

# 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  $E_1$  *et*  $E_2$  est vrai si et seulement si les énoncés  $E_1$  et  $E_2$  sont simultanément vrais
  - la **disjonction** : l'énoncé  $E_1$  *ou*  $E_2$  est vrai si et seulement si l'un des deux énoncés  $E_1$  et  $E_2$  est vrai (ou les deux)
  - ...

# Énoncé paramétré et quantification

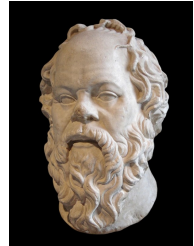
- Un énoncé logique *paramétré* représente une propriété (vraie ou fausse) d'un "*objet*" quelconque
- exemple : propriété pour un étudiant de *valider le cours de logique*
- "*valider le cours de logique*" sera vrai ou faux pour chaque étudiant considéré
  - *Toto valide le cours de logique*
- on peut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



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

- 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 ?

- 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...

- *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*)

- 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

- 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. . .

- 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



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



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 Schwarzentruher.

Logique : fondements et applications.  
Dunod, 2022.

- 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)

- 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*

- 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

# Introduction au cours de logique

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  $3 \times 3$

# 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  $3 \times 3$

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

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

# Exemple, complet (?)

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)

$$\neg x_{1,1}^1 \vee \neg x_{1,2}^1, \neg x_{1,1}^1 \vee \neg x_{1,3}^1, \neg x_{1,1}^1 \vee \neg x_{1,4}^1, \dots$$

- au total : 10287 clauses
- propager les variables résolues pour simplifier les clauses et résoudre plus de variables
- si  $x_{1,1}^8$  est vraie, alors
  - $\neg x_{1,1}^8 \vee \neg x_{1,2}^8$  devient  $\neg x_{1,2}^8$  qui sera propagé
  - $x_{1,2}^1 \vee x_{1,2}^2 \vee x_{1,2}^3 \vee x_{1,2}^4 \vee x_{1,2}^5 \vee x_{1,2}^6 \vee x_{1,2}^7 \vee x_{1,2}^8 \vee x_{1,2}^9$  devient  $x_{1,2}^1 \vee x_{1,2}^2 \vee x_{1,2}^3 \vee x_{1,2}^4 \vee x_{1,2}^5 \vee x_{1,2}^6 \vee x_{1,2}^7 \vee x_{1,2}^9$
  - $\neg x_{1,2}^8 \vee \neg x_{1,3}^8$  disparaît 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.

- Modélisation : programme qui engendre les clauses du Sudoku
- Recherche de solution : procédure DPLL<sup>1</sup> 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

# Introduction au cours de logique

1 Motivations

2 Organisation du cours

3 Exemples d'usage de la logique

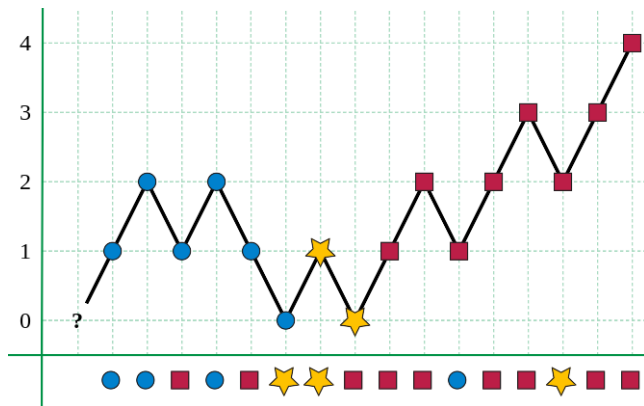
- Résoudre des problèmes
- **Modéliser, prouver**

- 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  $\text{length } t - 1$ )
- $m$  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

