Logique : introduction

Sylvain Salvati

Outline

Organisation de l'UE

- 6 Cours : 1h30 toutes les deux semaines,
- TD/TP : en alternance 1h30 par semaine,
- évaluation : CTD, et 2 DS (DS1 5 novembre)

$$0.2 * CTD + 0.4 * DS1 + 0.4 * DS2$$

Contenu de l'UE

- Logique propositionnelle : syntaxe, sémantique, problème de satisfaisabilité (SAT),
- Logique des prédicats : syntaxe, sémantique,

Outline

De l'antiquité au XIX$^{\text{ème}}$ siècle

Intérêt ancien de la logique pour les philosophes. Ils cherchent à avoir une vie bonne, c'est-à-dire une vie en lien étroit avec la vérité. Aristote développe un système formel de syllogismes. Il faut noter que :

- les mathématiciens n'ont jamais utilisé le système d'Aristote dans leur travaux,
- depuis Aristote jusqu'au XIX$^{\text{ème}}$ siècle, il y a eu très peu d'apports de la philosophie à la logique.

Le XIX$^{\text{ème}}$ siècle

Mathématisation de la logique :

- Georges Boole, *The Laws of Thought* (1854) : invention de la logique propositionnelle,
- Gottlob Frege, *Begriffsschrift* (1879) : invention de la quantification (gestion des variables), de jugements logiques.

Ces travaux posent les bases de la logique telle qu'elle va se développer par la suite.

Début du XX$^{\text{ème}}$ siècle : la crise des fondements des mathématiques

Le paradoxe de Bertrand Russel en théorie des ensembles :

$$U = \{x \mid x \notin x\}$$

Début du XX$^{\text{ème}}$ siècle : la crise des fondements des mathématiques

Le paradoxe de Bertrand Russel en théorie des ensembles :

$$U = \{x \mid x \notin x\}$$

- $U \notin U$ implique, par définition de U que $U \in U$,

Début du XX$^{\text{ème}}$ siècle : la crise des fondements des mathématiques

Le paradoxe de Bertrand Russel en théorie des ensembles :

$$U = \{x \mid x \notin x\}$$

- $U \notin U$ implique, par définition de U que $U \in U$,
- si $U \in U$, alors, toujours par définition de U, $U \notin U$,

Début du XX$^{\text{ème}}$ siècle : la crise des fondements des mathématiques

Le paradoxe de Bertrand Russel en théorie des ensembles :

$$U = \{x \mid x \notin x\}$$

- $U \notin U$ implique, par définition de U que $U \in U$,
- si $U \in U$, alors, toujours par définition de U, $U \notin U$,
- les propositions $U \in U$ et $U \notin U$ sont équivalentes : on peut prouver tout et son contraire.

Début du XX$^{\text{ème}}$ siècle : la crise des fondements des mathématiques

Le paradoxe de Bertrand Russel en théorie des ensembles :

$$U = \{x \mid x \notin x\}$$

- $U \notin U$ implique, par définition de U que $U \in U$,
- si $U \in U$, alors, toujours par définition de U, $U \notin U$,
- les propositions $U \in U$ et $U \notin U$ sont équivalentes : on peut prouver tout et son contraire.

Ce type de paradoxe menaçait la cohérence de tout l'édifice des mathématiques, et par là celui de la science physique, fondement de l'industrialisation en cours et par suite de l'économie...

Début du XX$^{\text{ème}}$ siècle : le programme de Hilbert

En réaction au problème rencontré par les mathématiques, Hilbert propose un programme *finitiste* :

- utiliser un ensemble fini de principes de raisonnement suffisant pour démontrer toutes les mathématiques,
- démontrer que cet ensemble est cohérent en l'utilisant lui-même,

Début du XX$^{\text{ème}}$ siècle : le programme de Hilbert

En réaction au problème rencontré par les mathématiques, Hilbert propose un programme *finitiste* :

- utiliser un ensemble fini de principes de raisonnement suffisant pour démontrer toutes les mathématiques,
- démontrer que cet ensemble est cohérent en l'utilisant lui-même,

Kurt Gödel démontre que ce programme est voué à l'échec :

- **premier théorème d'incomplétude** (1931) : il existe une proposition φ de l'arithmétique qui n'est pas démontrable et dont la négation n'est pas démontrable,
- **deuxième théorème d'incomplétude** (1931) : tout système logique contenant l'arithmétique est soit contradictoire, soit il ne peut démontrer sa propre cohérence.

