Eléments de logique pour l'informatique

Uli Fahrenberg
`uli@lmf.cnrs.fr`

d'après Christine Paulin

Département Informatique, Faculté des Sciences d'Orsay, Université Paris-Saclay

Licence Informatique - LDD Informatique, Mathématiques

2025–26

Introduction au cours de logique

1. Motivations
2. Organisation du cours
3. Exemples d'usage de la logique

La logique, chemin vers la vérité

- *Logique* vient du grec logos (raison, langage, raisonnement).
- La logique manipule des *énoncés* (phrases dont le sens est vrai ou faux).
  - “Tous les moutons ont 5 pattes”
  - “Aucun étudiant ne joue sur son téléphone pendant le cours de logique”
  - “Les trains ne passent pas à un passage à niveau ouvert”
- “La couleur du cheval blanc d’Henri IV” *n’est pas un énoncé*
- Importance du langage pour préciser de quoi on parle (souvent implicite)
- Un énoncé peut être vrai ou faux suivant la manière dont on interprète les mots
- La logique est une manière *scientifique* d’étudier la notion de vérité

Combinaison d'énoncés

- L'arithmétique étudie les propriétés des nombres : $0, 1, 2, 3, \dots$
- La logique (classique) s'intéresse à un espace beaucoup plus simple : *vrai*/*faux*
- Les connecteurs logiques sont des *opérations* qui permettent de former de nouveaux énoncés
  - la négation d'un énoncé E : la négation d'un énoncé E est vraie si et seulement si l'énoncé E est faux
  - la conjonction : l'énoncé $E_1 \texttt{ et } E_2$ est vrai si et seulement si les énoncés E_1 et E_2 sont simultanément vrais
  - la disjonction : l'énoncé $E_1 \texttt{ ou } E_2$ est vrai si et seulement si l'un des deux énoncés E_1 et E_2 est vrai (ou les deux)
  - ...

Enoncé paramétré et quantification

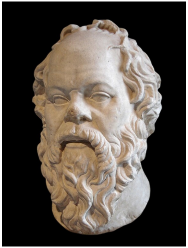
- Un énoncé logique *paramétré* représente une propriété (vraie ou fausse) d'un "*objet*" quelconque
- exemple : propriété pour un étudiant de *valider le cours de logique*
- "*valider le cours de logique*" sera vrai ou faux pour chaque étudiant considéré
  - *Toto valide le cours de logique*
- on peut former de nouveaux énoncés comme :
  - tous les objets vérifient la propriété,
  - il existe (au moins) un objet qui vérifie la propriété,
  - il existe précisemment 42 objets qui vérifient la propriété.

La logique, chemin vers la vérité

- Le raisonnement est un cheminement (une déduction, une preuve) qui permet de relier entre eux des énoncés : on distingue des *hypothèses*, et une *conclusion*.
- On veut s'assurer que dans toute situation où les *hypothèses sont vraies*, il en est de même de la conclusion.
- Un raisonnement suit des *règles* logiques précises.
- Le raisonnement est un procédé suffisament élémentaire pour *convaincre*
  - modus ponens, preuve par l'absurde, preuve par récurrence ...
- Montrer qu'un énoncé est vrai est, en général, indécidable (il n'y a pas de programme qui répond à cette question).
- Vérifier qu'un raisonnement suit bien les règles du jeu peut, le plus souvent, s'effectuer mécaniquement.
- Certains raisonnements sont adaptés à l'humain, d'autres aux machines.

Exemples de raisonnement

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



Tous les hommes sont mortels
or l'âne Francis n'est pas un homme
donc l'âne Francis est immortel.

Chaos de la logique

Dialogue entre le Logicien et le Vieux Monsieur, extrait de Rhinoceros (Ionesco)

- Je vais vous expliquer le syllogisme.
- Ah ! oui, le syllogisme !
- Le syllogisme comprend la proposition principale, la secondaire et la conclusion.
- Quelle conclusion ?
- Voici donc un syllogisme exemplaire. Le chat a quatre pattes. Isidore et Fricot ont chacun quatre pattes. Donc Isidore et Fricot sont chats.
- Mon chien aussi a quatre pattes.
- Alors, c'est un chat.
- Donc, logiquement, mon chien serait un chat.
- Logiquement, oui. Mais le contraire est aussi vrai.
- C'est très beau, la logique.
- A condition de ne pas en abuser...

Démonstration

- Démontrer c'est apporter une *évidence* du fait que quelque chose est vrai
- Plusieurs sortes de preuve
  (`http://www.pion.ch/Logic/preuves.html`)
  - par l'exemple : *On démontre le cas n = 2 qui contient la plupart des idées de la preuve générale.*
  - par généralisation : *Ça marche pour 17, donc ça marche pour tout nombre réel.*
  - par fin de cours : *Vue l'heure, je laisserai la preuve de ce théorème en exercice.*
  - par probabilité : *Une recherche longue et minutieuse n'a mis à jour aucun contre-exemple.*
  - par tautologie : *Le théorème est vrai car le théorème est vrai.*
- Dans ce cours
  - des méthodes rigoureuses
  - transposables sur ordinateur

Principes

- La logique *formalise un langage*, en *définit le sens* et propose *des règles du jeu* qui permettent de se convaincre de la vérité d'une argumentation ou au contraire de la réfuter.
- *Questions logiques :*
  - avec quels *objets* joue-t-on ? que représentent-ils ? quelles règles ?
  - les règles du jeu sont-elles trop *laxistes* (on déduit des choses fausses, on dit que le système est incohérent)
  - les règles du jeu sont-elles trop *strictes* (on n'arrive pas à prouver quelque chose qui est pourtant correct, on dit que le système est incomplet)
  - peut-on changer les règles du jeu ? changer de style ?
- *Questions informatiques :*
  - un ordinateur peut-il *raisonner* ?
  - est-ce qu'il existe un *algorithme* pour dire qu'une formule est vraie, est fausse ?
  - étant données des hypothèses et une conclusion, peut-on reconstruire une déduction ?

Fondement des mathématiques

- Le questionnement sur les fondements des mathématiques date du début du 20ème siècle avec la théorie des ensembles
- Quelques logiciens importants : Tarski, Russell, Hilbert, Gödel …
- Des surprises :
  - le raisonnement ne peut pas se ramener au calcul
  - impact du langage sur la cohérence : paradoxe de Russell
    - tout est ensemble, $x \in X$, compréhension $\{x|P(x)\}$
    - $X = \{x|x \notin x\}$, $X \in X$ si et seulement si $X \notin X$
    - il n'y a pas d'ensemble de tous les ensembles/il faut structurer les ensembles avec des *types*
  - théorème d'incomplétude de Gödel :
    - soit un système de déduction dans lequel on peut raisonner sur les entiers
    - il existe une formule C telle que on ne peut démontrer ni C, ni la négation de C
    - aucun système de démonstration "puissant" (ex. arithmétique, théorie des ensembles) ne capture toute la vérité…

Meta-mathématique

- *Logique mathématique* : les énoncés mathématiques, le raisonnement sont l'objet de l'étude comme les nombres en algèbre, les fonctions en analyse, les espaces vectoriels . . .
- On raisonne sur ces objets, on établit des théorèmes :
  - *deux niveaux* différents qu'il ne faut pas confondre.
- Analogie informatique : programmes qui manipulent d'autres programmes (*les compilateurs*)

Logique et informatique

- Lien entre calcul booléen (vrai/faux) et circuits logiques (0/1)
- Une démarche analogue : machine universelle reposant sur un nombre limité d'opérations ; liens entre syntaxe et sémantique.
- Intelligence artificielle : munir un ordinateur qui sait calculer de capacité de raisonnement nécessite de transformer le raisonnement en calcul (méthodes symboliques liées à la logique, enjeu de l'explication des méthodes statistiques).
- Outil de modélisation : contraintes dans les bases de données, développement de programmes, web sémantique, apprentissage symbolique, ... (logique au service de l'informatique)
- Structures informatiques pour représenter des propriétés logiques, outils pour les manipuler (informatique au service de la logique).