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



# Preuve de programme en Why3 : théorie logique

```
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 nbcdif : forall x 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 * nb 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—Maîtriser le langage logique

- 1 Définition du langage
- 2 Structure des formules
- 3 Formules vraies
- 4 Théorie et modélisation
- 5 Définition récursive sur les formules

# 1–Maîtriser le langage logique

- 1 Définition du langage
  - Objets
  - Formules
  - Traduire des énoncés
- 2 Structure des formules
- 3 Formules vraies
- 4 Théorie et modélisation
- 5 Définition récursive sur les formules

- 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
- 5 Définition récursive sur les formules

- 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  $0, 1, \text{Martin}, \emptyset, \mathbb{N} \dots$ ,
- Un terme peut être construit à partir d'*opérations*  $+, \times, \cup, \dots$   
 $3 + 5, \mathbb{N} \times \mathbb{N}, \text{le père de Martin}, \dots$   
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  
 $x + 1, \text{un étudiant de licence}, \dots$



## 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**.

# 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  $\epsilon$  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

# 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
- 5 Définition récursive sur les formules

# 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`, `arrêt` et `horaire` pour séparer les objets de la logique suivant leur catégorie.  
`ligne(91-06), arrêt(Massy), arrêt(Saclay), horaire(06h00)...`
  - table qui tient à jour les rotations de bus avec le numéro de ligne, l'arrêt de départ, celui d'arrivée ainsi que l'horaire de départ : predicat `trajet` d'arité 4.  
`trajet(91-06, Massy, Saclay, 06h00)`

- 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 connaît pas l'implémentation
  - *axiomatisation* d'une *théorie*
    - entiers :  $x \leq y$  versus  $\exists n, y = x + n$
    - ordres :  $x = y$  versus  $(x \leq y \wedge y \leq x)$
    - hommes et femmes, versus  $H(x) \stackrel{\text{def}}{=} \neg F(x)$
    - *est-mère*( $x, y$ ) versus *x=mère*( $y$ )

- 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

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

si *a* alors *b* 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 :  $A \vee \neg A$
- modus-ponens :  $((A \Rightarrow B) \wedge A) \Rightarrow B$
- loi de Peirce :  $((A \Rightarrow B) \Rightarrow A) \Rightarrow A, ((\neg A) \Rightarrow A) \Rightarrow A$
- $\forall x, \text{yeux-bleus}(x)$
- $\exists x, \text{yeux-bleus}(x)$
- $\exists x, (\text{yeux-bleus}(x) \Rightarrow \forall y, \text{yeux-bleus}(y))$
- 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)

$\text{term} \coloneqq \mathcal{X} \mid \mathcal{C} \mid \mathcal{F}(\text{list-terms})$

$\text{list-terms} \coloneqq \text{term} \mid \text{list-terms}, \text{term}$

# Différentes catégories syntaxiques : formules

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

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

# Notations infixes

- Notation standard :
  - Fonctions/Prédicats :  $\text{Symbole}(t_1, \dots, t_n)$
- 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

$$\begin{aligned}\text{term} &\vdash (\text{term } \mathcal{F}_l \text{ term}) \\ \text{form} &\vdash (\text{term } \mathcal{P}_l \text{ term})\end{aligned}$$

les parenthèses évitent les ambiguïtés

# 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  $P \wedge (Q \vee R)$



- 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. . .)

# 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
- 5 Définition récursive sur les formules

## Langage

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

Que signifient les formules suivantes en langage courant ?

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

# 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"*

- 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** *syntactiquement* 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!*

## 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 ( $0 = 0$ )
- $\perp$  (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)



# 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**

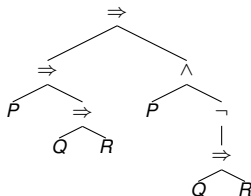
- On appelle **logique du premier ordre** (ou **calcul des prédicats**) le langage logique défini par une signature (symboles de fonctions et de prédicats) et qui n'utilise que les connecteurs logiques et les quantificateurs sur les variables de termes tels que définis précédemment.
- Le **calcul propositionnel** (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–Maîtriser le langage logique

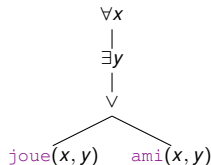
- 1 Définition du langage
- 2 **Structure des formules**
  - Représentation par des arbres
  - Règles de parenthésage
  - Variables libres et liées
- 3 Formules vraies
- 4 Théorie et modélisation
- 5 Définition récursive sur les formules

# 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
- $(P \Rightarrow (Q \Rightarrow R)) \Rightarrow (P \wedge \neg(Q \Rightarrow R))$



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



représenter sous forme d'arbre les formules

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

# Règles de précedence

- comment interpréter  $P \wedge Q \vee R$ ?  $\forall x, P(x) \Rightarrow Q(x)$
- règles de **précedence** :
  - La précedence 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 précedence plus **faible** que les autres connecteurs.



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

# Exercice : Règles de précédence

- 1  $P \wedge R \wedge \neg Q \Rightarrow P \vee Q \wedge R$
- 2  $(P \Rightarrow Q \Rightarrow \forall x, R) \Rightarrow P \wedge \exists x, Q \Rightarrow R$
- 3  $(P \Rightarrow Q) \Rightarrow \neg \forall x, Q \Rightarrow \neg P$

# Grammaire avec ambiguïté

$$\text{term} ::= \mathcal{X} \mid \mathcal{C} \mid \mathcal{F}(\text{list-terms}) \mid \text{term } \mathcal{F}_I \text{ term} \mid (\text{term})$$
$$\text{list-terms} ::= \text{term} \mid \text{list-terms}, \text{term}$$
$$\begin{aligned} \text{form} ::= & \top \mid \perp \mid \mathcal{V} \mid \mathcal{P}(\text{list-terms}) \mid \text{term } \mathcal{P}_I \text{ term} \mid (\text{form}) \\ & \mid \neg \text{form} \mid \text{form} \wedge \text{form} \mid \text{form} \vee \text{form} \mid \text{form} \Rightarrow \text{form} \\ & \mid \forall \mathcal{X}, \text{form} \mid \exists \mathcal{X}, \text{form} \end{aligned}$$

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



- 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 :  $\forall x, \text{chat}(x) \Rightarrow \text{gris}(x)$
