Logique et Programmation Logique

Quentin Bramas

Largement inspiré par le cours de J.Narboux, C.Dubois et d'autres

RT-INOC 1A

September 12, 2017

Table des matières I

Logique et mathématiques

- Fondements des mathématiques
- Résultats négatifs

Logique et informatique

- La logique sert en programmation:
  - pour définir la sémantique des langages de programmation
  - pour la spécifications de programmes
  - pour l'analyse de programmes
  - pour la preuve de programmes
  - dans le cadre de la programmation logique (résolution de problèmes, systèmes experts, intelligence artificielle)
- La logique sert dans l'industrie; SAT et ses applications:
  - ordonnancement
  - synthèse et vérification de circuits
  - trafic aérien
- L'informatique sert en logique/mathématiques:
  - pour générer des preuves (démonstration automatique)
  - pour vérifier des preuves (assistants de preuve)

Objectifs du cours

- Comprendre la différence entre syntaxe et sémantique
- Comprendre la différence entre théorie et méta-théorie
- Découvrir la logique propositionnelle et la logique des prédicats
- Découvrir un autre style de programmation

Évaluation

- 2 contrôles continus en TD. 30 à 45 minutes chacun, documents autorisés. Vous serez prévenus des dates (normalement le 12 octobre et le 8 novembre). (20% + 20% de la note)
- Un exercice de TP noté (fichier à rendre plus réponses à quelques questions). (10% de la note)
- Partiel final. 2h, documents autorisés. (50% de la note)

Rattrapage : 2h, documents autorisés.

Bibliographie

R. David, K. Nour, C. Raffalli, Introduction à la logique, théorie de la démonstration.
J.P.Delahaye, Outils logiques pour l'intelligence artificielle.
R. Caferra, Logique pour l'informatique et pour l'intelligence artificielle.

La logique

La logique peut être considérée comme une partie des mathématiques, c'est une science dont l'objet d'étude est le raisonnement, les théorèmes, ...
La théorie des groupes consiste à étudier les groupes.
La logique consiste à étudier les théorèmes.
On va donc être amené à énoncer des théorèmes à propos de théorèmes.
Il faut distinguer les deux !

Logique et raisonnement

Tous les hommes sont mortels,
or Socrate est un homme
donc Socrate est mortel

C'est un Syllogisme

(Logique & Raisonnement - e-penser:
`https://www.youtube.com/watch?v=_yEHa6dIKgg`)

Logique et paradoxes

- "Je mens"
- "Cette phrase est fausse"

Origine philosophique.
Volonté de définir formellement le raisonnement.

Logique et paradoxes

L'ensemble de tous les ensembles
L'ensemble de tous les ensembles qui ne se contiennent pas eux-mêmes
Un barbier rase tous ceux qui ne se rasent pas eux-mêmes

Logique et paradoxes

Soit A l'ensemble des entiers naturels qui peuvent être définis en français par une phrase contenant moins de cent mots. A est fini.

n est le plus petit entier naturel qui ne peut pas être défini au moyen d'une phrase en français contenant moins de cent mots

Logique et mathématiques

Axiome du choix :
Le produit cartésien d'un nombre infini d'ensembles non-vides est non-vide.

Programme

- On peut voir un programme comme une suite d'instructions dans un langage de programmation quelconque.
- Tous les programmes (quelque soit le nombre de valeurs en entrée et en sortie) peuvent être vus comme ne prenant qu'un nombre en entrée et ne retournant qu'un nombre en sortie.
- Le nombre de programmes écrits avec moins de n symboles est fini.

$\Rightarrow$ On peut numéroter tous les programmes P_1, P_2, ...

Programme

Proposition

Il existe des fonctions de $\mathbb{N}$ dans $\mathbb{N}$ qui ne sont pas calculables, c'est-à-dire ne pouvant pas être programmées.

Une fonction calculable est aussi appelée *récursive*.

Programme

Proposition

Il existe un programme qui simule l'exécution de n'importe quel programme. $\exists Q$ tel que

$$\forall i, n \in \mathbb{N} \quad Q(i, n) = P_i(n)$$

Proposition

Il n'existe pas de programme qui vérifie la terminaison des autres programmes. $\nexists R$ tel que

$$\forall i, n \in \mathbb{N} \quad \begin{cases} R(i, n) & = 1 \text{ si } P_i(n) \text{ s'arrête} \\ R(i, n) & = 0 \text{ sinon} \end{cases}$$

Programme

Définition

Un ensemble est dit récursif s'il est possible de vérifier par un programme qu'un élément appartient ou non à cet ensemble.

$$Q(x) = 1 \text{ si } x \in E,\ Q(x) = 0 \text{ sinon}$$

Définition

Un ensemble est dit récursivement énumérable s'il existe un programme qui permet de lister tous les éléments de cet ensemble (possiblement sans jamais terminer). De manière équivalente :

$$Q(x) = 1 \text{ si } x \in E,\ Q(x) = \perp \text{ sinon}$$

L'induction

Un outil formel élégant très utile en informatique pour définir des ensembles et des fonctions, et démontrer des propriétés sur ces objets.

Définition inductive

La définition inductive d'un ensemble $X \subset E$ consiste en la donnée :

- de certains éléments de X (les éléments de base)
- de règles de construction d'éléments de X à partir d'éléments déjà connus de X (étapes inductives)
- + règle implicite : il n'y a pas d'autre moyen de définir des éléments de X

Formellement :

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 règles Reg :

$$E^{a(Reg)} \to E \quad \text{où } a(Reg) \text{ est l'arité de } Reg$$

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

(B) $B \subset X$

(I) $\forall Reg \in K$:

$$x_1, x_2 \ldots x_{a(Reg)} \in X \quad \Rightarrow \quad Reg(x_1, x_2 \ldots x_{a(Reg)}) \in X$$

Présentation à base de règles

On peut aussi présenter les règles de construction du schéma d'induction à l'aide de règles d'inférence :

$$\frac{\textit{premisses}}{\textit{conclusion}}$$

Les prémisses peuvent être omises dans certains cas.