Introduction au cours de logique

1. Motivations
2. Organisation du cours
3. Exemples d'usage de la logique

Compétences logiques attendues

- Prérequis
  - les bases du calcul booléen
  - compréhension des formules et preuves mathématiques élémentaires
- Connaître et savoir manipuler le langage de la logique du premier ordre
  - savoir traduire des formules logiques en langue naturelle
  - savoir modéliser un problème en termes logiques
  - reconnaître certaines catégories de formules
  - savoir donner un sens aux formules de la logique
- Savoir mettre en œuvre plusieurs notions de démonstration
- Connaître les principales limites des méthodes d'un point de vue calcul

LDD Connaître et comprendre quelques résultats clés de la logique : complétude, compacité, démonstrations…

Compétences générales attendues

- Savoir présenter un raisonnement scientifiquement correct
- Apprendre un nouveau langage formel
- Distinguer syntaxe et sémantique
- Savoir manipuler des algorithmes sur des objets symboliques
- Méthodologie : mise en pratique d'objets mathématiques utiles en informatique

Plan du cours

1. Maîtriser le langage logique
2. Donner du sens aux formules
3. Manipuler les formules de la logique
4. Automatiser les démonstrations

Bibliographie

Serenella Cerrito.
Logique pour l'Informatique : une introduction à la déduction automatique.
Vuibert Publisher Co, 2008.

Stéphane Devismes, Pascal Lafourcade, and Michel Lévy.
Informatique théorique : Logique et démonstration automatique, Introduction à la logique propositionnelle et à la logique du premier ordre.
Ellipses, 2012.

Robert Cori and Daniel Lascar.
Logique Mathématique.
Axiomes. Masson, 1993.

René David, Karim Nour, and Christophe Raffalli.
Introduction à la Logique, Théorie de la démonstration.
Dunod, 2001.

Gilles Dowek.
La logique.
Le Pommier, 2015.

Gilles Dowek.
Les démonstrations et les algorithmes.
Les éditions de l'Ecole Polytechnique, 2010.

Pierre Le Barbenchon, Sophie Pinchinat, and François Schwarzentruber.
Logique : fondements et applications.
Dunod, 2022.

Informations sur le cours

- Espace ecampus des formations
  - emplois du temps, groupes, examens. . .
- Espace ecampus du cours *Eléments de logique pour l'informatique*
  - Espace partagé entre les parcours licence et LDD, classique et apprentissage
  - Notes de cours disponibles (distribuées)
  - Exercices pour le contrôle continu (à venir)
  - Annales des partiels-examens des années précédentes (beaucoup de corrigés)

Déroulé du cours

- Feuille de présence cours-TD
- Deux cours cette semaine
- Démarrage des TD la semaine prochaine
- LDD IM+magistère : complément de cours + exercices
- Evaluation
  - Partiel (40%) + Examen final (50%)
    - un exercice sous forme *QCM* (contrôle notions élémentaires)
  - CC (10%) exercices en ligne, devoir
- Des tests à compléter sur ecampus (respecter les dates)
  - Enquête rentrée sur la page d'accueil
  - Tests (calcul propositionnel) dans la section *Maîtriser le langage logique*

L'équipe

- Uli Fahrenberg
  - responsable cours, chargé TD groupe 6
  - nouveau prof. à l'Université Paris-Saclay, anciennement à l'EPITA
  - aussi responsable L3 MAG
  - `uli@lmf.cnrs.fr`
- Christine Paulin
  - chargée TD groupe 5, ancienne responsable cours
- Aquilina Al Khoury
  - chargée TD groupe 4
- Jérémy Marrez
  - chargé TD groupe 3
- Adrien Durier
  - chargé TD groupe 2
- Gérald Forhan
  - chargé TD groupe 1

1 Motivations

2 Organisation du cours

3 Exemples d'usage de la logique
- Résoudre des problèmes
- Modéliser, prouver

Jeu du Sudoku

8		4				2		9
		9				1		
1			3		2			7
	5		1		4		8	
				3				
	1		7		9		2	
5			4		3			8
		3				4		
4		6				3		1

Règles du jeu :

- les chiffres de 1 à 9 apparaissent une et une seule fois sur chaque ligne, chaque colonne et dans chaque cadran 3×3

Jeu du Sudoku

8		4				2		9
		9				1		
1			3		2			7
	5		1		4		8	
				3				
	1		7		9		2	
5			4		3			8
		3				4		
4		6				3		1

Règles du jeu :

- les chiffres de 1 à 9 apparaissent une et une seule fois sur chaque ligne, chaque colonne et dans chaque cadran 3×3

Solution :

8	7	4	6	5	1	2	3	9
2	3	9	8	4	7	1	6	5
1	6	5	3	9	2	8	4	7
6	5	7	1	2	4	9	8	3
9	4	2	5	3	8	7	1	6
3	1	8	7	6	9	5	2	4
5	2	1	4	7	3	6	9	8
7	8	3	9	1	6	4	5	2
4	9	6	2	8	5	3	7	1

Modélisation propositionnelle

- variables à valeur vrai/faux
- position $p = (i, j)$ sur la ligne i et la colonne j
- variable propositionnelle x_p^k : vraie si le chiffre k est à la position p
- au total $9 \times 9 \times 9 = 729$ variables, soit $2^{729} \simeq 10^{219}$ possibilités
- exemples de règles du jeu
  - à généraliser pour chacune des 81 positions

$$x_{1,1}^1 \vee x_{1,1}^2 \vee x_{1,1}^3 \vee x_{1,1}^4 \vee x_{1,1}^5 \vee x_{1,1}^6 \vee x_{1,1}^7 \vee x_{1,1}^8 \vee x_{1,1}^9$$

  - à généraliser pour chacune des 81 positions et chacune des 9 valeurs possibles

$$x_{1,1}^1 \Rightarrow (\neg x_{1,2}^1 \wedge \neg x_{1,3}^1 \wedge \neg x_{1,4}^1 \wedge \neg x_{1,5}^1 \wedge \neg x_{1,6}^1 \wedge \neg x_{1,7}^1 \wedge \neg x_{1,8}^1 \wedge \neg x_{1,9}^1)$$

  - ...
- grille initiale : force certaines variables à vrai

$$x_{1,1}^8 \quad x_{1,3}^4 \quad x_{1,7}^2 \ldots$$

Exemple, complèt (?)

8		4				2		9
		9				1		
1			3		2			7
	5		1		4		8	
				3				
	1		7		9		2	
5			4		3			8
		3				4		
4		6				3		1

$$(x_{1,1}^8 \wedge x_{1,3}^4 \wedge \dots \wedge x_{2,3}^9 \wedge \dots \wedge x_{9,9}^1) \wedge$$

$$(x_{1,1}^1 \Rightarrow (\neg x_{1,2}^1 \wedge \neg x_{1,3}^1 \wedge \neg x_{1,4}^1 \wedge \neg x_{1,5}^1 \wedge \neg x_{1,6}^1 \wedge \neg x_{1,7}^1 \wedge \neg x_{1,8}^1 \wedge \neg x_{1,9}^1)) \wedge$$

$$(x_{1,1}^2 \Rightarrow (\neg x_{1,2}^2 \wedge \neg x_{1,3}^2 \wedge \neg x_{1,4}^2 \wedge \neg x_{1,5}^2 \wedge \neg x_{1,6}^2 \wedge \neg x_{1,7}^2 \wedge \neg x_{1,8}^2 \wedge \neg x_{1,9}^2)) \wedge \dots$$

$$(x_{9,9}^9 \Rightarrow (\neg x_{9,1}^9 \wedge \neg x_{9,2}^9 \wedge \neg x_{9,3}^9 \wedge \neg x_{9,4}^9 \wedge \neg x_{9,5}^9 \wedge \neg x_{9,6}^9 \wedge \neg x_{9,7}^9 \wedge \neg x_{9,8}^9)) \wedge$$

$$(x_{1,1}^1 \Rightarrow (\neg x_{2,1}^1 \wedge \neg x_{3,1}^1 \wedge \neg x_{4,1}^1 \wedge \neg x_{5,1}^1 \wedge \neg x_{6,1}^1 \wedge \neg x_{7,1}^1 \wedge \neg x_{8,1}^1 \wedge \neg x_{9,1}^1)) \wedge \dots$$

$$(x_{1,1}^1 \Rightarrow (\neg x_{1,2}^1 \wedge \neg x_{1,3}^1 \wedge \neg x_{2,1}^1 \wedge \neg x_{2,2}^1 \wedge \neg x_{2,3}^1 \wedge \neg x_{3,1}^1 \wedge \neg x_{3,2}^1 \wedge \neg x_{3,3}^1)) \wedge \dots$$

$$(x_{1,1}^1 \vee x_{1,1}^2 \vee x_{1,1}^3 \vee x_{1,1}^4 \vee x_{1,1}^5 \vee x_{1,1}^6 \vee x_{1,1}^7 \vee x_{1,1}^8 \vee x_{1,1}^9) \wedge \dots$$

$$(x_{9,9}^1 \vee x_{9,9}^2 \vee x_{9,9}^3 \vee x_{9,9}^4 \vee x_{9,9}^5 \vee x_{9,9}^6 \vee x_{9,9}^7 \vee x_{9,9}^8 \vee x_{9,9}^9)$$