  - il existe des chats qui ne sont pas gris :  $\exists x, \text{chat}(x) \wedge \neg \text{gris}(x)$

# 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)

- 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 ?

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

$$0 < x \times \mathbf{y} \vee (\exists \mathbf{y}_1, x < \mathbf{y}_1) \wedge (\exists \mathbf{y}_2, \mathbf{y}_2 + \mathbf{y}_2 < x)$$

- 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

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

- cette formule est-elle close ?

# Formules syntaxiquement égales

- Deux formules  $P$  et  $Q$  sont **syntactiquement 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  $\forall x, P(x) \Rightarrow Q(x) \Rightarrow \perp$

②  $\forall x, ((P(x) \Rightarrow Q(x)) \Rightarrow \perp)$

③  $\forall x, (P(x) \Rightarrow (Q(x) \Rightarrow \perp))$

④  $(\forall x, (P(x) \Rightarrow Q(x))) \Rightarrow \perp$

⑤  $(\forall x, P(x)) \Rightarrow (Q(x) \Rightarrow \perp)$

Comparer les représentations sous forme d'arbre !

- 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–Maîtriser 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
- 5 Définition récursive sur les formules



- **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 (indépendamment 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 résout l'une des trois questions résout 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 connaît les formules atomiques vraies alors la logique nous dit si une formule **propositionnelle** est vraie ou fausse
- table de vérité** :

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

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—Maîtriser 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
- 5 Définition récursive sur les formules

# Formules avec quantificateurs

- les *variables d'objets* représentent des inconnues dans un univers a priori indéterminé
$$\exists x, \text{yeux-bleus}(x) \quad \forall x y, x < y \Rightarrow (x + 1) \leq y$$
- 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  $x$  a 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 interprète  $P$  par une relation  $n$ -aire sur l'ensemble  $D$  (à quelle condition sur les entrées la formule est vraie).
- Exemples : yeux-bleus, ami, ...

- Validité : 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

- 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 !

## 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



- 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–Maîtriser le langage logique

- 1 Définition du langage
- 2 Structure des formules
- 3 Formules vraies
- 4 Théorie et modélisation**
  - Exemple de modélisation propositionnelle avancée
- 5 Définition récursive sur les formules

# Pourquoi des théories ?

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

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

- 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 :  $\forall x y z, (x + y) + z = x + (y + z)$
  - élément 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 propriétés 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  $p = q \dots$

$$\begin{aligned} &\forall p q, P(p) \wedge P(q) \wedge \neg(p = q) \Rightarrow \\ &\exists d, D(d) \wedge p \in d \wedge q \in d \wedge (\forall d', D(d') \wedge p \in d' \wedge q \in d' \Rightarrow d = d') \end{aligned}$$

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

# Théorie de l'égalité

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

- 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

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

$$\forall x_1 y_1, (x_1 = y_1) \Rightarrow (-x_1) = (-y_1)$$

$$\forall x_1 y_1, (x_1 = y_1) \Rightarrow D(x_1) \Rightarrow D(y_1)$$

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

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

- on pourra écrire  $t \neq u$  pour la formule  $\neg(t = u)$
- pour n'importe quelle formule  $\Phi$  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'équivalence 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
  - $\forall x, S(x) \neq 0$
  - $\forall x, x = 0 \vee \exists y, x = S(y)$  (inutile en présence de récurrence)
  - $\forall x y, S(x) = S(y) \Rightarrow x = y$
  - $\forall x, x + 0 = x$
  - $\forall x y, x + S(y) = S(x + y)$
  - $\forall x, x \times 0 = 0$
  - $\forall x y, x \times S(y) = (x \times y) + x$
- Récurrence (nombre dénombrable d'axiomes, un pour chaque formule  $\Phi$ )

$$\forall x_1 \dots x_n, \Phi[x \leftarrow 0] \Rightarrow (\forall x, \Phi \Rightarrow \Phi[x \leftarrow S(x)]) \Rightarrow \forall x, \Phi$$

- 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  $f \in \mathbb{N} \rightarrow \mathbb{N}$  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  $(x - y) : F[x, y, z] \stackrel{\text{def}}{=} y + z = x \vee x < y \wedge z = 0$
- 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 récurifs



*That's all Folks!*

## Definition (Théorie)

Une théorie est définie par

- un ensemble de symboles de fonctions et de prédicats (la **signature**) et
  - un ensemble de formules **closes** (sans variables libres) construites sur ce langage (les **axiomes**).
- 
- *syntaxe* et *sémantique*
  - 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 :
    - Partir d'un ensemble restreint d'objets de base et de leurs propriétés
    - Construire logiquement les concepts avancés et les théorèmes

# La dernière fois : la théorie de l'égalité

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

- 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  $n$ ) :  
 $\forall x_1 \dots x_n y_1 \dots y_n, (x_1 = y_1 \wedge \dots \wedge x_n = y_n) \Rightarrow P(x_1, \dots, x_n) \Rightarrow P(y_1, \dots, y_n)$

# Propriétés des théories

Soit une théorie  $\mathcal{A}$

- $\mathcal{A}$  est **récursive** s'il existe un algorithme qui étant donné une formule  $P$  permet de calculer si  $P \in \mathcal{A}$
- $\mathcal{A}$  est **cohérente** si elle possède au moins un modèle, en particulier elle ne permet pas de déduire  $\perp$  (n'importe quoi)
- $\mathcal{A}$  est **décidable** s'il existe un algorithme qui étant donné une formule  $P$  permet de savoir si  $P$  est valide dans la théorie ou pas
- $\mathcal{A}$  est **complète** si pour toute formule  $P$ , soit  $P$  est valide dans la théorie, soit  $\neg P$  est valide dans la théorie.

Quelques résultats

- exemples de théories décidables : calcul propositionnel, arithmétique linéaire (Presburger), ordres discrets, théorie des réels...
- une théorie récursive et complète est décidable
- théorème d'incomplétude de Gödel : toute théorie cohérente qui contient l'arithmétique est incomplète