Exemples de définitions inductives

- Définition inductive de $\mathbb{N}$:

  (B) $0 \in \mathbb{N}$ (I) si $n \in \mathbb{N}$ alors $n+1 \in \mathbb{N}$

  K est réduit à un singleton, son unique opération est l'opération *successeur*.
  On peut encore écrire :

  (B) $\frac{}{0}$, (I) $\frac{n \in \mathbb{N}}{n+1}$

- Définition inductive de A^*, ensemble de tous les mots formés sur l'alphabet A
  - (B1) $\epsilon \in A^*$ (le mot vide)
  - (B2) $A \subset A^*$
  - (I) si $m \in A^*$ et $n \in A^*$ alors $mn \in A^*$ (opération de concaténation)

- Définition inductive de L_A, ensemble des listes formées d'éléments de A
  - (B) $[] \in L_A$
  - (I) si $l \in L_A$ alors pour tout $a \in A, a :: l \in L_A$

- Définition inductive des expressions arithmétiques formées à partir d'identificateurs pris dans A et des opérateurs + et * :
  c'est la partie de $(A \cup \{+, *\})^*$ définie inductivement par :
  - (B) $A \subset E$
  - (I) si e et f sont dans E alors $e + f$ et $e * f$ sont aussi dans E

- Définition inductive du langage $D \subset \{(,)\}^*$ de Dyck des parenthésages bien formés.
  - (B) $\epsilon \in D$
  - (I1) si $x \in D$ alors $(x) \in D$
  - (I2) si $x \in D$ et $y \in D$ alors $xy \in D$

$(()())(())$ est un élément de D.
(1) $\epsilon \in D$ (par B)
(2) $() \in D$ (par I1 sur 1)
(3) $()() \in D$ (par I2 sur 2 et 2)
(4) $(()()) \in D$ (par I1 sur 3)
(5) $(()) \in D$ (par I1 sur 2)
$(()())(()) \in D$ (par I2 sur 4 et 5)

- Définition inductive de $S_{a,b} \subset \{a, b\}^*$
  - (B) $\epsilon \in S_{a,b}$
  - (I1) *si x est dans $S_{a,b}$, alors axb est dans $S_{a,b}$*
  - (I2) *si x est dans $S_{a,b}$, alors bxa est dans $S_{a,b}$*
  - (I3) *si x et y sont dans $S_{a,b}$ alors xy est aussi dans $S_{a,b}$*

Exemples ? Quel est cet ensemble ?

1ère caractérisation

Théorème

Soit $X \subset E$, défini inductivement par les éléments de base B et les règles K.

L'ensemble ainsi défini est

$$X = \bigcap_{Y \text{ vérifie (B) et (I)}} Y$$

(justifié par le fait que X est *le plus petit ensemble* vérifiant ...)

2ème caractérisation : approche itérative

Définition explicite

Théorème

Soit $X \subset E$, défini inductivement par les éléments de base B et les règles K. Tout élément de X peut s'obtenir à partir de la base en appliquant un nombre fini d'étapes inductives.

$\Rightarrow$ On peut représenter graphiquement un élément de X par un arbre : feuilles = éléments de base, nœuds = les règles de K appliquées à chaque étape.

Démonstration
L'ensemble des éléments qui s'obtiennent à partir de la base en appliquant un nombre fini d'étapes vérifie (B) et (I), donc il contient X.

Principe d'induction - Démonstration par induction

Définition du principe d'induction (structurelle)

Soit X un ensemble défini inductivement par B et K et soit $P(x)$ un prédicat exprimant une propriété de x, élément de X.
Si les conditions suivantes sont vérifiées :

B'' $P(b)$ est vraie pour chaque b de B

I'' si $P(x_1), P(x_2) \dots P(x_{a(Reg)})$ alors $P(Reg(x_1, x_2 \dots x_{a(Reg)}))$ pour chaque Reg de K

alors P(x) est vraie pour tout x de X.

Faire une démonstration par induction, c'est vérifier B'' et I''.

Démonstration du principe d'induction :

Posons $A = \{x \in X / P(x)\ vraie\}$. Montrons $A = X$.
De la même façon que précedemment, on montre que :

- A vérifie (B) et (I), alors X est inclus dans A.
- Et bien sûr, A est inclus dans X, d'où le résultat.

Exemples

Pour montrer une propriété $P(l)$ sur les listes de L_A, il suffira donc de montrer que :

- $P([])$ vraie
- pour tout l, si $P(l)$ vraie alors pour tout $a \in A$, $P(a :: l)$ vraie

Pour montrer une propriété $P(e)$ sur les expressions arithmétiques définies comme précédemment, il suffira donc de montrer que :

- $P(Idf)$ vraie, pour tout identificateur Idf de A
- pour tout e_1 et e_2 telles que $P(e_1)$ vraie et $P(e_2)$ alors $P(e_1 + e_2)$ est vraie
- pour tout e_1 et e_2 telles que $P(e_1)$ vraie et $P(e_2)$ alors $P(e_1 * e_2)$ est vraie

Exercice

Montrer que tout mot du langage de Dyck a autant de parenthèses ouvrantes que de parenthèses fermantes.