

INFO-F-302, Cours d'Informatique
Fondamentale
Logique pour l'Informatique

Emmanuel Filiot
Département d'Informatique
Faculté des Sciences
Université Libre de Bruxelles

Année académique 2011-2012
Basé sur le cours *Logique Informatique* du Pr. J.-F. Raskin

Organisation pratique du cours

Références

- Logic in Computer Science : Modeling and Reasoning about Systems, M. R. A. Huth and M. Ryan, Cambridge University Press, 1999.
- Mathematical Logic for Computer Science, M. Ben-Ari, Prentice Hall, 1993.

Page web du cours www.ulb.ac.be/di/info-f-302/

Matériel Les slides utilisés lors des cours seront disponibles sur la page web du cours au fur et à mesure.

Travaux pratiques Assistant : Dr. Alexander Heussner

Contact

- email : efiliot@ulb.ac.be
- tel : 650 64 64
- http : www.ulb.ac.be/di/ssd/filiot

INTRODUCTION

Qu'est-ce que la logique ?

Définition du Littré

“Science qui a pour objet les procédés du raisonnement”.

Définition du Trésor

“Science relative aux processus de la pensée rationnelle (induction, déduction, hypothèse p. ex.) et à la formulation discursive des vérités.”.

Définition du Larousse

“Science du raisonnement en lui-même, abstraction faite de la matière à laquelle il s'applique et de tout processus psychologique.”.

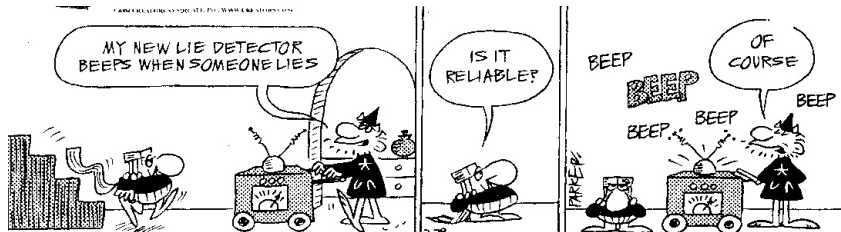
Origines de la logique : Grèce Antique

- enseignement du discours (*logos* en grecque) et de la rhétorique.
Aristote (-384, -322) .
- Formaliser les règles de *déduction* : les règles de la logique doivent permettre de dériver des énoncés vrais à partir de *prémisses*.
- Exemple : syllogisme
 - 1 Tous les hommes sont mortels.
 - 2 Or X est un homme.
 - 3 Donc X est mortel.
- Si les prémisses sont vrais (1 et 2), les règles de la logique assurent que 3 est vrai, peu importe la valeur de X .
- En notation ensembliste, cela s'exprime ainsi : si $H \subseteq M$ et $X \in H$, alors $X \in M$.

Fausse vérités et paradoxes

- l'utilisation du langage naturel comme notation est imprécise et peut mener à des énoncés faux ...
 - ① Tout ce qui est rare est cher
 - ② Or une chose pas chère, c'est rare
 - ③ Donc une chose pas chère, c'est cher.
- ... ou à des paradoxes
 - ▶ Cette phrase est fausse. Problème de *l'auto référence*.
 - ▶ Un crocodile s'empare d'un bébé crocodile et propose à sa mère de récupérer son bébé si elle devine ce qu'il va en faire, ce à quoi elle répond "tu vas le dévorer". Que va donc faire le crocodile ?
 - ▶ Le barbier rase tous les hommes qui ne se rasent pas eux-mêmes. Qui rase le barbier ?

Un petit dernier ...



Logique Moderne

- 19^e siècle : la logique devient un outil scientifique (Boole, Hilbert, Gödel, Church, Turing ...).
- Formaliser les mathématiques comme un langage. Formaliser le concept de *démonstration*.
- Définir des systèmes logiques tels que
 - ▶ tout ce qui est prouvable est valide (*correction*).
 - ▶ tout ce qui est valide est prouvable (*complétude*).
- Gödel (1931) : dans un système logique qui permet d'exprimer l'arithmétique, il existe des énoncés valides qui ne sont pas prouvables (*théorème d'incomplétude*).

Logique en Informatique

Les logiques sont utilisées en informatique pour

Modéliser de manière *formelle* des “objets” rencontrés par les informaticiens ;

Ex : Bases de données, bases de connaissances, pré-post conditions d'une procédure, etc.

Raisonner Après une phase de modélisation, l'informaticien doit être capable de se servir du modèle, de l'analyser de manière *rigoureuse*, voire d'automatiser cette analyse.