Trouver une solution propositionnelle

- les formules peuvent être “simplifiées”
  - Ensemble de clauses (disjonctions)
    $$\neg x^1_{1,1} \vee \neg x^1_{1,2}, \neg x^1_{1,1} \vee \neg x^1_{1,3}, \neg x^1_{1,1} \vee \neg x^1_{1,4}, \dots$$
  - au total : 10287 clauses
- propager les variables résolues pour simplifier les clauses et résoudre plus de variables
- si $x^8_{1,1}$ est vraie, alors
  - $\neg x^8_{1,1} \vee \neg x^8_{1,2}$ devient $\neg x^8_{1,2}$ qui sera propagé
  - $x^1_{1,2} \vee x^2_{1,2} \vee x^3_{1,2} \vee x^4_{1,2} \vee x^5_{1,2} \vee x^6_{1,2} \vee x^7_{1,2} \vee x^8_{1,2} \vee x^9_{1,2}$ devient $x^1_{1,2} \vee x^2_{1,2} \vee x^3_{1,2} \vee x^4_{1,2} \vee x^5_{1,2} \vee x^6_{1,2} \vee x^7_{1,2} \vee x^9_{1,2}$
  - $\neg x^8_{1,2} \vee \neg x^8_{1,3}$ disparait car toujours vrai
- Si toutes les clauses restantes ont au moins 2 variables alors on explore les deux possibilités pour une variable : vraie ou fausse
- Si on tombe sur une contradiction (clause fausse = règle du jeu non respectée), alors on revient en arrière pour explorer une autre branche.

Mise en œuvre

- Modélisation : programme qui engendre les clauses du Sudoku
- Recherche de solution : procédure DPLL [1] qui cherche des valeurs de variables qui rendent vraies un ensemble de clauses (SAT : satisfiabilité)
- Mise en oeuvre en Ocaml :
  - 170 lignes pour les clauses et la procédure SAT (une solution ou toutes les solutions, sans optimisation)
  - 150 lignes pour la modélisation du Sudoku générique en la dimension

1. Davis–Putnam–Logemann–Loveland

Introduction au cours de logique

1 Motivations

2 Organisation du cours

3 Exemples d'usage de la logique
- Résoudre des problèmes
- Modéliser, prouver

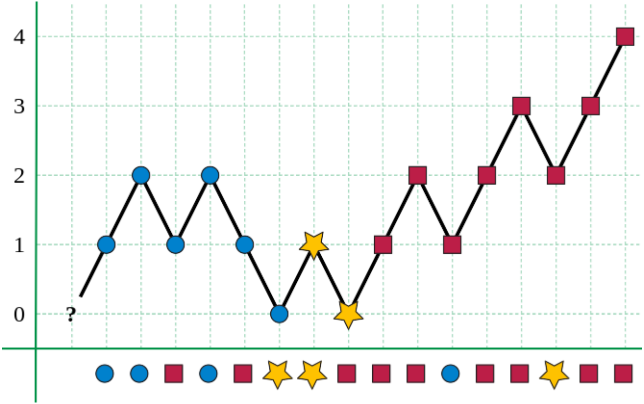
Modéliser, prouver

- une modélisation propositionnelle se résout par calcul mais est peu naturelle et ne couvre que des propriétés *finies*
- la logique des prédicats permet de raisonner sur des ensembles potentiellement infinis d'objets, de manière plus concise

Preuve de programmes : Algorithme de vote majoritaire de Boyer-Moore pour chercher une valeur *m* ayant possiblement la majorité absolue dans un tableau *t* avec un seul compteur *c*

```
let majority t =
  let rec fmaj i c m =
      if i = Array.length t then m
      else let x = t.(i) in
          if c = 0 || m = x then fmaj (i+1) (c+1) x
          else fmaj (i+1) (c-1) m
  in fmaj 0 0 t.(0)
```

Principe de l'algorithme



- examen séquentiel des valeurs (i de 0 à `length` $t - 1$)
- m est le candidat majoritaire potentiel (aucun autre n'a la majorité)
- deux valeurs différentes $m \neq x$ s'annulent mutuellement
- il y a c occurrences de m qui n'ont pas été annulées

Modélisation logique

- Propriété attendue :
  - aucune autre valeur que le résultat a la majorité absolue dans t
  - $x \neq m$ apparait au plus $(\texttt{length}\, t)/2$ fois dans t
- Invariant à l'étape i, c, m :
  - $x \neq m$ apparait au plus $(i - c)/2$ fois parmi les i premiers éléments de t
  - m apparait au plus $\frac{i-c}{2} + c$ fois parmi les i premiers éléments de t
- Terminaison : $(\texttt{length}\, t) - i$ décroit strictement en restant positif
- Théorie utilisée :
  - arithmétique (linéaire)
  - tableau (en lecture)
  - spécifique : nombre d'apparitions d'une valeur dans un segment de tableau

```
use int.Int use array.Array

(* nombre d'occurrences de x dans les i premiers éléments de t *)
function nbc (x:int) (t:array int) (i : int) : int
axiom nbc0 : forall x t. nbc x t 0 = 0
axiom nbceq : forall x t i.
      0<=i<length t -> t[i]=x -> nbc x t (i+1) = 1 + nbc x t i
axiom nbcneq : forall  x:int. forall t i.
      0<=i<length t -> t[i]<>x -> nbc x t (i+1) = nbc x t i
function nb (x:int) (t:array int) : int = nbc x t (length t)

(* majorité absolue *)
predicate maj (m :int) (t:array int) = length t < 2 * nb m t
(* invariant de l'algorithme *)
predicate inv (t : array int) (i : int) (c : int) (m : int)
    = 0<=c /\ 0<=i<=length t
    /\ 2 * nbc m t i <= i+c /\ forall x. x<>m -> 2 * nbc x t i <=
```

Preuve de programme en Why3 : programme annoté

Logique de Hoare (voir cours de GLA)

```
let majority (t : array int) : int
requires {length t <> 0}
ensures {forall x. x <> result -> not (maj x t)}
=
let rec fmaj (i : int) (c : int) (m :int) : int
  requires {inv t i c m}
  ensures {forall x. x <> result -> not (maj x t)}
  variant {length t - i}
  =
  if i = length t then m
  else let x = t[i] in
       if c = 0 || m = x then fmaj (i+1) (c+1) x
       else fmaj (i+1) (c-1) m
in fmaj 0 0 t[0]
```

Preuve automatique en utilisant alt-ergo (*SMT-solver*)


That's all Folks!

1–Maitriser le langage logique

1. Définition du langage
2. Structure des formules
3. Formules vraies
4. Théorie et modélisation

1–Maitriser le langage logique

1. Définition du langage
   - Objets
   - Formules
   - Traduire des énoncés
2. Structure des formules
3. Formules vraies
4. Théorie et modélisation

Langage formel

- On utilise un langage *formel* pour écrire les énoncés logiques
- Éviter les *ambiguïtés* du langage naturel et les notations imprécises.
  - Je peux t'offrir de l'eau *ou* du vin/Je peux t'offrir de l'eau *et* du vin
  - Le menu propose fromage *ou* dessert/Le menu propose fromage *et* dessert
  - S'il ne pleut pas, je sors jouer/S'il pleut, je ne sors pas jouer
- Moins de *redondance* que le langage naturel : plus simple à étudier, plus simple à implémenter
- Un énoncé écrit dans le langage de la logique sera appelé formule

Introduction au cours de logique

1 Définition du langage
- Objets
- Formules
- Traduire des énoncés

2 Structure des formules

3 Formules vraies

4 Théorie et modélisation

Termes

- Les énoncés parlent d'*objets*
  - entiers, individus, livres, ensembles, fonctions ...
- Dans le langage de la logique, un objet est représenté par un terme
- Un terme peut être une constante $0, 1, \mathtt{Martin}, \emptyset, \mathbb{N}$ ...,
- Un terme peut être construit à partir d'*opérations* $+, \times, \cup, \ldots$
  $3 + 5$, $\mathbb{N} \times \mathbb{N}$, `le père de Martin`, ...
  Une opération est un *symbole* associé à une arité, entier naturel qui représente le nombre d'arguments attendus.
- Dans une formule, on utilise des variables : symboles qui représentent des objets indéterminés
  $x + 1$, `un étudiant de licence`, ...

Signature, termes

Definition (Signature, arité)

Une signature est un ensemble de symboles $\mathcal{F}$ chacun associé à un entier naturel appelé arité.
Un symbole d'arité 0 est appelé constante, un symbole d'arité 1 est dit unaire, un symbole d'arité 2 est dit binaire.