# Modélisation propositionnelle avancée : 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

- position  $p = (i, j)$  sur la ligne  $i$  et la colonne  $j$
- variable propositionnelle  $x_p^k$  : le chiffre  $k$  est à la position  $p$

- donner des ensembles de formules pour les propriétés suivantes :
  - $A(p)$  : au moins un chiffre à la position  $p$
  - $B(p)$  : pas deux chiffres différents à la position  $p$
  - $C(p_1, p_2, \dots, p_9)$  tout chiffre  $k = 1, \dots, 9$  se trouve au moins à l'une des positions  $p_1, p_2, \dots, p_9$
  - $D(p_1, p_2, \dots, p_9)$  pas deux fois le même chiffre à des positions différentes en utilisant  $D'(p, q)$  pour cette propriété concernant deux cases.
- relier la résolution du Sudoku et l'existence de modèles d'un ensemble de formules.



# Modélisation au premier ordre : Sudoku

- La modélisation propositionnelle introduit de nombreuses variables et axiomes.
- On peut considérer les positions et chiffres comme des objets de la logique
- Un symbole de prédicat à trois arguments  $\text{pos}(i, j, k)$  vrai lorsque le chiffre  $k$  est à la position  $(i, j)$  et l'égalité
- Au plus un seul chiffre  $k$  par case  $(i, j)$  :

$$\forall i j k_1 k_2, \text{pos}(i, j, k_1) \wedge \text{pos}(i, j, k_2) \Rightarrow k_1 \neq k_2$$


- Exprimer logiquement que les objets peuvent prendre exactement 9 valeurs différentes sans utiliser l'arithmétique.
- Comment représenter simplement que deux positions sont dans le même cadran ?

- Connaître la définition d'une théorie.
- Connaître la théorie de l'égalité.
- Distinguer modélisation en calcul propositionnel et modélisation en logique du premier ordre.

# 1–Maîtriser le langage logique

- 1 Définition du langage
- 2 Structure des formules
- 3 Formules vraies
- 4 Théorie et modélisation
- 5 **Définition récursive sur les formules**
  - Définir une fonction sur les formules
  - Raisonner sur les formules
  - Définition récursive sur les termes

# Définition récursive sur les formules

- définir des opérations mathématiques sur les formules (taille, substitution, valeur de vérité, transformations ...)
  - les implémenter sur machine
  - équations récursives :
    - une équation (et une seule) pour chaque construction possible de formule :
      - formules atomiques :  $\top$ ,  $\perp$ , prédicat
      - connecteurs propositionnels  $\neg A$ ,  $A \vee B$ ,  $A \wedge B$ ,  $A \Rightarrow B$ .
      - formules quantifiées  $\forall x, A$ ,  $\exists x, A$
    - le membre droit de l'équation peut contenir des appels récursifs à la fonction sur les sous-formules  $A$ ,  $B$
    - correspond à un calcul par parcours de l'arbre
-   $A$  et  $B$  sont des (meta)-variables mathématiques qui représentent des formules quelconques.
- facilite le raisonnement (cf TD)

# Exemple

compter le nombre de connecteurs propositionnels dans une formule :

$$\begin{aligned}\text{nbsymbp}(p) &= \text{si } p \text{ atomique} \\ \text{nbsymbp}(\neg A) &= 1 + \text{nbsymbp}(A) \\ \text{nbsymbp}(A \wedge B) &= \\ \text{nbsymbp}(A \vee B) &= \\ \text{nbsymbp}(A \Rightarrow B) &= \\ \text{nbsymbp}(\forall x, A) &= \\ \text{nbsymbp}(\exists x, A) &= \end{aligned}$$

- fonction `contient-neg` qui teste si une formule comporte au moins une négation
- fonction `nbatom` qui calcule le nombre de formules atomiques dans une formule

# Exemple

compter le nombre de connecteurs propositionnels dans une formule :

$$\begin{aligned}\text{nbsymbp}(p) &= 0 \text{ si } p \text{ atomique} \\ \text{nbsymbp}(\neg A) &= 1 + \text{nbsymbp}(A) \\ \text{nbsymbp}(A \wedge B) &= \\ \text{nbsymbp}(A \vee B) &= \\ \text{nbsymbp}(A \Rightarrow B) &= \\ \text{nbsymbp}(\forall x, A) &= \\ \text{nbsymbp}(\exists x, A) &= \end{aligned}$$

- fonction `contient-neg` qui teste si une formule comporte au moins une négation
- fonction `nbatom` qui calcule le nombre de formules atomiques dans une formule

# Exemple

compter le nombre de connecteurs propositionnels dans une formule :

$$\begin{aligned} \text{nbsymbp}(p) &= 0 \text{ si } p \text{ atomique} \\ \text{nbsymbp}(\neg A) &= 1 + \text{nbsymb}(A) \\ \text{nbsymbp}(A \wedge B) &= \text{nbsymb}(A) + 1 + \text{nbsymb}(B) \\ \text{nbsymbp}(A \vee B) &= \text{nbsymb}(A) + 1 + \text{nbsymb}(B) \\ \text{nbsymbp}(A \Rightarrow B) &= \text{nbsymb}(A) + 1 + \text{nbsymb}(B) \\ \text{nbsymbp}(\forall x, A) &= \text{nbsymb}(A) \\ \text{nbsymbp}(\exists x, A) &= \text{nbsymb}(A) \end{aligned}$$

- fonction `contient-neg` qui teste si une formule comporte au moins une négation
- fonction `nbatom` qui calcule le nombre de formules atomiques dans une formule

# Restriction au calcul propositionnel

- seulement des variables propositionnelles (symboles de prédicat sans argument)
- pas de quantificateur
- Définir **récurivement**  $\text{Vars}(P)$  ensemble des variables propositionnelles qui apparaissent dans  $P$

$$\begin{array}{lll} \text{Vars}(X) & = & X \in \mathcal{V}_p \\ \text{Vars}(\perp) & = & \\ \text{Vars}(\top) & = & \\ \text{Vars}(\neg P) & = & \\ \text{Vars}(P_1 \circ P_2) & = & \circ \in \{\wedge, \vee, \Rightarrow\} \end{array}$$



# Restriction au calcul propositionnel

- seulement des variables propositionnelles (symboles de prédicat sans argument)
- pas de quantificateur
- Définir **récurivement**  $\text{Vars}(P)$  ensemble des variables propositionnelles qui apparaissent dans  $P$

$$\begin{aligned}\text{Vars}(X) &= \{X\} & X \in \mathcal{V}_p \\ \text{Vars}(\perp) &= \emptyset \\ \text{Vars}(\top) &= \emptyset \\ \text{Vars}(\neg P) &= \text{Vars}(P) \\ \text{Vars}(P_1 \circ P_2) &= \text{Vars}(P_1) \cup \text{Vars}(P_2) \quad \circ \in \{\wedge, \vee, \Rightarrow\}\end{aligned}$$

# Substitution

Remplacer une variable propositionnelle  $X$  par une formule  $Q$  dans une formule propositionnelle  $P$  :

$$X[X \leftarrow Q] =$$

$$P[X \leftarrow Q] =$$

$$(\neg A)[X \leftarrow Q] =$$

$$(A \circ B)[X \leftarrow Q] =$$

$$X \in \mathcal{V}_p$$

$$P \text{ est atomique } P \neq X$$

$$\circ \in \{\wedge, \vee, \Rightarrow\}$$

## Exemple

Calculer :  $((\neg x \wedge y) \vee \neg y \Rightarrow x)[x \leftarrow (p \Rightarrow p)]$

# Substitution

Remplacer une variable propositionnelle  $X$  par une formule  $Q$  dans une formule propositionnelle  $P$  :

$$X[X \leftarrow Q] = Q$$

$$X \in \mathcal{V}_p$$

$$P[X \leftarrow Q] = P$$

$$P \text{ est atomique } P \neq X$$

$$(\neg A)[X \leftarrow Q] = \neg(A[X \leftarrow Q])$$

$$(A \circ B)[X \leftarrow Q] = (A[X \leftarrow Q] \circ B[X \leftarrow Q]) \quad \circ \in \{\wedge, \vee, \Rightarrow\}$$

## Exemple

Calculer :  $((\neg x \wedge y) \vee \neg y \Rightarrow x)[x \leftarrow (p \Rightarrow p)]$

$$((\neg(p \Rightarrow p) \wedge y) \vee \neg y \Rightarrow (p \Rightarrow p))$$

# Représentation des formules propositionnelles