Début du XX$^{\text{ème}}$ siècle : le programme de Hilbert

En réaction au problème rencontré par les mathématiques, Hilbert propose un programme *finitiste* :

- utiliser un ensemble fini de principes de raisonnement suffisant pour démontrer toutes les mathématiques,
- démontrer que cet ensemble est cohérent en l'utilisant lui-même,

Kurt Gödel démontre que ce programme est voué à l'échec :

- **premier théorème d'incomplétude** (1931) : il existe une proposition φ de l'arithmétique qui n'est pas démontrable et dont la négation n'est pas démontrable,
- **deuxième théorème d'incomplétude** (1931) : tout système logique contenant l'arithmétique est soit contradictoire, soit il ne peut démontrer sa propre cohérence.

Néanmoins, Gödel a démontré en 1929 (en utilisant des arguments qui dépassent les capacités de l'arithmétique) que tout théorème de l'arithmétique est démontrable. Ce théorème est appelé **Théorème de complétude**.

Contenu calculatoire des théorème d'incomplétude

Les théorèmes d'incomplétude de Gödel sont basés sur les idées suivantes :

- les nombres permettent de représenter les énoncés logiques,
- l'arithmétique permet de mettre en relation les énoncés logiques suivants les règles de la logique,
- on peut représenter la logique elle-même dans l'arithmétique et la réduire à du calcul sur les nombres.

Contenu calculatoire des théorème d'incomplétude

Les théorèmes d'incomplétude de Gödel sont basés sur les idées suivantes :

- les nombres permettent de représenter les énoncés logiques,
- l'arithmétique permet de mettre en relation les énoncés logiques suivants les règles de la logique,
- on peut représenter la logique elle-même dans l'arithmétique et la réduire à du calcul sur les nombres.

NB : c'est précisément ce qui se passe dans les ordinateurs, toute information est ultimement réduite à une séquence de bits, manipulée comme des nombre (mais par des portes logiques !)...

Interprétation calculatoire des théorème d'incomplétude

Alan Turing (1936) extrait l'aspect lié au calcul du théorème de Gödel et en déduit plusieurs choses :

- un modèle de calcul qui formalise ce que sont les algorithmes : les machines de Turing,
- une construction d'universalité : il exhibe une machine qui permet d'exécuter les autres : il s'agit d'une machine programmable,
- exhibe un problème qui n'a pas de solution algorithmique : le problème de l'arrêt des machines de Turing (en corollaire, la logique est indécidable).

Interprétation calculatoire des théorème d'incomplétude

Alan Turing (1936) extrait l'aspect lié au calcul du théorème de Gödel et en déduit plusieurs choses :

- un modèle de calcul qui formalise ce que sont les algorithmes : les machines de Turing,
- une construction d'universalité : il exhibe une machine qui permet d'exécuter les autres : il s'agit d'une machine programmable,
- exhibe un problème qui n'a pas de solution algorithmique : le problème de l'arrêt des machines de Turing (en corollaire, la logique est indécidable).

C'est le début de l'informatique.

Outline

Avènement de l'informatique

- Les idées de Turing ont conduit à la construction des premiers ordinateurs après la seconde guerre mondiale.
- La logique a toujours accompagné les développements de l'informatique, tant au niveau matériel qu'au niveau logiciel.

Syntaxe, Sémantique : une distinction fondamentale

En logique, tout comme en informatique on distingue deux aspects des objets que l'on manipule :

Syntaxe	**Sémantique**
formules	modèles (graphes, entiers,. . .)
programmes	exécutions, effets, . . .

Syntaxe, Sémantique : une distinction fondamentale

En logique, tout comme en informatique on distingue deux aspects des objets que l'on manipule :

Syntaxe	**Sémantique**
formules	modèles (graphes, entiers,...)
programmes	exécutions, effets, ...