Definition (Terme)

Etant donné une signature $\mathcal{F}$ et un ensemble $\mathcal{X}$ de variables, un terme t est soit une variable, soit formé d'un symbole f d'arité n et d'une suite ordonnée de n termes $t_1, \ldots, t_n$.

- f est le symbole de tête, $t_1, \ldots, t_n$ sont les sous-termes directs du terme t.
- $\mathcal{T}(\mathcal{F}, \mathcal{X})$ est l'ensemble des termes sur la signature $\mathcal{F}$ et l'ensemble des variables $\mathcal{X}$.
- $\mathcal{T}(\mathcal{F})$ est l'ensemble des termes qui ne contiennent pas de variable, appelés aussi termes clos.

Exemples de signature

- Entiers naturels :
  - constantes 0 et 1
  - opérations binaires : addition $+$ et la multiplication $\times$
  - notation infixe : $(t + u)$, $(t \times u)$
  - termes : $x + 1$, $(0 + 1)$, $(1 + 1) \times (1 + 1 + 1)$,...
- Mots binaires de longueur arbitraire
  - constante ϵ pour représenter le mot vide
  - deux fonctions unaires c_0 et c_1 pour représenter l'ajout d'un 0 ou d'un 1 en tête du mot
  - le mot 1011 est représenté par le terme $c_1(c_0(c_1(c_1(\epsilon))))$.
  - autres opérations possibles : concaténation de deux mots, décalage vers la gauche ou la droite

1 Définition du langage
- Objets
- Formules
- Traduire des énoncés

2 Structure des formules

3 Formules vraies

4 Théorie et modélisation

Formules atomiques

Les formules atomiques ne se décomposent pas en formules plus simples.

- $\top$ (top) : la propriété toujours vraie, tautologie ($0 = 0$)
- $\perp$ (bottom) : la propriété toujours fausse, l'absurde ($0 = 1$)
- symbole de prédicat associé à un ou plusieurs termes (suivant l'arité) propriétés de base des objets : ce qui ne s'explique pas en terme logique mais qui *s'observe*
  - Exemples de symboles
    - arité 0 (variable propositionnelle) : "`signal-passage-à-niveau-ouvert`",
    - arité 1 (symbole unaire, ensemble) : "`yeux-bleus`", "`pair`"
    - arité 2 (symbole binaire) : l'égalité $=$, la comparaison $\leq$, l'appartenance $\in$,
    - arité quelconque : table d'une base de données
  - Exemples de formules atomiques
    - $0 = 1$, $2 + 2 = 4$, $x = x$,
    - $x + 1 \leq x$
    - *Martin a les yeux bleus* : `yeux-bleus(Martin)`
    - `Etudiant`(Durand, Bob, 347890, 01/01/1990)

Exemples de signature : systèmes d'information

- modélisation logique des entités/ensembles et des tables/relations par des symboles de prédicat.
- Trajets de bus :
  - identifiants de ligne (numéro) des arrêts et des horaires : trois prédicats unaires `ligne`, `arret` et `horaire` pour séparer les objets de la logique suivant leur catégorie.

    $$\mathtt{ligne}(\mathtt{91-06}), \mathtt{arret}(\mathtt{Massy}), \mathtt{arret}(\mathtt{Saclay}), \mathtt{horaire}(\mathtt{06h00})\dots$$

  - table qui tient à jour les rotations de bus avec le numéro de ligne, l'arret de départ, celui d'arrivée ainsi que l'horaire de départ : predicat *trajet* d'arité 4.

    $$\mathtt{trajet}(\mathtt{91-06}, \mathtt{Massy}, \mathtt{Saclay}, \mathtt{06h00})$$

Choix des symboles de prédicat

- Le choix des symboles de prédicat dépend de la modélisation
  - primitive de base versus notions dérivées via une formule
  - analogie avec les variables d'un problème mathématique ou physique
  - chaque symbole peut s'interpréter librement
  - on peut aussi parfois choisir des symboles de fonction au lieu de symboles de prédicat (notions différentes en logique).
- Exemples
  - *tables* primitives dans une base de données, versus résultat d'une requête
  - *interface* d'une bibliothèque dont on ne connait pas l'implémentation
  - *axiomatisation* d'une *théorie*
    - entiers : $x \leq y$ versus $\exists n, y = x + n$
    - ordres : $x = y$ versus $(x \leq y \land y \leq x)$
    - hommes et femmes, versus $H(x) \stackrel{\text{def}}{=} \neg F(x)$
    - $\texttt{est-mere}(x, y)$ versus `x=mere(y)`

Syntaxe et sémantique

- Un symbole est juste un nom (syntaxe),
- la sémantique lui attribue un *sens* en lui associant une relation mathématique entre les objets modélisés
- il y a parfois un sens usuel implicite (ex : ordre sur les entiers) mais d'un point de vue logique, *toutes les interprétations sont possibles*.
- les sens possibles des symboles seront restreints par l'introduction de théories

Formules complexes

Les *formules complexes* (celles qui ne sont pas atomiques) se construisent à l'aide de connecteurs et de quantificateurs logiques.
Partie propositionnelle (connecteurs) :

- $\neg P$ la négation d'une formule, prononcée "**non** P"
- $P \wedge Q$ la conjonction de deux formules, prononcée "P **et** Q"
- $P \vee Q$ la disjonction de deux formules, prononcée "P **ou** Q"
- $P \Rightarrow Q$ l'implication de deux formules, prononcée "P **implique** Q" ou bien "**si** P **alors** Q"

Et les quantificateurs du premier ordre :

- $\forall x, P$ la quantification universelle, prononcée "**pour tout** x, P"
- $\exists x, P$ la quantification existentielle, prononcée "**il existe** x **tel que** P"

Une formule sans quantificateur $\forall$ et $\exists$ est dite formule propositionnelle

Sens intuitif des connecteurs et quantificateurs

- $\neg P$: P est faux
- $P \wedge Q$: P et Q sont tous les deux vrais
- $P \vee Q$: soit P soit Q est vrai (ou les deux)
- $P \Rightarrow Q$: si P est vrai alors Q est vrai et si P est faux alors Q peut être vrai ou faux (P est faux ou bien Q est vrai)
- $\forall x, P$: P est vrai pour toutes les *valeurs* possibles de x
- $\exists x, P$: il existe au moins une *valeur* de x pour laquelle P est vrai

Logique et formules booléennes en programmation

- expressions pour représenter des conditions booléennes
- opérations pour la négation, la conjonction, la disjonction
- pas d'implication : on trouve à la place une conditionnelle

  $$\texttt{si}\ a\ \texttt{alors}\ b\ \texttt{sinon}\ c$$

  - b et c peuvent être des booléens ou représenter d'autres types d'objet
- les quantificateurs ne correspondent pas à des constructions de programme car en général (cas infini) ils ne sont pas *calculables*.

Exercice

On suppose que a, b et c sont des formules logiques.
Représenter la phrase `si` a `alors` b `sinon` c comme une formule logique n'utilisant que les connecteurs logiques

- faire la table de vérité
- donner une première représentation sans utiliser d'implication
- donner une seconde représentation qui contienne la formule $a \Rightarrow b$

Exemples de formules complexes

- tiers exclu : $A \lor \neg A$
- modus-ponens : $((A \Rightarrow B) \land A) \Rightarrow B$
- loi de Peirce : $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$, $((\neg A) \Rightarrow A) \Rightarrow A$
- $\forall x, \texttt{yeux-bleus}(x)$
- $\exists x, \texttt{yeux-bleus}(x)$
- $\exists x, (\texttt{yeux-bleus}(x) \Rightarrow \forall y, \texttt{yeux-bleus}(y))$
- formule *paramétrée* : l'entier x est impair : $\exists y, x = 2 \times y + 1$

Différentes catégories syntaxiques : termes

- variables (objets) : $\mathcal{X}$
- symboles de fonctions
  - constantes d'arité 0 : $\mathcal{C}$
  - fonctions d'arité au moins 1 : $\mathcal{F}$
- termes (ou objets)

$$\begin{aligned}\texttt{term} &\coloneqq \mathcal{X} \mid \mathcal{C} \mid \mathcal{F}(\texttt{list-terms})\\ \texttt{list-terms} &\coloneqq \texttt{term} \mid \texttt{list-terms}, \texttt{term}\end{aligned}$$