Ex : validation d'un modèle de données, prise de décision à partir de faits et d'une base de connaissances, preuve de correction d'une procédure/d'un programme ;

Exemple

Considérons la situation décrite par les affirmations suivantes :

- 1 Si le train arrive en retard **et** il n'y a pas de taxis à la gare **alors** l'invité arrive en retard.
- 2 L'invité n'est pas en retard.
- 3 Le train est arrivé en retard.

Et la déduction suivante : il y avait des taxis à la gare.

Exemple

Considérons la situation décrite par les affirmations suivantes :

- 1 Si le train arrive en retard **et** il n'y a pas de taxis à la gare **alors** l'invité arrive en retard.
- 2 L'invité n'est pas en retard.
- 3 Le train est arrivé en retard.

Et la déduction suivante : il y avait des taxis à la gare.

Question

Pourquoi peut-on déduire qu'il y avait des taxis à la gare ?

Premièrement, si on met l'affirmation 1 et l'affirmation 3 ensemble, on peut affirmer que s'il n'y avait pas eu de taxis à la gare, alors l'invité serait arrivé en retard. D'après l'affirmation 2, l'invité n'est pas arrivé en retard. Donc il y avait des taxis à la gare.

Autre Exemple

Considérons un autre exemple :

- 1 Si il pleut **et** l'invité a oublié son parapluie **alors** l'invité est trempé.
- 2 L'invité n'est pas trempé.
- 3 Il pleut.

Et la déduction suivante : l'invité n'a pas oublié son parapluie.

Autre Exemple

Considérons un autre exemple :

- 1 Si il pleut **et** l'invité a oublié son parapluie **alors** l'invité est trempé.
- 2 L'invité n'est pas trempé.
- 3 Il pleut.

Et la déduction suivante : l'invité n'a pas oublié son parapluie.

Question

Pourquoi peut-on déduire que l'invité n'a pas oublié son parapluie ?

Premièrement, si on met l'affirmation 1 et l'affirmation 3 ensemble, on peut affirmer que si l'invité avait oublié son parapluie, alors il serait trempé. D'après l'affirmation 2, l'invité n'est pas trempé. Donc l'invité n'a pas oublié son parapluie.

Remarque

La deuxième démonstration suit la même **structure logique** que la première démonstration. Il suffit de remplacer les fragments de phrase

	Exemple du train	Exemple du parapluie
comme suit :	Le train est arrivé en retard	Il pleut
	Il y a des taxis à la gare	L'invité a son parapluie
	L'invité est en retard	L'invité est mouillé.

Formalisation Logique

Exemple du train	Exemple du parapluie	Proposition
Le train est arrivé en retard	Il pleut	p
Il y a des taxis à la gare	L'invité a son parapluie	q
L'invité est en retard	L'invité est mouillé	r

Démonstration

- 1 Hypothèse : si p et non q , alors r
- 2 Hypothèse : p
- 3 Hypothèse : non r
- 4 Dédution : si non q alors r
- 5 Dédution : comme non r , alors q .

Formalisation logique

Alors nous pouvons *formaliser* les situations décrites dans ces deux premiers exemples par la *formule logique* suivante :

$$((p \wedge \neg q) \rightarrow r) \wedge \neg r \wedge p$$

Et nous pouvons *formaliser* la déduction par :

$$((p \wedge \neg q) \rightarrow r) \wedge \neg r \wedge p \models q$$

Où “ \models ” se lit : “ q est une conséquence logique de $((p \wedge \neg q) \rightarrow r) \wedge \neg r \wedge p$ ”.

Conditions Booléennes dans les langages de programmation

Considérons le test suivant :

if $(x \geq 3 \text{ and } z \leq 3) \text{ or } (y \leq 2 \text{ and } z \leq 3)$ **then** ...

Il peut-être remplacé par

if $(x \geq 3 \text{ or } y \leq 2) \text{ and } z \leq 3$ **then** ...

Car $(p \vee q) \wedge r$ est logiquement équivalent à $(p \wedge r) \vee (q \wedge r)$.

