de langage de programmation (*syntaxe*, *sémantique*, etc.)
- On part du *minimum* (juste l'implication), et on procède par ajoûts successifs : contradiction et négation, connecteurs, calculs des prédicats
- Premiers pas avec un assistant de preuve

Exemples informatiques visés

- Spécification de fonctions simples
- Preuves de correction de fonctions (récursives)

Organisation

Séances

- 3 cours en amphi : généralités, quelques compléments
- 15 Séances de cours intégrés (2 par semaine)
- 6 Séances de TD sur machine (groupées par 2): utilisation de l'asssistant de preuve coq

Contrôle des connaissances

- Contrôle continu : 4 tests (papier) d'un 1/4 h, un TD machine noté : on prend la moyenne
- Examen final d'1h30
- Moyenne des deux notes (en seconde session: si l'examen de seconde session est meilleur que le CC, on ne garde que l'examen)

Équipe pédagogique

- **Le cours en amphi:** Ph. Duchon
- **Les cours-TD/TM:**
  - S. Archipoff (Info A2)
  - F. Carrère (Info A4)
  - Ph. Duchon (Info A5, Maths-Info)
  - M. Senhaji (Info A1, A3)

Documentation

Site http://www.labri.fr/~duchon/Enseignements/L-et-P

- poly (pas mis à jour...à prendre avec des pincettes)
- un livre très complet sur coq: **Le Coq'Art** http://www.labri.fr/perso/casteran
- **Introduction à la logique - Théorie de la démonstration**, R. David, K. Nour, Ch. Raffalli (Dunod)
- divers liens: tutoriel coq, QED...
- à part le poly, toutes ces ressources vont bien plus loin que le contenu du cours

Le vocabulaire de la logique

- Propositions
- Séquents
- Vérité et preuves

Propositions

"Une proposition est un énoncé (mathématique, informatique, mais pas que) susceptible d'être démontré ou réfuté, pour lequel il fait sens de parler de vérité." Voir référence sur le poly. (Synonyme pour nous: *formule*)

Exemples (1)

- "Mon mot de passe est "password" "
- "42 est un nombre premier"
- "41 et 43 sont des nombres premiers jumeaux"
- "Le langage `C` comporte plusieurs failles de sécurité"
- " *Quicksort* est un algorithme correct de tri de tableaux"

Exemples (2)

“ La fonction ci-dessous (de recherche dichotomique) est buggée”

```
int binary_search(long t[], int n, long v) {
  int low = 0, high = n - 1;
  while (low <= high) {
    int middle = (low + high) / 2;
    if (t[middle] < v)
       low = middle + 1;
    else if (t[middle] > v)
       high = middle -1;
    else return middle;
 }
 return -1;
}
```

Exemples (3)

"La suite de Syracuse finit toujours par atteindre le cycle $(4, 2, 1)$ quel que soit l'élément initial"

initialisation Le premier élément est un nombre entier strictement positif.

calcul de l'élément suivant Soit n un élément de la suite; par construction, il est strictement positif.

- Si n est pair, l'élément suivant est $n/2$
- Sinon, l'élément suivant est $3n + 1$

Par exemple, si l'on démarre de 23, on obtient la suite:

$$23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, \ldots$$

Non-propositions

- 42
- $b^2 - 4ac$
- L'algorithme *Quicksort*
- L'ensemble des entiers naturels

Quelques logiques

Il n'y a pas *une* logique, mais plusieurs, caractériséees par la syntaxe de leurs propositions et leur domaine d'application.

Logique minimale:

Formules *atomiques* et *implication* comme seul connecteur:

$$(P \to Q \to R) \to (Q \to P \to R)$$

Application : Étude du *raisonnement hypothétique*

Quelques logiques (2)

Logiques propositionnelles:

On ajoute à la logique minimale la *contradiction* $\perp$, la *négation* $\sim$, et les *connecteurs* $\vee$, $\wedge$, $\iff$

$$P \vee \perp \rightarrow P$$

$$\sim(P \rightarrow Q \rightarrow R) \rightarrow P \wedge Q \wedge \sim R$$

Quelques logiques (3)

Calcul des prédicats:

