LOGIQUE
L3 Informatique

Salim Lardjane
Université Bretagne Sud

1. Introduction

Introduction

Les concepts logiques ont eu et continuent d'avoir une importance capitale en informatique.

Exemple : deux concepts «typiquement informatiques», celui de *règle de production* et celui de *langage formel* ont été inventés par des logiciens en 1921 (Post) et 1879 (Frege) respectivement.

A. Turing a proposé, en 1936, sa notion d'ordinateur abstrait comme partie de sa solution à un problème posé par D. Hilbert (un mathématicien).

Introduction

Quelques sujets étudiés en logique et de la plus grande importance pratique en informatique :

1. La NP-complétude du problème de la consistance (satisfaisabilité, validité) du calcul propositionnel classique (SAT), prouvée par Cook en 1971.

La très large utilisation du calcul propositionnel (vérification des systèmes critiques, systèmes dits intelligents, robotique, contraintes, etc.) donnent une idée de l'importance de ce résultat ;

2. L'étude des *structures finies* qui s'est développée de façon étroitement liée à l'informatique et dont les applications sont nombreuses en base de données, systèmes multi-agents, etc. ;

3. Les logiques dites non classiques (logiques modales, temporelles, polyvalentes, etc.) sont extrêmement utiles, entre autres en analyse des programmes et représentation des connaissances (en particulier des connaissances distribuées) .

Introduction

Bien que personne ne puisse dire quels seront les concepts et techniques utiles à un informaticien dans disons, 10, 20, 30 ou 40 ans, il est rès vraisemblable que l'être humain ne change pas beaucoup et qu'il devra faire face à un monde de plus en plus complexe.

Les concepts *abstraits* seront les plus utiles du point de vue de la compréhension aussi bien que de l'efficacité (considérer par exemple les multiples avantages - conception, codage, mise à jour - que présente l'utilisation de langages de très haut niveau).

Introduction

D'ores et déjà, quelques notions fondamentales de la logique sont utilisées par les ingénieurs informaticiens (parfois sous des noms ou des présentations différentes) . Par exemple : proposition, définition, sémantique, inférence, langage/métalangage, intention, extension, sous-classe de formules, modèle.

La notion de *proposition* dans son acception *intuitioniste*, par exemple, représente une intention ou une tâche ou un problème.

La notion de *définition*, étudié depuis des siècles par les logiciens, est intimement liée à celle de *spécification* et intervient dans les processus de construction des programmes (par exemple dans le *pliage* et *dépliage*) .

Introduction

Le rôle de la logique comme langage de spécification/vérification/développement logiciel est unanimement reconnu.

La notion de *sémantique* prépare directement à celle de compilation (l'assignation d'une sémantique étant une *traduction*).

L'inférence est une façon d'expliciter l'information. Elles est ainsi intimement liée à l'algorithmique.

Celle d'intention est liée, outre à spécification, à celle de modélisation de systèmes (et non simplement des programmes) où il faut modéliser aussi l'environnement (y compris les utilisateurs).

Introduction

La logique introduit des concepts très généraux, mais qui ont des applications pratiques extrêmement importantes.

Ainsi, la *méthode de Davis et Putnam* est utilisée par exemple en validation des systèmes.

La notion de *subsomption* est utilisée, entre autres, dans les langages de représentation des connaissances.

Elle est capitale dans ce qui est appelé des *ontologies* (vues comme un ensemble de concepts, des relations entre ces concepts et la façon de raisonner sur ces concepts) , très utilisées dans les taxonomies et définies parfois comme spécification explicite d'une vue abstraite et simplifiée d'un monde que l'on veut représenter.

Par ailleurs, les *bases de données* sont l'un des sujets traditionnels d'application de l'informatique.

On peut voir une base de données relationnelle comme une *structure du premier ordre* et la logique du premier ordre comme un langage de requêtes.

La recherche de langages de requêtes plus puissants montre le besoin pratique de logiques avec un plus grand pouvoir d'expression.

Quelques applications actuelles *concrètes* de la logique :

1. Les systèmes multi-agents, avec des applications aussi importantes que la robotique, internet, etc.

Ces systèmes peuvent avoir des configurations extrêmement complexes et il devient nécessaire de disposer de techniques (formelles) pour raisonner dans (et sur) ces systèmes.