```
type connecteur = Impl | Et | Ou
type form = Var of int | Bot | Top
           | Neg of form
           | Bin of form * connecteur * form

let rec vars = function
  Var n -> [n]
  | Bot | Top -> []
  | Neg f -> vars f
  | Bin(f,_,g) -> vars f @ vars g

let rec subst x q = function
  Var n -> if n = x then q else Var n
  | Bot -> Bot | Top -> Top
  | Neg f -> Neg (subst x q f)
  | Bin(f,o,g) -> Bin(subst x q f,o,subst x q g)
```

# Valeur de vérité (formule propositionnelle)

On se donne une interprétation  $I \in \mathcal{V}_p \rightarrow \mathbb{B}$ , et on définit  $\text{val}(I, P) \in \mathbb{B}$

$$\begin{aligned}\text{val}(I, \perp) &= F \\ \text{val}(I, \top) &= V \\ \text{val}(I, x) &= I(x) & x \in \mathcal{V}_p \\ \text{val}(I, \neg A) &= \text{si } \text{val}(I, A) = V \text{ alors } F \text{ sinon } V \\ \text{val}(I, A \wedge B) &= \text{si } \text{val}(I, A) = V \text{ alors } \text{val}(I, B) \text{ sinon } F \\ \text{val}(I, A \vee B) &= \text{si } \text{val}(I, A) = V \text{ alors } V \text{ sinon } \text{val}(I, B) \\ \text{val}(I, A \Rightarrow B) &= \text{si } \text{val}(I, A) = V \text{ alors } \text{val}(I, B) \text{ sinon } V\end{aligned}$$

## Exemple

- $P \stackrel{\text{def}}{=} ((x \Rightarrow y) \Rightarrow y) \Rightarrow x$ ,  $I \stackrel{\text{def}}{=} \{x \mapsto F, y \mapsto V\}$ , calculer  $\text{val}(I, P)$

## Propriété

- La valeur d'une formule ne dépend que de la valeur de l'interprétation pour les variables qui apparaissent dans la formule.

$$\text{val}(I_1, P) = \text{val}(I_2, P) \quad \text{si pour tout } x \in \text{Vars}(P), I_1(x) = I_2(x)$$

# Schéma de récurrence pour les formules

Soit une propriété mathématique  $\phi(P)$  qui dépend d'une formule  $P$ .

- Si on peut montrer que :
  - 1  $\phi(p)$  est vérifiée lorsque  $p$  est une formule atomique (en particulier  $\phi(\top)$  et  $\phi(\perp)$  sont vérifiés)
  - 2 pour une formule  $A$  quelconque, en supposant que  $\phi(A)$  est vérifié, on peut montrer  $\phi(\neg A)$
  - 3 pour des formules  $A$  et  $B$  quelconques, en supposant que  $\phi(A)$  et  $\phi(B)$  sont vérifiées, on peut montrer  $\phi(A \wedge B)$  ainsi que  $\phi(A \vee B)$  et  $\phi(A \Rightarrow B)$
  - 4 pour une formule  $A$  quelconque, et une variable  $x$ , en supposant que  $\phi(A)$  est vérifié, on peut montrer  $\phi(\forall x, A)$  et  $\phi(\exists x, A)$
- Alors on peut en déduire que pour toute formule logique  $P$ ,  $\phi(P)$  est vérifié.

# Exemple

Schéma utile pour montrer des propriétés de fonctions sur les formules définies récursivement.

$\text{nbatom}(p)$  : nombre d'occurrences de sous-formules atomiques

$$\begin{aligned}\text{nbatom}(p) &= 1 \text{ si } p \text{ atomique} \\ \text{nbatom}(\neg A) &= \text{nbatom}(A) \\ \text{nbatom}(\forall x, A) &= \text{nbatom}(A) \\ \text{nbatom}(\exists x, A) &= \text{nbatom}(A) \\ \text{nbatom}(A \wedge B) &= \text{nbatom}(A) + \text{nbatom}(B) \\ \text{nbatom}(A \vee B) &= \text{nbatom}(A) + \text{nbatom}(B) \\ \text{nbatom}(A \Rightarrow B) &= \text{nbatom}(A) + \text{nbatom}(B)\end{aligned}$$

**Lemme :** Pour toute formule  $P$ , on a  $\text{nbatom}(P) \leq 1 + \text{nbsymbp}(P)$ .

# Preuve par récurrence

- propriété  $\phi(P)$  à montrer par récurrence structurale sur  $P$  :  
 $\text{nbatom}(P) \leq 1 + \text{nbsymbp}(P)$ .
- Examen de chacun des cas possibles pour la formule  $P$ 
  - 1  $P$  formule atomique  $p$ . Par définition,  $\text{nbatom}(p) = 1$  et  $\text{nbsymbp}(p) = 0$  et donc  $\text{nbatom}(p) \leq 1 + \text{nbsymbp}(p)$ ,  $\phi(p)$  est vérifié.
  - 2  $P$  est une négation  $\neg A$ ,
    - $A$  quelconque vérifie l'hypothèse de récurrence  $\phi(A)$  ( $\text{nbatom}(A) \leq 1 + \text{nbsymbp}(A)$ ).
    - montrons  $\phi(\neg A)$
    - par définition,  $\text{nbatom}(\neg A) = \text{nbatom}(A)$  et  $\text{nbsymbp}(\neg A) = 1 + \text{nbsymbp}(A)$ .
    - En utilisant l'hypothèse de récurrence on a donc

$$\begin{aligned}\text{nbatom}(\neg A) = \text{nbatom}(A) &\leq 1 + \text{nbsymbp}(A) \\ &= \text{nbsymbp}(\neg A) \leq 1 + \text{nbsymbp}(\neg A)\end{aligned}$$

- donc  $\phi(\neg A)$  est vérifiée.

- 3 pour les autres cas, voir les notes de cours