Notions de prédicat, de relation; quantificateurs existentiel et universel

$$(\forall x, \exists y,\ x \neq y) \rightarrow \sim(\forall x\, y,\ x = y)$$

$$(\forall h, \mathrm{humain}(h) \rightarrow \mathrm{mortel}(h)) \wedge \mathrm{humain}(\mathrm{socrate})$$
$$\rightarrow \mathrm{mortel}(\mathrm{socrate})$$

$$\forall M, \exists n, \mathrm{premier}(n) \wedge (n > M)$$

$$\forall n\, p\, r, (p \neq 0 \wedge n \ \mathbf{mod}\ p = b \rightarrow (\exists q, n = a \times q + r \wedge 0 \leq r < b))$$

Séquents

Certaines propositions (par exemple: $x = 2$) ne peuvent être tenues pour vraies ou fausses par elles-mêmes (indépendemment d'un *contexte* formé d' *hypothèses*).

Définition

Un *séquent* est une structure composée:

- d'un *contexte* formé d'un ensemble Γ de propositions appelées *prémisses* ou *hypothèses* ,
- d'une proposition A appelée *conclusion* du séquent

Notations usuelles : $H_1, H_2, \ldots, H_n \vdash A$
$\Gamma \vdash A$
$\vdash A$

Validité d'un séquent

Intuitivement, un séquent est valide si, chaque fois que *toutes* ses hypothèses sont *simultanément* vraies, alors sa conclusion est vraie.

Quels sont les séquents valides?

$$x \in \mathbb{R},\ (x-2)(x-1/2)=0 \vdash x=2 \tag{1}$$
$$x \in \mathbb{R},\ (x-2)(x-1/2)=0 \vdash x=2 \lor x=1/2 \tag{2}$$
$$x \in \mathbb{Z},\ (x-2)(x-1/2)=0 \vdash x=2 \tag{3}$$
$$x = 62,\ x=3 \vdash x=2 \tag{4}$$
$$x \in \mathbb{R},\ x^2+x+1=0 \vdash x=2 \tag{5}$$
$$x \in \mathbb{C},\ x^2+x+1=0 \vdash x=2 \tag{6}$$

Notions de preuve

- En mathématiques, une **preuve** est un **discours** (plus ou moins symbolique, plus ou moins détaillé) dont l'objectif est de (se) convaincre qu'**une certaine affirmation est vraie**.
- Le niveau de détail est à adapter au public(!) (un mathématicien n'écrira pas la même preuve selon qu'il s'adresse à d'autres mathématiciens, ou à des étudiants de première année)
- critères de "qualité" d'une preuve: "élégance", "pouvoir explicatif" (**Proofs from the Book**)
- **Preuve formelle:** à l'opposé, on parle de preuve formelle quand on détaille suffisamment pour que la **vérification** de la preuve puisse être systématisée au point d'être confiée à une machine. Cela implique la définition précise d'un formalisme.
- (Dans notre cours, on s'intéresse surtout à cette notion de preuve formelle)

Vérité et Preuves

Comment [se] convaincre de la validité d'un séquent $\Gamma \vdash A$?

Deux approches

- *Sémantique:* Définir la valeur de vérité de toute proposition, puis vérifier que chaque fois que toutes les hypothèses dans Γ sont vraies, alors A est vraie. Exemple : tables de vérité en logique propositionnelle.
- *Syntaxique:* Définir ce qu'est une *preuve* de $\Gamma \vdash A$. On définit un *langage* pour les preuves, avec ses règles de correction, comme les langages de programmation. Une preuve devient un objet informatique, comme un programme.

Sémantique

- $+$ Vérification automatisable pour *certaines* logiques
- $-$ Les calculs cachent le contenu intuitif
- $-$ Grande complexité (par exemple tables de vérité)

Syntaxique

- $-$ souvent *incomplète*
- $+$ permet de mieux comprendre le rôle des connecteurs et quantificateurs
- $+$ existence d' *assistants de preuves* (preuve interactive et vérification automatique)
- $+$ familiarité avec la programmation (isomorphisme de Curry-Howard)

Quelques notions importantes

Les notions suivantes sont souvent mal comprises, et par conséquent mal utilisées:

- Axiomes et hypothèses
- L'implication
- Le faux et la négation

Schéma d'un raisonnement

1. Contexte global (définitions, axiomes, hypothèses)
2. Raisonnement
3. Conclusion

Les points 2 et 3 peuvent être vérifiés par machine, et donc ne posent pas de problèmes particuliers.