Pour une logique, on se pose classiquement les problèmes suivants reliant syntaxe et sémantique :

- **Vérification de modèle** : est-ce qu'une formule est vraie dans un modèle fixé ?
- **Satisfaisabilité** : existe-t-il un modèle dans lequel la formule est vraie ?
- **Tautologie** : est-ce qu'une formule est vraie dans tous les modèles ?
- **Démonstration** : qu'est-ce qu'une démonstration ?

Exemples d'applications de la logique en informatique : vérification de modèle

- **en programmation** : tests en tous genres (conditionnelles, fin de boucles, assertions,...),
- **recherche de données** : trouver de tous les tuples vérifiant une propriété donnée dans une *base de données*,
- **recherche de données** : plus généralement recherche d'information dans des structures de données :

```
[recette for recette in recettes
 if all(not sucre(ingredient)
        for ingredient recette.ingredients()) and
 any(vitamines(ingredient)
     for ingredient in recette.ingredients())]
```

Les recettes non sucrées qui contiennent des vitamines (en Python).

Exemples d'applications de la logique en informatique : satisfaisabilité

- l'*optimisation de programmes* et l'*élimination de code mort* peuvent avoir recourt à des routines de satisfaisabilité de formules pour fonctionner,
- *problèmes de résolution/d'optimisation sous contraintes* : recherche d'un modèle (une solution) optimal suivant certains critères. Un exemple est l'optimisation linéaire (c.f. ce cours avec SAT, ou en L3 ML – AMPL),
- *programmation par contraintes* : plus généralement, la résolution de problèmes combinatoires (problèmes de graphes, représentations des connaissances, jeux, ...) fait souvent appel à des programmes génériques de recherche de modèle (typiquement des SMT solveurs (solveurs modulo théorie), ou des solveurs de programmes linéaires à valeurs entières),
- *programmation synchrone* : génération de code de machines à états finis à partir de spécifications logiques (problème connu sous le nom de *synthèse de Church*).

Exemples d'applications de la logique en informatique : tautologie

- *démonstration automatique de théorèmes*,
- *vérifications et optimisation de programmes*,
- *certification de programmes* : propagation des contraintes logiques en suivant le flot d'exécution d'un programme (logique de Hoare), puis démonstration automatique de la spécification du programme.

Exemples d'applications de la logique en informatique : démonstrations

Souvent les démonstrations contiennent implicitement un algorithme. Par exemple démontrer la propriété suivante :

$$\forall a,\, b \in \mathbb{N},\, \exists d,\, r \in \mathbb{N},\, r < b \wedge a = d \times b + r$$

conduit à définir un algorithme de division euclidienne.
Plusieurs méthodes sont employées pour extraire le contenu calculatoire des démonstrations :

- la méthode *dialectica* de Gödel,
- la méthode de *réalisabilité* de Kleene,
- la correspondance de *Curry-Howard-de Bruijn* qui est au cœur de la théorie des types et du logiciel CoQ.

Outline

On The Unusual Effectiveness of Logic in Computer Science

THE BULLETIN OF SYMBOLIC LOGIC
Volume 7, Number 2, June 2001

ON THE UNUSUAL EFFECTIVENESS OF LOGIC IN COMPUTER SCIENCE

JOSEPH Y. HALPERN, ROBERT HARPER, NEIL IMMERMAN, PHOKION G. KOLAITIS, MOSHE Y. VARDI, AND VICTOR VIANU

On The Unusual Effectiveness of Logic in Computer Science

Cet article compare l'utilité de la logique pour l'informatique à celle des mathématiques pour la physique. Il cite entre autres applications :

- **la complexité descriptive** : les problèmes peuvent être classés en fonction de leur difficulté algorithmique. Cela donne lieu à des hiérarchies de classes de complexité. Celles-ci correspondent assez exactement à des problèmes de vérification de modèles finis,
- **les logiques epistémiques** : comment raisonner sur la propagation des croyances dans les systèmes multi-agents,
- **construction de processeurs** : à mesure que les processeurs sont devenus plus complexes, de nombreuses méthodes de vérification formelle basées sur la logique et les automates ont été déployées pour vérifier leur bon fonctionnement.