On a bien examiné tous les cas possibles, on en conclut que pour toute formule logique  $P$ , on a  $\text{nbatom}(P) \leq 1 + \text{nbsymbp}(P)$ .



# Exemple

Schéma utile pour montrer des propriétés de fonctions sur les formules définies récursivement.

- Soit une formule propositionnelle  $Q$  et une variable propositionnelle  $X$ .
- On souhaite montrer que pour toute formule propositionnelle  $P$  telle que  $X \notin \text{Vars}(P)$ , on a  $P[X \leftarrow Q] = P$
- La propriété  $\phi(P)$  à montrer par récurrence structurelle sur  $P$  :  
si  $X \notin \text{Vars}(P)$  alors  $P[X \leftarrow Q] = P$
- Examen de chacun des cas possibles pour la formule  $P$

# Preuve détaillée

## 1 $P$ formule atomique :

Comme  $X \notin \text{Vars}(P)$ , on a que  $P \neq X$  et donc  $P[X \leftarrow Q] = P$  par définition de la substitution

## 2 $P$ de la forme $\neg A$ :

- L'hypothèse de récurrence sur  $A$  nous assure que si  $X \notin \text{Vars}(A)$  alors  $A[X \leftarrow Q] = A$
- Comme  $X \notin \text{Vars}(P)$  et  $\text{Vars}(\neg A) = \text{Vars}(A)$ , on a que  $X \notin \text{Vars}(A)$ .
- L'hypothèse de récurrence nous permet de déduire  $A[X \leftarrow Q] = A$
- Par définition de la substitution  $(\neg A)[X \leftarrow Q] = \neg(A[X \leftarrow Q])$
- On en déduit le résultat attendu :  $(\neg A)[X \leftarrow Q] = \neg A$

## 3 $P$ de la forme $A \circ B$ (les trois connecteurs $\vee, \wedge, \Rightarrow$ se comportent pareil) :

- L'hypothèse de récurrence sur  $A$  nous assure que si  $X \notin \text{Vars}(A)$  alors  $A[X \leftarrow Q] = A$  et de même pour  $B$ , si  $X \notin \text{Vars}(B)$  alors  $B[X \leftarrow Q] = B$
- Comme  $X \notin \text{Vars}(P)$  et  $\text{Vars}(A \circ B) = \text{Vars}(A) \cup \text{Vars}(B)$ , on a que  $X \notin \text{Vars}(A)$  et  $X \notin \text{Vars}(B)$ .
- Par hypothèses de récurrence, on déduit  $A[X \leftarrow Q] = A$  et  $B[X \leftarrow Q] = B$
- Par définition de la substitution  $(A \circ B)[X \leftarrow Q] = (A[X \leftarrow Q]) \circ (B[X \leftarrow Q])$
- On a donc le résultat attendu :  $(A \circ B)[X \leftarrow Q] = A \circ B$

On a examiné tous les cas possibles, on en conclut que pour toute formule propositionnelle  $P$ , on a bien que si  $X \notin \text{Vars}(P)$  alors  $P[X \leftarrow Q] = P$



*That's all Folks!*

- Construire des fonctions sur les termes et les formules en utilisant un système d'équations récursives
- Faire des raisonnements simples par récurrence sur la structure des termes ou des formules

# La dernière fois : définition récursive sur les formules

- définir des opérations mathématiques sur les formules (taille, substitution, valeur de vérité, transformations ...)
- les implémenter sur machine
- équations récursives :
  - une équation (et une seule) pour chaque construction possible de formule :
    - formules atomiques :  $\top$ ,  $\perp$ , prédicat
    - connecteurs propositionnels  $\neg A$ ,  $A \vee B$ ,  $A \wedge B$ ,  $A \Rightarrow B$ .
    - formules quantifiées  $\forall x, A$ ,  $\exists x, A$
  - le membre droit de l'équation peut contenir des appels récursifs à la fonction sur les sous-formules  $A$ ,  $B$
  - correspond à un calcul par parcours de l'arbre



$A$  et  $B$  sont des (meta)-variables mathématiques qui représentent des formules quelconques.

# La dernière fois : exemple

compter le nombre de connecteurs propositionnels dans une formule :

$$\begin{aligned} \text{nbsymbp}(p) &= 0 \text{ si } p \text{ atomique} \\ \text{nbsymbp}(\neg A) &= 1 + \text{nbsymbp}(A) \\ \text{nbsymbp}(A \wedge B) &= \text{nbsymbp}(A) + 1 + \text{nbsymbp}(B) \\ \text{nbsymbp}(A \vee B) &= \text{nbsymbp}(A) + 1 + \text{nbsymbp}(B) \\ \text{nbsymbp}(A \Rightarrow B) &= \text{nbsymbp}(A) + 1 + \text{nbsymbp}(B) \\ \text{nbsymbp}(\forall x, A) &= \text{nbsymbp}(A) \\ \text{nbsymbp}(\exists x, A) &= \text{nbsymbp}(A) \end{aligned}$$

# La dernière fois : schéma de récurrence pour les formules

Soit une propriété mathématique  $\phi(P)$  qui dépend d'une formule  $P$ .

- Si on peut montrer que :

- 1  $\phi(p)$  est vérifiée lorsque  $p$  est une formule atomique
- 2 pour une formule  $A$  quelconque, en supposant que  $\phi(A)$  est vérifié, on peut montrer  $\phi(\neg A)$