Logique Propositionnelle

- les propositions sont représentées par des *variables propositionnelles* p, q, r, \dots
- les formules sont des suites de symboles construites à partir des variables propositionnelles et des connecteurs Booléens $\wedge, \vee, \neg, \rightarrow, \dots$. On parlera de *syntaxe*.
- Exemple : $(p \wedge q) \rightarrow (p \vee q)$.
- La *sémantique* des formules est définie en interprétant les variables propositionnelles par des valeurs Booléennes vrai (\top) et faux (\perp), et en interprétant les connecteurs Booléens comme des fonctions Booléennes.
- Une formule est valide si elle est vraie pour toute interprétation de ses variables. Ex : $p \rightarrow p, p \vee \neg p, p \rightarrow (q \rightarrow (p \wedge q))$.
- Questions : comment décider si une formule est valide ? comment faire le lien entre la syntaxe et la sémantique (prouvabilité vs validité) ?

Vers plus d'expressivité

Phrase

Chaque étudiant est plus jeune qu'un professeur.

Nous pouvons identifier cette phrase avec une variable propositionnelle p .
La logique propositionnelle n'est pas assez fine pour modéliser cette situation de manière précise

Vers plus d'expressivité

Phrase

Chaque étudiant est plus jeune qu'un professeur.

Nous pouvons identifier cette phrase avec une variable propositionnelle p . La logique propositionnelle n'est pas assez fine pour modéliser cette situation de manière précise

Décomposition

Quels sont les éléments essentiels de cette phrase ?

- 1 être un étudiant
- 2 être un professeur
- 3 être plus jeune que quelqu'un d'autre

Vers plus d'expressivité : les prédicats

Phrase

Chaque étudiant est plus jeune qu'un professeur.

- 1 être un étudiant
- 2 être un professeur
- 3 être plus jeune que quelqu'un d'autre

Ces qualités sont représentées par des *prédicats*.

- 1 $E(\textit{alice})$: alice est une étudiante
- 2 $P(\textit{john})$: john est un professeur
- 3 $J(\textit{alice}, \textit{john})$: alice est plus jeune que john

Vers plus d'expressivité : les variables

De manière générale, on utilise des variables x, y, z, \dots pour abstraire des valeurs concrètes :

- 1 $E(x)$: x est un étudiant
- 2 $P(y)$: y est un professeur
- 3 $J(x, y)$: x est plus jeune que y

Vers plus d'expressivité : les quantificateurs

Phrase

Chaque étudiant est plus jeune qu'**un** professeur.

On a besoin de quantificateurs :

universel \forall , pour tout

existentiel \exists , il existe

Vers plus d'expressivité : la logique du premier ordre

Phrase

Chaque étudiant est plus jeune qu'**un** professeur.

Voici comment représenter cette phrase en logique du premier ordre (appelée aussi logique des prédicats) :

$$\forall x : (E(x) \rightarrow (\exists y : P(y) \wedge J(x, y)))$$

qui se lit : “pour tout x , si x est un étudiant, alors il existe y tel que y est un professeur et x est plus jeune que y .”

Logique du premier ordre : autre exemple

- Considérons les prédicats suivants :

- ▶ $O(x)$: x est un oiseau
- ▶ $V(x)$: x sait voler

- et les phrases suivantes :

- ▶ il n'est pas vrai que tous les oiseaux savent voler

$$\neg(\forall x : (O(x) \rightarrow V(x)))$$

- ▶ de manière équivalente, il existe un oiseau qui ne sait pas voler :

$$\exists x : O(x) \wedge \neg V(x)$$

La logique du premier ordre : un dernier exemple

Considérons les règles suivantes :

- ① les parents d'un enfant sont des ancêtres de cet enfant ;
- ② les ancêtres d'une personne qui est l'ancêtre d'une deuxième personne sont également ancêtres de cette deuxième personne ;
- ③ deux personnes sont d'une même famille si et seulement si elles ont un ancêtre commun.

La logique du premier ordre – un dernier exemple

Formalisation de la règle *SpecApparente* :

- 1 $\forall x, y : \text{Parent}(x, y) \rightarrow \text{Ancetre}(x, y)$
- 2 $\forall x, y : \exists z : \text{Ancetre}(x, z) \wedge \text{Ancetre}(z, y) \rightarrow \text{Ancetre}(x, y)$
- 3 $\forall x, y : \text{Apparente}(x, y) \leftrightarrow \exists z : \text{Ancetre}(z, x) \wedge \text{Ancetre}(z, y)$

La logique du premier ordre – un dernier exemple

La formalisation de cet exemple permet maintenant de faire des “calculs” pour raisonner sur cette règle. Par exemple, il est possible de poser de manière précise des questions comme :

- est-ce qu'un parent x et son enfant y sont apparentés ?

$$\text{SpecApparente} \models? \forall x, y : \text{Parent}(x, y) \rightarrow \text{Apparente}(x, y)$$

- est-ce que la relation *Apparente* est symétrique ?

$$\text{SpecApparente} \models? \forall x, y : \text{Apparente}(x, y) \rightarrow \text{Apparente}(y, x)$$

La logique du premier ordre – un dernier exemple

La formalisation de cet exemple permet maintenant de faire des “calculs” pour raisonner sur cette règle. Par exemple, il est possible de poser de manière précise des questions comme :

- est-ce qu'un parent x et son enfant y sont apparentés ?

$$\text{SpecApparente} \models? \forall x, y : \text{Parent}(x, y) \rightarrow \text{Apparente}(x, y)$$

- est-ce que la relation *Apparente* est symétrique ?