Les écueils à éviter

Axiomes:

Les axiomes peuvent être dangereux : s'ils sont incohérents ou décrivent mal la réalité, ils peuvent faire accepter n'importe quel programme faux comme correct.
Par exemple, si un logiciel de preuve de programmes **C** utilise l'axiome : $\forall\, i : \mathrm{int},\ i < i + 1$, alors tout programme (même buggé) peut être prouvé correct.

Définitions:

Exemple : mauvaises définitions d'un tableau trié :

- $\forall\, i\, j,\ (0 \leq i \leq j < n \rightarrow t[i] < t[j])$
- $\forall\, i\, j,\ (0 \leq i \leq j < n\ \wedge\ t[i] < t[j])$
- $\forall\, i\, j,\ (0 \leq i < j < n \rightarrow t[i] < t[j])$

Axiomes contre Hypothèses

- Un axiome est une proposition A que l'on admet comme vraie (sans preuve!), et qui peut être utilisée dans une preuve de B. Exemple : $\forall\, i : \mathrm{int},\ i < i + 1$
- Une hypothèse est une proposition H que l'on admet seulement *dans une partie d'une preuve* (la *portée* de cette hypothèse). Si, dans cette portée, on montre la proposition B, on obtient, non une preuve de B, mais une preuve de $H \to B$.

```
{ Supposons H
   ... /* on raisonne */
   B  /* utilise l'hypothese H */
}
H → B [→i]
```

Conclusion: En cas de preuve assistée par ordinateur, l'utilisateur doit se méfier de sa modélisation. La machine ne vérifie que la démonstration et non son contexte. On essaiera de remplacer le maximum d'axiomes par des *définitions*, des *théorèmes*, ou des *hypothèses*.

Les difficultés

Nature de l'implication

Oubliez la “définition” de $A \rightarrow B$ comme $\sim A \vee B$.

- Cette définition ne rend pas compte du *sens* de $A \rightarrow B$: “Si on peut prouver A, alors on peut aussi prouver B”
- Toutes les logiques n'admettent pas l'équivalence entre $A \rightarrow B$ et $\sim A \vee B$

Nature de la contradiction

Une logique sert à caractériser les propositions qui sont *vraies* (ou, si l'on veut, les *séquents démontrables*). Un contexte dans lequel *toutes* les propositions seraient démontrables rendrait toute démonstration inutile.
On propose de définir la *contradiction*, notée $\perp$, comme *" tout est vrai"* (y compris $2 = 3$, $2 \neq 3$, "Socrate est immortel").
Cette "définition" rend compte du "principe d'explosion" : si on peut prouver $\Gamma \vdash \perp$ alors on peut prouver n'importe quel séquent $\Gamma \vdash A$.
La *négation* peut alors être comme une simple abréviation: $\sim A$ est une abréviation de $A \rightarrow \perp$.

Confusions à éviter (erreurs graves)

- Notations $\vdash$ et $\rightarrow$
- L'implication $\rightarrow$ et la conjonction "donc"
- La proposition $\perp$ et le booléen `false`, encore moins avec l'entier 0!
- Les connecteurs $\rightarrow$ et $\wedge$

Un exemple de preuve (en Français)

Théorème

$\sqrt{2}$ n'est pas rationnel.

Preuve:

On procède par l'absurde. Supposons que $\sqrt{2}$ soit rationnel, et écrivons-le p/q avec p et q premiers entre eux (fraction irréductible).
On a donc $(p/q)^2 = 2$, soit $p^2 = 2q^2$.
Donc p^2 est pair. Donc p est pair.
On définit donc p' par $p = 2p'$; on a donc $p^2 = 4p'^2$.
On a donc $4p'^2 = 2q^2$, donc $2p'^2 = q^2$.
Donc q^2 est pair. Donc q est pair.
On a supposé p et q premiers entre eux, et on a prouvé que p et q étaient tous les deux pairs: contradiction.
Donc l'hypothèse faite est fausse, et $\sqrt{2}$ n'est pas rationnel.

Un autre exemple de preuve (ébauche)

Une fonction OCaml

```
let rec insertion x l = match l with
  | []     -> [x]
  | y::ll -> if y>=x then x::l
             else y::(insertion x ll)
```

Théorème

Si x est un nombre entier et L une liste croissante d'entiers, alors insertion x L est une liste croissante d'entiers.

Preuve

La preuve naturelle se fait *par récurrence* sur la longueur de la liste; on prouve l'hérédité sous forme de *preuve par cas* qui suit la définition de la fonction.