Différentes catégories syntaxiques : formules

- symboles de prédicats
  - d'arité 0 : $\mathcal{V}$
  - d'arité au moins 1 : $\mathcal{P}$
- formules logiques :

$$
\begin{aligned}
\texttt{form} \coloneqq{} & \top \mid \bot \mid \mathcal{V} \mid \mathcal{P}(\texttt{list-terms}) \\
& \mid \neg\texttt{form} \mid (\texttt{form} \land \texttt{form}) \mid (\texttt{form} \lor \texttt{form}) \mid (\texttt{form} \Rightarrow \texttt{form}) \\
& \mid (\forall \mathcal{X}, \texttt{form}) \mid (\exists \mathcal{X}, \texttt{form})
\end{aligned}
$$

Notations infixes

- Notation standard :
  - Fonctions/Prédicats : $\texttt{Symbole}(t_1, \dots, t_n)$
- Quelques notations usuelles infixes pour des symboles binaires $f(t, u)$
  - $t \circ u$
  - exemples : $t + u$, $t \times u$, $t = u$, $t \leq u \dots$
- Extension de la grammaire

$$\begin{aligned} \texttt{term} &\coloneqq (\texttt{term}\ \mathcal{F}_I\ \texttt{term}) \\ \texttt{form} &\coloneqq (\texttt{term}\ \mathcal{P}_I\ \texttt{term}) \end{aligned}$$

les parenthèses évitent les ambiguités

Notations, règles de parenthésage

- $A \Leftrightarrow B$ est la même chose que $(A \Rightarrow B) \wedge (B \Rightarrow A)$
- plusieurs variables dans un quantificateur, par exemple : $\forall x\, y, P$ représente la formule $\forall x, \forall y, P$
- attention $\forall x \in A, P$ ne fait pas partie du langage ni $\exists ! x, P$
- l'usage de notations infixes rend nécessaire l'ajout de parenthèses dans la syntaxe des formules
  - comment interpréter $P \wedge Q \vee R$?
    1. $(P \wedge Q) \vee R$
    2. $P \wedge (Q \vee R)$

Calcul des prédicats et calcul propositionnel

- On appelle logique du premier ordre (ou calcul des prédicats) le langage logique défini par une signature (symboles de fonctions et de prédicats) et qui n'utilise que les connecteurs logiques et les quantificateurs sur les variables de termes tels que définis précédemment.
- Le calcul propositionnel (aussi appelé logique propositionnelle) est un cas particulier dans lequel la signature est réduite à des symboles de prédicats d'arité 0 (appelées variables propositionnelles) et dans lequel on n'utilise pas de quantificateurs.
- Il existe d'autres logiques (logique d'ordre supérieur, logiques temporelles, logiques modales. . .)

1. Définition du langage
   - Objets
   - Formules
   - Traduire des énoncés
2. Structure des formules
3. Formules vraies
4. Théorie et modélisation

Exercice de traduction

Langage

- $\texttt{ami}(x, y)$: x est l'ami de y
- $\texttt{joue}(x, y)$: x joue avec y
- constante $\texttt{self}$ qui représente l'individu qui s'exprime.

Que signifient les formules suivantes en langage courant ?

1. $\forall x, (\texttt{ami}(\texttt{self}, x) \Rightarrow \neg\texttt{joue}(\texttt{self}, x))$
2. $\forall x, \texttt{joue}(\texttt{self}, x)$
3. $\forall x, \exists y, \texttt{joue}(x, y)$
4. $\forall y, \exists x, \texttt{joue}(y, x)$
5. $\exists y, \forall x, \texttt{joue}(x, y)$

Exemple de traduction

- le langage comporte un symbole de fonction $+$ binaire noté de manière infixe qui représente l'opération de sommation
- `pair`(x) représente la propriété "*x* est un entier pair"
- Exprimer par une formule les propriétés :
  - *"la somme de deux entiers pairs est un entier pair"*
  - *"la somme de deux entiers impairs est un entier pair"*

A savoir faire

- Connaître les notions de signature et arité.
- Distinguer les termes et les formules, et parmi les formules celles qui sont atomiques.
- Reconnaître les termes et les formules *syntaxiquement* bien formés.
- Comprendre le sens intuitif des connecteurs propositionnels et quantificateurs logiques.
- Lire une formule logique et traduire une propriété exprimée en langue naturelle en utilisant le langage de la logique.


That's all Folks!

La dernière fois

Definition (Signature, arité)

Une signature est un ensemble de symboles $\mathcal{F}$ chacun associé à un entier naturel appelé arité.
Un symbole d'arité 0 est appelé constante, un symbole d'arité 1 est dit unaire, un symbole d'arité 2 est dit binaire.

Definition (Terme)

Etant donné une signature $\mathcal{F}$ et un ensemble $\mathcal{X}$ de variables, un terme t est soit une variable, soit formé d'un symbole f d'arité n et d'une suite ordonnée de n termes $t_1, \dots, t_n$.

- f est le symbole de tête, $t_1, \dots, t_n$ sont les sous-termes directs du terme t.
- $\mathcal{T}(\mathcal{F}, \mathcal{X})$ est l'ensemble des termes sur la signature $\mathcal{F}$ et l'ensemble des variables $\mathcal{X}$.
- $\mathcal{T}(\mathcal{F})$ est l'ensemble des termes qui ne contiennent pas de variable, appelés aussi termes clos.

La dernière fois

Les formules atomiques ne se décomposent pas en formules plus simples.

- $\top$ (top) : la propriété toujours vraie, tautologie ($0 = 0$)
- $\bot$ (bottom) : la propriété toujours fausse, l'absurde ($0 = 1$)
- symbole de prédicat associé à un ou plusieurs termes (suivant l'arité) propriétés de base des objets : ce qui ne s'explique pas en terme logique mais qui *s'observe*
  - Exemples de symboles
    - arité 0 (variable propositionnelle) : "`signal-passage-à-niveau-ouvert`",
    - arité 1 (symbole unaire, ensemble) : "`yeux-bleus`", "`pair`"
    - arité 2 (symbole binaire) : l'égalité $=$, la comparaison $\leq$, l'appartenance $\in$,
    - arité quelconque : table d'une base de données
  - Exemples de formules atomiques
    - $0 = 1$, $2 + 2 = 4$, $x = x$,
    - $x + 1 \leq x$
    - *Martin a les yeux bleus* : `yeux-bleus(Martin)`
    - `Etudiant`(Durand, Bob, 347890, 01/01/1990)

La dernière fois

Les *formules complexes* (celles qui ne sont pas atomiques) se construisent à l'aide de connecteurs et de quantificateurs logiques.
Partie propositionnelle (connecteurs) :

- $\neg P$ la négation d'une formule, prononcée "**non** P"
- $P \wedge Q$ la conjonction de deux formules, prononcée "P **et** Q"
- $P \vee Q$ la disjonction de deux formules, prononcée "P **ou** Q"
- $P \Rightarrow Q$ l'implication de deux formules, prononcée "P **implique** Q" ou bien "**si** P **alors** Q"

Et les quantificateurs du premier ordre :

- $\forall x, P$ la quantification universelle, prononcée "**pour tout** x, P"
- $\exists x, P$ la quantification existentielle, prononcée "**il existe** x **tel que** P"

Une formule sans quantificateur $\forall$ et $\exists$ est dite formule propositionnelle

La dernière fois

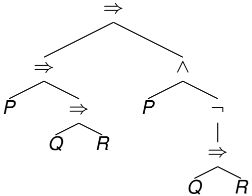
- On appelle logique du premier ordre (ou calcul des prédicats) le langage logique défini par une signature (symboles de fonctions et de prédicats) et qui n'utilise que les connecteurs logiques et les quantificateurs sur les variables de termes tels que définis précédemment.
- Le calcul propositionnel (aussi appelé logique propositionnelle) est un cas particulier dans lequel la signature est réduite à des symboles de prédicats d'arité 0 (appelées variables propositionnelles) et dans lequel on n'utilise pas de quantificateurs.
- Il existe d'autres logiques (logique d'ordre supérieur, logiques temporelles, logiques modales...)