Les notions et outils intervenant dans leur étude comprennent la logique temporelle, les logiques pour la représentation des connaissances, la déduction, l'abduction (c'est-à-dire découverte d'hypothèses qui expliquent une observation ou permettent d'aboutir à une conclusion) ainsi que la notion de *preuve* qui peut servir à convaincre un agent (en particulier un agent humain) de la pertinence d'une information, etc.

2. Les logiques *modales* (logiques temporelles, dynamiques, etc.) sont utilisées industriellement pour capturer des comportements réactifs, des systèmes concurrents, etc.

3. Un nombre croissant d'ingénieurs informaticiens s'orientent vers l'économie et les finances.

Dans ces disciplines la modélisation des acteurs et leurs connaissances (croyances) sur les autres acteurs (du marché, de la société) est capitale pour la compréhension et la décision. Des assertions telles que «X connaît (croit) que Y connaît (croit) que Z connaît (croit) que . . . » peuvent être traitées formellement par les *logiques de la connaissance* (de la croyance).

4. La logique est très importante pour le *traitement automatique du langage naturel*, dont les applications sont très nombreuses (par exemple sur internet, en recherche d'information, dans les systèmes de question/réponse, etc.).

Quelques applications plus directement liées aux problèmes techniques dans la programmation de systèmes informatiques :

1. On peut *prouver* qu'un programme n'est pas correct, mais on ne peut pas en général prouver qu'un programme est correct à l'aide d'exemples.

L'importance de la détection d'erreurs éventuelles dans les pro grammes (ou la conception des circuits) peut être mise en évidence par deux exemples particulièrement spectaculaires séparés par plus de trente ans.

La sonde vers Venus, Mariner 1 (juin 1962), rata sa cible à cause d'une erreur dans le programme de l'ordinateur de bord (une instruction syntaxiquement correcte avait une signification complètement différente de celle que l'on avait voulu écrire et qui

était syntaxiquement proche de l'instruction écrite).

Des techniques logiques ont été développées pour prouver la *correction* des programmes, pour détecter des erreurs de programmation, pour construire des programmes satisfaisant une certaine spécification.

Ces techniques peuvent, dans une bonne mesure, être automatisées.

Pour raisonner mécaniquement sur un programme il faut : une sémantique formelle, une théorie logique formelle et un démonstrateur automatique de théorèmes pour cette théorie.

On parle de plus en plus de *programmes certifiés corrects*.

L'importance pratique de cette notion ne saurait être surestimée (vu l'importance de l'informatique dans les avions ou dans les hôpitaux, par exemple) .

Introduction

Dans les années 1990, des erreurs ont été découvertes dans le fonctionnement d'une puce électronique alors qu'elle était déjà sur le marché. Elle était vendue par le plus grand fabriquant de ces composants. Cette erreur se mettait en évidence sur certaines données rares.

Cette erreur a donné des arguments irréfutables aux défenseurs de la nécessité de remplacer la simulation avec des tests exhaustifs par une *vérification formelle*, afin de prouver la correction d'une conception avant la fabrication.

Theorem proving et *model checking* sont deux techniques permettant de *certifier* la correction des puces en question.

Dans ces deux approches la logique (classique et non classique) joue un rôle-clé.

Introduction

Quelques succès retentissants dans le domaine du logiciel aussi bien que dans celui du matériel ont prouvé qu'il s'agit d'approches applicables aux problèmes réels.

1. La *vérification formelle*, un domaine considéré longtemps comme théorique, a aujourd'hui des retombées économiques énormes.

2. La *programmation logique* est l'utilisation de la logique comme langage de programmation.

Le langage Prolog, ainsi que d'autres qui on ont suivi, en particulier les langages de programmation logique avec contraintes, sont une conséquence naturelle des notions et méthodes étudiées en logique.

3. Certains paradoxes logiques, notamment celui nommé paradoxe de Russell sont intimement liés au *problème de l'arrêt*, c'est-à-dire l'existence d'un algorithme pouvant déterminer pour tout programme s'il s'arrêtera ou pas.

L'impossibilité de l'existence d'un tel algorithme est bien connue.

4. Depuis quelques années, il y a un essor de systèmes informatiques ayant un comportement intelligent.

Les principes de conception de ces systèmes sont étroitement liés à la logique (classique et non classique).