- 3 pour des formules  $A$  et  $B$  quelconques, en supposant que  $\phi(A)$  et  $\phi(B)$  sont vérifiées, on peut montrer  $\phi(A \wedge B)$  ainsi que  $\phi(A \vee B)$  et  $\phi(A \Rightarrow B)$
- 4 pour une formule  $A$  quelconque, et une variable  $x$ , en supposant que  $\phi(A)$  est vérifié, on peut montrer  $\phi(\forall x, A)$  et  $\phi(\exists x, A)$

- Alors on peut en déduire que pour toute formule logique  $P$ ,  $\phi(P)$  est vérifié.

# La dernière fois : exemple

$\text{nbatom}(p)$  : nombre d'occurrences de sous-formules atomiques

$$\begin{aligned}\text{nbatom}(p) &= 1 \text{ si } p \text{ atomique} \\ \text{nbatom}(\neg A) &= \text{nbatom}(A) \\ \text{nbatom}(\forall x, A) &= \text{nbatom}(A) \\ \text{nbatom}(\exists x, A) &= \text{nbatom}(A) \\ \text{nbatom}(A \wedge B) &= \text{nbatom}(A) + \text{nbatom}(B) \\ \text{nbatom}(A \vee B) &= \text{nbatom}(A) + \text{nbatom}(B) \\ \text{nbatom}(A \Rightarrow B) &= \text{nbatom}(A) + \text{nbatom}(B)\end{aligned}$$

**Lemme :** Pour toute formule  $P$ , on a  $\text{nbatom}(P) \leq 1 + \text{nbsymbp}(P)$ .



# La dernière fois : preuve par récurrence

- propriété  $\phi(P)$  à montrer par récurrence structurelle sur  $P$  :  
 $\text{nbatom}(P) \leq 1 + \text{nbsymbp}(P)$ .
- Examen de chacun des cas possibles pour la formule  $P$ 
  - 1  $P$  formule atomique  $p$ . Par définition,  $\text{nbatom}(p) = 1$  et  $\text{nbsymbp}(p) = 0$  et donc  $\text{nbatom}(p) \leq 1 + \text{nbsymbp}(p)$ ,  $\phi(p)$  est vérifié.
  - 2  $P$  est une négation  $\neg A$ ,
    - $A$  quelconque vérifie l'hypothèse de récurrence  $\phi(A)$  ( $\text{nbatom}(A) \leq 1 + \text{nbsymbp}(A)$ ).
    - montrons  $\phi(\neg A)$
    - par définition,  $\text{nbatom}(\neg A) = \text{nbatom}(A)$  et  $\text{nbsymbp}(\neg A) = 1 + \text{nbsymbp}(A)$ .
    - En utilisant l'hypothèse de récurrence on a donc

$$\begin{aligned}\text{nbatom}(\neg A) = \text{nbatom}(A) &\leq 1 + \text{nbsymbp}(A) \\ &= \text{nbsymbp}(\neg A) \leq 1 + \text{nbsymbp}(\neg A)\end{aligned}$$

- donc  $\phi(\neg A)$  est vérifiée.

- 3 pour les autres cas, voir les notes de cours

On a bien examiné tous les cas possibles, on en conclut que pour toute formule logique  $P$ , on a  $\text{nbatom}(P) \leq 1 + \text{nbsymbp}(P)$ .

# Exercice TD : Sous-formules

On se restreint aux formules du calcul propositionnel.

On dit qu'une formule  $Q$  est une **sous-formule** de  $P$  si  $Q = P$  ou bien si la formule  $Q$  apparaît sous un connecteur de  $P$ .

C'est-à-dire  $P = \neg P'$  et  $Q$  est une sous-formule de  $P'$  ou bien  $P = P_1 \circ P_2$  et  $Q$  est une sous-formule de  $P_1$  ou bien une sous-formule de  $P_2$  avec  $\circ$  un des connecteurs binaires :  $\{\vee, \wedge, \Rightarrow\}$ .

- 1 Donner toutes les sous-formules de la formule  $\neg(p \vee (q \wedge r)) \Rightarrow (p \wedge q)$
- 2 Donner les équations qui définissent la fonction **sf** qui à une formule propositionnelle  $P$  associe l'ensemble de ses sous-formules.
- 3 Trouver un majorant du nombre de sous-formules d'une formule  $P$  qui utilise  $n$  connecteurs logiques. Donner un exemple où ce majorant est atteint. Prouver ce résultat par récurrence structurelle sur la formule.
- 4 (optionnel) Même question pour un minorant du nombre de sous-formules.

# Définition récursive sur les termes

- Pour traiter le cas des formules atomiques en logique du premier ordre, il faut souvent prendre en compte les termes arguments des prédicats.
- Les termes sont définis à partir de la signature et contiennent possiblement des variables
- Les termes se représentent par des arbres donc chaque nœud est étiqueté par un symbole de fonction, le nombre de sous-arbres est donné par l'arité du symbole et les feuilles sont les constantes et les variables.



- Le même principe de définition récursive de fonctions (mathématiques) s'applique sur les termes.
- Une équation pour les variables et une pour chaque symbole de la signature avec de possibles appels récursifs sur les sous-termes.

$$\begin{aligned} G(x) &= \dots && \text{x variable} \\ G(f(t_1, \dots, t_n)) &= \dots G(t_1) \dots G(t_n) \dots \end{aligned}$$

# Exemple

- Signature : constante  $c$ , fonction unaire  $f$ , fonction binaire  $g$ .
- On définit une fonction  $\text{clos}$  qui étant donné un terme  $t$  teste s'il est clos (pas de variables)