1–Maitriser le langage logique

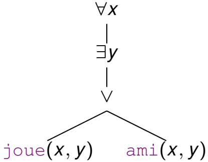
1. Définition du langage
2. Structure des formules
   - Représentation par des arbres
   - Règles de parenthésage
   - Variables libres et liées
3. Formules vraies
4. Théorie et modélisation

Structure des formules

- Une formule se représente comme un arbre
  - les nœuds internes sont les connecteurs et les quantificateurs $\forall x$, et $\exists x$
  - les feuilles sont les formules atomiques
- $(P \Rightarrow (Q \Rightarrow R)) \Rightarrow (P \wedge \neg(Q \Rightarrow R))$



- $\forall x, \exists y, (\mathtt{joue}(x, y) \vee \mathtt{ami}(x, y))$



Exercice

représenter sous forme d'arbre les formules

- $(P \Rightarrow Q) \Rightarrow (\neg Q \Rightarrow \neg P)$
- $\forall x, ((\forall y, \neg \mathtt{ami}(x, y)) \Rightarrow \mathtt{joue}(x, x))$

Règles de précédence

- comment interpréter $P \wedge Q \vee R$? $\forall x, P(x) \Rightarrow Q(x)$
- règles de précédence :
  - La précédence de $\neg$ est la plus *forte*, vient ensuite la conjonction $\wedge$ puis la disjonction $\vee$ et finalement l'implication $\Rightarrow$.
  - Les connecteurs $\wedge$, $\vee$ et $\Rightarrow$ associent à *droite*

  $A \Rightarrow B \Rightarrow C$ se parenthèse $A \Rightarrow (B \Rightarrow C)$
- Les quantificateurs $\forall$ et $\exists$ ont une précédence plus *faible* que les autres connecteurs.

  $\forall x, P \Rightarrow Q$ se parenthèse $\forall x, (P \Rightarrow Q)$

Exercice : Règles de précédence

1. $P \land R \land \neg Q \Rightarrow P \lor Q \land R$
2. $(P \Rightarrow Q \Rightarrow \forall x, R) \Rightarrow P \land \exists x, Q \Rightarrow R$
3. $(P \Rightarrow Q) \Rightarrow \neg\forall x, Q \Rightarrow \neg P$

Grammaire avec ambiguité

$$\begin{aligned}
\mathtt{term} &\coloneqq \mathcal{X} \mid \mathcal{C} \mid \mathcal{F}(\mathtt{list\text{-}terms}) \mid \mathtt{term}\ \mathcal{F}_I\ \mathtt{term} \mid (\mathtt{term}) \\
\mathtt{list\text{-}terms} &\coloneqq \mathtt{term} \mid \mathtt{list\text{-}terms}, \mathtt{term}
\end{aligned}$$

$$\begin{aligned}
\mathtt{form} \coloneqq\ & \top \mid \bot \mid \mathcal{V} \mid \mathcal{P}(\mathtt{list\text{-}terms}) \mid \mathtt{term}\ \mathcal{P}_I\ \mathtt{term} \mid (\mathtt{form}) \\
& \mid \neg\mathtt{form} \mid \mathtt{form} \wedge \mathtt{form} \mid \mathtt{form} \vee \mathtt{form} \mid \mathtt{form} \Rightarrow \mathtt{form} \\
& \mid \forall\mathcal{X}, \mathtt{form} \mid \exists\mathcal{X}, \mathtt{form}
\end{aligned}$$

- Les règles de précédence permettent de lever les ambiguïtés
- Les parenthèses peuvent être ajoutées librement pour faciliter la compréhension.

Variables liées

- les quantificateurs sont associés à une variable $\forall x, P$ et $\exists x, P$, on dit que la variables est liée.
- une variable liée est dite *muette* : son nom peut être changé, sans changer le sens de la formule

  $$\forall x, \exists y, x < y \qquad \forall t, \exists u, t < u$$

  attention au problème de *capture* : $\forall y, \exists y, y < y$
- en langage courant, souvent on ne mentionne pas le nom
  - tous les chats sont gris : $\forall x, \mathtt{chat}(x) \Rightarrow \mathtt{gris}(x)$
  - il existe des chats qui ne sont pas gris : $\exists x, \mathtt{chat}(x) \wedge \neg \mathtt{gris}(x)$

Variables libres

- une occurrence x est libre ssi pas sous un quantificateur $\forall x$, ou $\exists x$,
- les variables libres sont les *paramètres* de la formule :
  - exemple : “x est pair” : $\exists y, x = 2 \times y$
  - la formule est vraie ou fausse en fonction de la valeur de la variable
  - analogie avec une procédure, méthode en programmation (donner des valeurs aux paramètres pour exécuter)
- une variable libre peut être remplacée par un terme plus complexe : substitution
  - “4 est pair” : $\exists y, 4 = 2 \times y$
  - “$2 \times z + 4$ est pair” : $\exists y, 2 \times z + 4 = 2 \times y$
  - attention à la capture lorsqu’on substitue une variable par un terme : “$3 \times y$ est pair” :
    - $\exists y, 3 \times y = 2 \times y$ (substitution incorrecte)
    - $\exists y', 3 \times y = 2 \times y'$ (substitution correcte après renommage)

Variables libres

- une même variable peut apparaître libre et liée dans une formule :

$$0 < x \times y \vee (\exists y, x < y) \wedge (\exists y, y + y < x)$$

- un terme qui ne contient pas de variables est appelé terme clos
- une formule qui ne contient pas de *variable libre* est appelée formule close
- Exercice :
  - donner les variables libres et liées de la formule

$$\forall b, b > 0 \Rightarrow \exists q, \exists r, a = b \times q + r \wedge r < b$$

  - cette formule est-elle close ?

Variables libres

- une même variable peut apparaître libre et liée dans une formule :

$$0 < x \times \mathbf{y} \vee (\exists y_1, x < y_1) \wedge (\exists y_2, y_2 + y_2 < x)$$

- un terme qui ne contient pas de variables est appelé terme clos
- une formule qui ne contient pas de *variable libre* est appelée formule close
- Exercice :
  - donner les variables libres et liées de la formule

$$\forall b, b > 0 \Rightarrow \exists q, \exists r, a = b \times q + r \wedge r < b$$

  - cette formule est-elle close ?

Formules syntaxiquement égales

- Deux formules P et Q sont syntaxiquement égales si elles ont la même représentation sous forme d'arbre modulo le renommage des variables liées. On écrit alors $P = Q$.
- seul le lien entre l'utilisation de la variable et le quantificateur qui l'a introduit est important
- Il est relativement “facile” d'écrire un programme qui vérifie que deux formules sont égales
- Les variables liées compliquent la vérification et font que deux formules égales peuvent avoir des représentations différentes en machine

Exercice : Formules syntaxiquement égales

Dire quelles formules sont égales ou différentes

1. $\forall x, \forall y, P(x,y)$
2. $\forall y, \forall x, P(x,y)$
3. $\forall y, \forall z, P(y,z)$

1. $\forall x, P(x) \Rightarrow Q(x) \Rightarrow \bot$
2. $\forall x, ((P(x) \Rightarrow Q(x)) \Rightarrow \bot)$
3. $\forall x, (P(x) \Rightarrow (Q(x) \Rightarrow \bot))$
4. $(\forall x, (P(x) \Rightarrow Q(x))) \Rightarrow \bot$
5. $(\forall x, P(x)) \Rightarrow (Q(x) \Rightarrow \bot)$

Comparer les représentations sous forme d'arbre !

A savoir faire

- Connaître les règles de précédences sur les connecteurs et les quantificateurs de la logique.
- Représenter une formule sous forme d'arbre.
- Reconnaître les variables libres et les variables liées d'une formule logique.
- Connaître la définition de terme clos et formule close.

1–Maitriser le langage logique

1 Définition du langage

2 Structure des formules

3 Formules vraies
- Cas propositionnel
- Modéliser un problème à l'aide de la logique
- Formules avec quantificateurs

4 Théorie et modélisation

Valeurs de vérité

- valeur de vérité, ensemble des booléens $\mathbb{B} = \{V, F\}$ (parfois notées $\{1, 0\}$)
- on cherche les conditions dans lesquelles une formule est vraie ou fausse
- on parle de sémantique (sens de la formule) par opposition à la syntaxe (forme)
- plusieurs formules syntaxiquement différentes peuvent avoir le même sens
- la même formule peut avoir différents sens suivant l'interprétation des symboles de la signature
    - Je n'ai pas d'ami
    - $1 + 1 = 0$

Formules (closes) vraies

- Les formules contiennent des symboles de fonctions et de prédicats qui correspondent à des *primitives*, de sens indéterminé
- On ne connaît la vérité d'une formule qu'*après* avoir défini l'interprétation des symboles
  - Si un programme utilise une *bibliothèque* :
    - il se compile en utilisant l'interface de la bibliothèque
    - il ne s'exécute qu'en présence du code d'implémentation de cette bibliothèque.
  - Une *base de données*
    - les requêtes se définissent en fonction de la structure (tables)
    - chaque "état" de la base de données correspond à une interprétation.