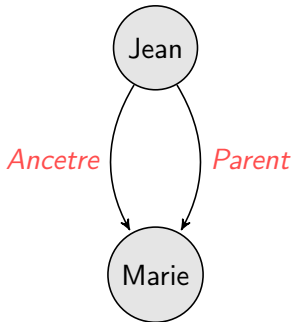
$$\text{SpecApparente} \models? \forall x, y : \text{Apparente}(x, y) \rightarrow \text{Apparente}(y, x)$$

Note

Répondre à ces questions de manière formelle revient à faire une preuve qui établit la propriété ou à construire un contre-exemple.

La logique du premier ordre – un dernier exemple

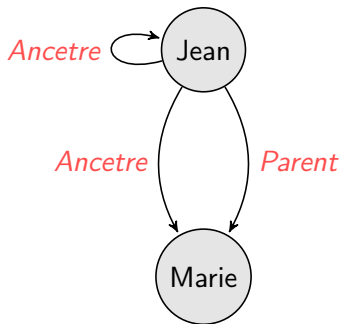
Considérons le situation suivante :



On n'a pas *Apparente*(*Jean*, *Marie*), pourquoi ?

La logique du premier ordre – un dernier exemple

Considérons le situation suivante :



On n'a pas *Apparente*(Jean, Marie), pourquoi ? Il faut ajouter l'hypothèse :

$$\forall x : \text{Ancetre}(x, x)$$

Donc une formalisation ne représente pas toujours ce que l'on veut vraiment, et on est parfois amené à l'améliorer.

Programmation logique

- Supposons que nous ayons une formule ϕ capable d'exprimer que pour tout tableau d'entiers A , il existe un tableau d'entier A' tel que A' est une permutation de A et A' est ordonné.
- Supposons que nous ayons un *prouveur automatique* capable de prouver ϕ
- Supposons enfin que ce *prouveur automatique* établisse des preuves constructives, i.e. qu'étant donné un tableau A , il sera effectivement capable de nous convaincre que A' existe en le construisant.
- Alors la formule ϕ n'est rien d'autre qu'un programme pour trier des tableaux d'entiers !

Programmation logique

- Supposons que nous ayons une formule ϕ capable d'exprimer que pour tout tableau d'entiers A , il existe un tableau d'entier A' tel que A' est une permutation de A et A' est ordonné.
- Supposons que nous ayons un *prouveur automatique* capable de prouver ϕ
- Supposons enfin que ce *prouveur automatique* établisse des preuves constructives, i.e. qu'étant donné un tableau A , il sera effectivement capable de nous convaincre que A' existe en le construisant.
- Alors la formule ϕ n'est rien d'autre qu'un programme pour trier des tableaux d'entiers !

C'est le principe de la programmation logique :

- au lieu d'écrire un programme en plusieurs étapes, on écrit une formule qui décrit la relation entre les entrées et les sorties.
- on laisse le prouveur automatique chercher les réponses
- programmation *descriptive* et non *procédurale*

Programmation logique – exemple en Prolog

Il y a un langage qui est basé sur cette idée : Prolog.

Exemple de programme Prolog

```
ancetre(x, x).  
ancetre(x, y) :- parent(x, z), ancetre(z, y).  
parent(paul, martin).  
parent(paul, marie).  
parent(marie, luc).  
?- ancetre(x, luc).
```

Autres exemples d'application de la logique en informatique

- design de circuits numériques
- vérification *hardware* et *software*, *model-checking*. Logiques temporelles.
- bases de données : SQL.
- théorie de la complexité. Classes *P* et *NP*.
- ...

Objectifs du cours

- vous familiariser avec les concepts de base de la logique
- appliquer la logique pour modéliser et résoudre certains problèmes
- avoir des notions de preuve de programmes.

Plan du cours

- 1 la logique propositionnelle ;
- 2 la logique du premier ordre ;
- 3 introduction à la preuve de programmes.