Formules (closes) valides

- La valeur de vérité d'une formule (vrai ou faux) est définie par rapport à une interprétation des symboles qu'elle contient
- On ne peut pas dire *a priori* si une formule est vraie ou fausse
  - Par abus de langage, on dit parfois qu'une formule est vraie (resp. fausse) si elle est tout le temps vraie (resp. fausse)
- Une formule (indépendament d'une interprétation) peut être
  - valide (tautologie) : vraie pour toutes les interprétations
  - insatisfiable : fausse pour toutes les interprétations
  - satisfiable : vraie pour au moins une interprétation
- analogie avec les équations

$$(x+1)^2 = x^2 + 2x + 1 \qquad x^2 = -1 \qquad x^2 = 4$$

Lien entre validité et satisfiabilité

- une formule valide est a fortiori satisfiable (perte d'information)
- un algorithme qui résoud l'une des trois questions résoud les autres.

Proposition

Les trois propriétés suivantes sont équivalentes

1. P *est valide*
2. $\neg P$ *est insatisfiable*
3. $\neg P$ *n'est pas satisfiable*

Preuve:

- P *est valide ssi* P *est vrai pour toute interprétation (définition valide)*
  ssi $\neg P$ *est faux pour toute interprétation (table de vérité de* $\neg$*)*
  ssi $\neg P$ *est insatisfiable (définition insatisfiable)*
- Q *est insatisfiable ssi* Q *est faux pour toute interprétation (définition insatisfiable)*
  ssi il n'y a pas d'interprétation qui rend Q *vrai (reformulation)*
  ssi Q *n'est pas satisfiable (définition de satisfiable)*

Modèle et valeur de vérité

- pour dire si une formule est vraie, il faut expliciter dans quelle interprétation on se place (on utilise parfois le terme modèle) : que représentent les symboles ?
- $2 \leq 4$, $0 = 1$, `Martin a les yeux bleus`, `Bob X. est inscrit en L3 Info. à l'U. Paris-Saclay`
- si on connait les formules atomiques vraies alors la logique nous dit si une formule *propositionnelle* est vraie ou fausse
- table de vérité :

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
F	F					
F	V					
V	F					
V	V					

Exercice

Les formules suivantes sont-elles satisfiables ? valides ?

1. $\neg(p \wedge q) \Rightarrow \neg p \vee \neg q$
2. $\neg(p \wedge q) \Rightarrow \neg p \vee q$
3. $\neg p \wedge p$
4. $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$
5. $(\neg p \Rightarrow p) \Rightarrow p$

Modélisation propositionnelle de problèmes

- pour modéliser certains problèmes, on introduit des variables propositionnelles correspondant à des situations dont on cherche à déterminer si elles sont vraies ou fausses
- des formules logiques représentent les contraintes associées au problème
- on cherche à déterminer si le problème a une solution (satisfiabilité), le nombre de solutions ...
- les outils informatiques qui réalisent ces tâches s'appellent des *SAT-solver*
- de nombreuses applications industrielles
  - vérification de hardware
  - résolution de contraintes, planification...

Exercice : Enigme

Arthur, Bob et Casimir sont soupçonnés d'avoir peint en bleu le chat de la voisine. Ils font les déclarations suivantes :

- Arthur : Bob est coupable et Casimir est innocent.
- Bob : Si Arthur est coupable, Casimir aussi.
- Casimir : Je suis innocent mais au moins l'un des deux autres est coupable.

1. On pose : a = "Arthur est coupable", b = "Bob est coupable" et c = "Casimir est coupable". Avec ces notations transcrire les trois déclarations ci-dessus dans le langage de la logique propositionnelle (notées F_A, F_B et F_C).
2. Construire la table de vérité des formules F_A, F_B et F_C.
3. En utilisant la question précédente, répondez aux questions suivantes :
   1. Montrer que si Casimir a menti alors Arthur aussi.
   2. Si Casimir a menti que peut-on dire de la déclaration de Bob ?
   3. En supposant que tous ont dit la vérité, qui est coupable qui est innocent ?
   4. En supposant que tous sont coupables, qui a dit vrai ? qui a menti ?
   5. Est-il possible que tous les innocents aient menti et que tous les coupables aient dit la vérité ?

1–Maitriser le langage logique

1 Définition du langage

2 Structure des formules

3 Formules vraies
- Cas propositionnel
- Modéliser un problème à l'aide de la logique
- Formules avec quantificateurs

4 Théorie et modélisation

Formules avec quantificateurs

- les *variables d'objets* représentent des inconnues dans un univers a priori indéterminé
  $\exists x, \mathtt{yeux\text{-}bleus}(x) \qquad \forall x\, y, x < y \Rightarrow (x+1) \leq y$
- une interprétation (modèle) va expliciter le domaine d'interprétation des objets (noté D) non vide
  - Ex. : entiers signés ou non, sur 32 ou 64 bits, population dans une BD.
- on interprète les symboles de constante et les opérateurs dans ce domaine (ex. `maxint`, la division…)
- un terme t sans variable représente une valeur t_D (le résultat du calcul) dans le domaine $t_D \in D$ (ex. $2+3$)
- un terme avec variable $x+3$ s'interprète comme une valeur $t_D \in D$ en se donnant en plus un environnement qui définit les valeurs des variables
  - analogie avec la mémoire d'un ordinateur

Interprétation des formules avec quantificateurs

- les symboles de prédicat `yeux-bleus`, $\leq$... peuvent aussi s'interpréter de différentes manières. L'interprétation leur associe une relation entre les objets
  - relation unaire (`yeux-bleus`) : ensemble d'objets
  - relation binaire ($\leq$) : graphe orienté
  - relations d'arité supérieure : tables
- analogie avec les langages de programmation
  - la syntaxe : les règles pour écrire un programme syntaxiquement correct
  - la sémantique : les règles qui expliquent quel sera le résultat de l'exécution d'un programme (dans un certain contexte)
  - plusieurs sens possibles pour le même programme
  - plusieurs compilateurs peuvent donner des résultats différents

Vérité d'une formule quantifiée

- A quelle condition $\forall x, P(x)$ est-il vrai ?
- on ne peut parler de la valeur de vérité d'une formule que dans une interprétation qui définit le domaine (D) des objets, et l'interprétation des symboles (constantes, fonctions, prédicats) ainsi que la valeur des variables libres de la formule (l'environnement)
- $\forall x, P$ est vraie si pour tout objet $d \in D$, la formule P est vraie dans l'environnement dans lequel x a la valeur d.
- $\exists x, P$ est vraie s'il existe un objet $d \in D$ tel que la formule P est vraie dans l'environnement dans lequel x à la valeur d.

Interprétation des symboles

- Domaine D des objets : ensemble non vide
- A chaque constante on associe un élément du domaine
- A chaque symbole de fonction on associe une fonction sur le domaine
- Une formule atomique $P(t_1, \dots, t_n)$ représente une vérité qui dépend de la valeur des arguments $t_1, \dots, t_n$.
  On interpréte P par une relation n-aire sur l'ensemble D (à quelle condition sur les entrées la formule est vraie).
- Exemples : `yeux-bleus`, `ami`, ...

Validité, satisfiabilité

- Validité : vrai pour tout domaine, toute interprétation des symboles, toutes valeurs des variables libres
- Satisfiabilité : vrai pour au moins un domaine, une interprétation des symboles, une valeur des variables libres
- Insatisfiabilité : faux pour tout domaine, toute interprétation des symboles, toutes valeurs des variables libres

Formules valides

- Les formules suivantes sont-elles valides (vraies pour tout domaine et interprétation du prédicat P) ? Sont-elles satisfiables ?
  1. $(\forall x, P(x)) \Rightarrow (\exists x, P(x))$
  2. $(\forall x, P(x)) \wedge (\exists x, \neg P(x))$
  3. $\neg(\forall x, P(x)) \Rightarrow \exists x, \neg P(x)$
  4. $(\exists x, P(x)) \Rightarrow \forall x, P(x)$
  5. $(\forall x, P(x)) \Rightarrow \forall x, \neg P(x)$

Remarque

- Quand il y a une infinité d'objets, on ne peut plus faire des tables de vérité !

Modélisation en logique du premier ordre I

Trouver l'erreur

- Tout ce qui est rare est cher.
- Un cheval bon marché est rare,
- donc un cheval bon marché est cher

1. introduire des prédicats pour modéliser la situation
2. exprimer les propriétés précédentes comme des formules logiques

A savoir faire

- Tables de vérité des formules propositionnelles
- Définir une interprétation simple
- Evaluer la vérité d'une formule dans une interprétation
- Liens entre validité et satisfiabilité

Ces notions seront approfondies dans le chapitre 2

1–Maitriser le langage logique

1. Définition du langage
2. Structure des formules
3. Formules vraies
4. Théorie et modélisation

Pourquoi des théories ?

- un symbole peut être interprété de plusieurs manières différentes
  - $\mathtt{ami}$, $\mathtt{joue}$
  - $\mathtt{pair}$
- Formule valide : vraie dans toutes les interprétations possibles
  - des vérités "*absolues*"
  - rien n'empêche une interprétation dans laquelle $\mathtt{ami}(t, u)$ est vrai mais $\mathtt{ami}(u, t)$ est faux ou bien $\mathtt{pair}(1)$ est vrai.
- Pour limiter les interprétations, on ajoute des axiomes (formule sans variables libres)

$$0 \neq 1 \qquad\qquad \forall x\, y, \mathtt{ami}(x, y) \Rightarrow \mathtt{ami}(y, x)$$

- Les axiomes sont des *vérités* qui n'ont pas besoin d'être démontrées.
  - on se limite aux interprétations qui rendent vrais les axiomes.
  - dans les démonstrations, les axiomes sont traités comme des hypothèses supplémentaires

Definition (Théorie)

Une théorie est définie par un ensemble de symboles de fonctions et de prédicats (la signature de la théorie) et un ensemble de formules closes (sans variables libres) construites sur ce langage, appelés les axiomes de la théorie.

- Une théorie peut être définie par un ensemble fini ou infini d'axiomes.
- Un modèle d'une théorie est donné par une interprétation de la signature dans laquelle tous les axiomes ont pour valeur vraie.
- Une formule est valide dans une théorie si elle vraie dans tous les modèles de la théorie, on dit aussi que c'est une conséquence logique des axiomes de la théorie.
- axiomatiser une théorie (géométrie,. . .)
    - Partir d'un ensemble restreint d'objets de base et de leurs propriétés
    - Construire logiquement les concepts avancés et les théorèmes

Axiomes pour les groupes I

Exemple type : les *entiers relatifs* ($\mathbb{Z}$)

- symboles de fonctions :
  - constante 0,
  - opération binaire $+$,
  - opération unaire $-$ (le $t - u$ binaire n'est pas primitif : $t + (-u)$)
- symbole de prédicat binaire : égalité.
- 3 axiomes (+ ceux de l'égalité)
  - associativité : $\forall x\, y\, z, (x + y) + z = x + (y + z)$
  - élement neutre : $\forall x, x + 0 = x \wedge 0 + x = x$
  - inverse : $\forall x, x + (-x) = 0 \wedge (-x) + x = 0$
- Modèle : entiers relatifs, groupe des permutations. . .
- Conséquence : $\forall x, -(-x) = x$

$$- - x = - - x + 0 = - - x + (-x + x) = (- - x + -x) + x = 0 + x = x$$

(utilise les propriétes de symétrie, transitivité et congruence de l'égalité)

Axiomes pour la géométrie

- Euclide : points, segment, droite, demi-droite et cercle
- Hilbert : points, droites, incidence (un point est sur une droite), un point est entre deux autres (segment), congruence de segments, angles, triangles
  - par deux points, il passe une et une seule droite
  - sur une droite, il y a au moins 2 points distincts et un point qui n'est pas sur la droite
  - parallèle : soit une droite d et un point x qui n'est pas sur la droite, il existe une unique droite qui passe par x et qu n'a pas de point commun avec d
- Modélisation logique :
  - prédicats unaires P et D (points et droites)
  - prédicats binaires : $x \in d$ et $p = q \dots$

$$\forall p\, q, P(p) \wedge P(q) \wedge \neg(p = q) \Rightarrow$$
$$\exists d, D(d) \wedge p \in d \wedge q \in d \wedge (\forall d', D(d') \wedge p \in d' \wedge q \in d' \Rightarrow d = d')$$

  - la géométrie (espaces euclidiens) est un modèle de la théorie de Hilbert

Théorie de l'égalité

Un symbole binaire quelconque, noté $=$ de manière infixe.

- réflexivité : $\forall x, x = x$
- symétrie : $\forall x\, y, x = y \Rightarrow y = x$
- transitivité : $\forall x\, y\, z, (x = y \wedge y = z) \Rightarrow x = z$
- congruence/symboles de fonction f (arité n) :
  $\forall x_1 \dots x_n\, y_1 \dots y_n, (x_1 = y_1 \wedge \dots \wedge x_n = y_n) \Rightarrow f(x_1, \dots, x_n) = f(y_1, \dots, y_n)$
- congruence/symboles de prédicat P (arité n) :
  $\forall x_1 \dots x_n\, y_1 \dots y_n, (x_1 = y_1 \wedge \dots \wedge x_n = y_n) \Rightarrow P(x_1, \dots, x_n) \Rightarrow P(y_1, \dots, y_n)$

Exemples

$$\begin{array}{l}
\forall x_1\, x_2\, y_1\, y_2, (x_1 = y_1) \wedge (x_2 = y_2) \Rightarrow (x_1 + x_2) = (y_1 + y_2) \\
\forall x_1\, y_1, (x_1 = y_1) \Rightarrow (-x_1) = (-y_1) \\
\forall x_1\, y_1, (x_1 = y_1) \Rightarrow D(x_1) \Rightarrow D(y_1)
\end{array}$$

L'égalité préserve le prédicat d'égalité trivialement (symétrie et transitivité)

$$\forall x_1\, x_2\, y_1\, y_2, (x_1 = y_1) \wedge (x_2 = y_2) \Rightarrow (x_1 = x_2) \Rightarrow (y_1 = y_2)$$

Egalité

- on pourra écrire $t \neq u$ pour la formule $\neg(t = u)$
- pour n'importe quelle formule Φ avec une variable libre x :

$$\forall x\, y, (x = y) \Rightarrow (\Phi \Rightarrow \Phi[x \leftarrow y])$$

  Preuve par récurrence sur la structure de la formule (voir suite)
- le symbole d'égalité sera utilisé avec les axiomes précédents implicites
- l'interprétation de l'égalité est une relation d'equivalence quelconque (pas forcément l'égalité du domaine)
- les interprétations des symboles de fonction et de prédicat doivent respecter la congruence
  - exemple des rationnels, des ensembles finis. . .

Théorie arithmétique (Peano)

- Constante : 0
- Opération unaire *successeur* $(+1)$: $S(\_)$,
- Opérations binaires addition et multiplication : $\_ + \_$ et $\_ * \_$
- Symbole de prédicat binaire d'égalité $(=)$
- Axiomes de l'égalité
- Axiomes arithmétique
  - $\forall x, S(x) \neq O$
  - $\forall x, x = O \vee \exists y, x = S(y)$ (inutile en présence de récurrence)
  - $\forall x\, y, S(x) = S(y) \Rightarrow x = y$
  - $\forall x, x + O = x$
  - $\forall x\, y, x + S(y) = S(x + y)$
  - $\forall x, x \times O = O$
  - $\forall x\, y, x \times S(y) = (x \times y) + x$
- Récurrence (nombre dénombrable d'axiomes, un pour chaque formule Φ)

$$\forall x_1 \ldots x_n, \Phi[x \leftarrow O] \Rightarrow (\forall x, \Phi \Rightarrow \Phi[x \leftarrow S(x)]) \Rightarrow \forall x, \Phi$$

Illustration

- A chaque entier naturel $n \in \mathbb{N}$ on fait correspondre un terme noté $\tilde{n}$ correspondant à $S^n(O)$.
- A partir des symboles de la théorie, on peut définir de nouvelles notions
  - $t \leq u \stackrel{\text{def}}{=} \exists d, t + d = u$
- A partir des axiomes de la théorie, on peut déduire de nouvelles propriétés
  - $\forall x, 0 \leq x$
  - $\forall x\, y, x \leq y \Leftrightarrow S(x) \leq S(y)$
  - transitivité : $\forall x\, y\, z, x \leq y \Rightarrow y \leq z \Rightarrow x \leq z$

Application de la théorie de l'arithmétique

- La théorie arithmétique est suffisante pour *représenter* les fonctions et prédicats calculables sur les entiers.
  à une fonction $f \in \mathbb{N} \to \mathbb{N}$ correspond une formule $F[x, y]$

  $$f(n) = m \quad \text{ssi} \quad F[\tilde{n}, \tilde{m}] \text{ est vrai dans l'arithmétique de Peano}$$

- Exemples
  - soustraction $(x - y)$: $F[x, y, z] \stackrel{\text{def}}{=} y + z = x \vee x < y \wedge z = O$
  - minimisation (+ petit n tel que $G(x, n)$) :
    $\mu G[x, y] \stackrel{\text{def}}{=} G(x, y) \wedge \forall z, z < y \Rightarrow \neg G(x, z)$
  - Fonctions complexes via un codage des couples et des suites pour simuler les calculs récursifs


That's all Folks!