```
 $\text{clos}(x)$       = faux si  $x$  est une variable  
 $\text{clos}(c)$       = vrai  
 $\text{clos}(f(t))$    =  $\text{clos}(t)$   
 $\text{clos}(g(t,u))$  =  $\text{clos}(t)$  et  $\text{clos}(u)$ 
```

# Définition récursive sur les termes

## Definition

$\mathcal{F}$  signature,  $\mathcal{X}$  ensemble de variables et  $\mathcal{D}$  un ensemble quelconque.

Pour définir une application  $G \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \rightarrow \mathcal{D}$ , on se donne :

- 1 Une application  $V$  dans  $\mathcal{X} \rightarrow \mathcal{D}$  ;
- 2 Pour chaque constante  $c \in \mathcal{F}_0$ , un élément  $g_c \in \mathcal{D}$
- 3 Pour chaque symbole de fonction  $f \in \mathcal{F}_n$ , une application  $G_f$  dans

$$\underbrace{\mathcal{T}(\mathcal{F}, \mathcal{X}) \times \dots \times \mathcal{T}(\mathcal{F}, \mathcal{X})}_{n \text{ fois}} \times \underbrace{\mathcal{D} \times \dots \times \mathcal{D}}_{n \text{ fois}} \rightarrow \mathcal{D}$$

Il existe une unique application  $G$  dans  $\mathcal{T}(\mathcal{F}, \mathcal{X}) \rightarrow \mathcal{D}$  qui vérifie :

$$\begin{aligned} G(x) &= V(x) & (x \in \mathcal{X}) \\ G(c) &= g_c & (c \in \mathcal{F}_0) \\ G(f(t_1, \dots, t_n)) &= G_f(t_1, \dots, t_n, G(t_1), \dots, G(t_n)) & (f \in \mathcal{F}_n) \end{aligned}$$

# Fonctions génériques utiles

## Exemple (Taille d'un terme)

*size* compte le nombre de symboles dans un terme.

- si  $x \in \mathcal{X}$  alors  $size(x) = 0$
- si  $c \in \mathcal{F}_0$  alors  $size(c) = 1$
- si  $f \in \mathcal{F}_n$  alors  $size(f(t_1, \dots, t_n)) = 1 + size(t_1) + \dots + size(t_n)$

$t \stackrel{\text{def}}{=} plus(0, S(0))$  vérifie  $size(t) = 4$ .

## Exemple (Hauteur d'un terme)

*ht* : compte le nombre maximal de symboles imbriqués dans un terme.

- si  $x \in \mathcal{X}$  alors  $ht(x) = 0$
- si  $c \in \mathcal{F}_0$  alors  $ht(c) = 1$
- si  $f \in \mathcal{F}_n$  alors  $ht(f(t_1, \dots, t_n)) = 1 + \max(ht(t_1), \dots, ht(t_n))$

$ht(t) = 3$ .

# Exercice : variables d'un terme

## Exercice

Ecrire une fonction `vars` qui prend en argument un terme et renvoie l'ensemble des variables qui apparaissent dans ce terme.

# Exercice : variables d'un terme

## Exercice

Ecrire une fonction `vars` qui prend en argument un terme et renvoie l'ensemble des variables qui apparaissent dans ce terme.

## Solution

- si  $x \in \mathcal{X}$  alors  $\text{vars}(x) = \{x\}$
- si  $c \in \mathcal{F}_0$  alors  $\text{vars}(c) = \emptyset$
- si  $f \in \mathcal{F}_n$  alors  $\text{vars}(f(t_1, \dots, t_n)) = \text{vars}(t_1) \cup \dots \cup \text{vars}(t_n)$



# Substitution sur les termes

- remplacement simultanée de plusieurs variables par des termes.
- application  $\sigma \in \mathcal{X} \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$ , appelée **substitution** qui associe un terme à chaque variable.
- On note  $\{x_1 \leftarrow u_1; \dots; x_n \leftarrow u_n\}$  la substitution  $\sigma$  telle que  $\sigma(x) = u_i$  si  $x = x_i$  et  $\sigma(x) = x$  sinon.
- On définit pour chaque terme  $t$ , le résultat de la **substitution** dans  $t$  de toute variable  $x$  par  $\sigma(x)$  que l'on note  $t[\sigma]$ .
- Si  $\sigma$  est de la forme  $\{x_1 \leftarrow u_1; \dots; x_n \leftarrow u_n\}$ , alors le terme  $t[\sigma]$  sera noté  $t[x_1 \leftarrow u_1; \dots; x_n \leftarrow u_n]$ .

# Définition et exemple

La définition de  $t[\sigma]$  se fait de manière récursive sur  $t$  :

- si  $x \in \mathcal{X}$  alors  $x[\sigma] = \sigma(x)$
- si  $c \in \mathcal{F}_0$  alors  $c[\sigma] = c$
- si  $f \in \mathcal{F}_n$  alors  $f(t_1, \dots, t_n)[\sigma] = f(t_1[\sigma], \dots, t_n[\sigma])$

## Exemple

$t = \text{plus}(\text{mult}(x, y), S(x))$  et  $\sigma = \{x \leftarrow \text{mult}(y, 0); y \leftarrow 0\}$ .  
On a  $t[\sigma] = \text{plus}(\text{mult}(\text{mult}(y, 0), 0), S(\text{mult}(y, 0)))$

# Propriétés de la substitution

- le résultat de  $t[\sigma]$  ne dépend que de la valeur de la substitution  $\sigma$  sur les variables de  $t$ .
- soit deux substitutions  $\sigma_1$  et  $\sigma_2$  et un terme  $t$ , si pour toute variable  $x \in \text{vars}(t)$  on a  $\sigma_1(x) = \sigma_2(x)$  alors  $t[\sigma_1] = t[\sigma_2]$ .
- La preuve se fait aisément par récurrence structurelle sur le terme  $t$  suivant le schéma ci-dessous.

# Récurrance sur les termes

On peut utiliser un schéma de preuve par récurrence sur les termes de  $\mathcal{T}(\mathcal{F}, \mathcal{X})$

Soit  $\phi(t)$  une propriété mathématique qui dépend d'un terme  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . Si :

- 1 pour toute variable  $x \in \mathcal{X} : \phi(x)$  ;
- 2 pour chaque constante  $c \in \mathcal{F}_0 : \phi(c)$  ;
- 3 pour chaque symbole  $f \in \mathcal{F}_n$  : pour tous termes  $t_1 \dots t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  si les propriétés  $\phi(t_1) \dots \phi(t_n)$  sont vérifiées (hypothèses de récurrence), alors il en est de même de  $\phi(f(t_1, \dots, t_n))$  ;

alors pour tout terme  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  la propriété  $\phi(t)$  est vérifiée.

# Exemple

On montre pour tout terme  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  que  $\text{ht}(t) \leq \text{size}(t)$

La preuve se fait par récurrence sur la structure du terme  $t$ .

La propriété à montrer est  $\phi(t) \stackrel{\text{def}}{=} \text{ht}(t) \leq \text{size}(t)$

**variable** soit  $x \in \mathcal{X}$ ,  $\text{ht}(x) \leq \text{size}(x)$  vrai car  $\text{ht}(x) = 0$  et  $\text{size}(x) = 0$

**constante**  $\text{ht}(c) \leq \text{size}(c)$  vrai car  $\text{ht}(c) = 1 = \text{size}(c)$ .

**symbole** si  $f \in \mathcal{F}_n$  et  $t_1, \dots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  sont des termes quelconques qui vérifient l'hypothèse de récurrence  $\text{ht}(t_i) \leq \text{size}(t_i)$ . On doit montrer  $\text{ht}(f(t_1, \dots, t_n)) \leq \text{size}(f(t_1, \dots, t_n))$ .

$$\begin{aligned} \text{ht}(f(t_1, \dots, t_n)) &= 1 + \max(\text{ht}(t_1), \dots, \text{ht}(t_n)) && \text{(déf de ht)} \\ &\leq 1 + \text{ht}(t_1) + \dots + \text{ht}(t_n) && \text{(car ht}(t_i) \geq 0) \\ &\leq 1 + \text{size}(t_1) + \dots + \text{size}(t_n) && \text{(hyp. de récurrence)} \\ &= \text{size}(f(t_1, \dots, t_n)) && \text{(déf de size)} \end{aligned}$$

On en déduit que  $\text{ht}(t) \leq \text{size}(t)$  est vérifié pour tout les termes du langage.

# Substitution sur les formules

- substitution  $P[x \leftarrow t]$  d'une variable  $x$  par un terme  $t$  dans une formule  $P$
- éviter la capture d'une variable du terme  $t$  par un des quantificateurs interne de  $P$ .
- définition récursive de  $P[x \leftarrow t]$ 
  - Formules atomiques :
    - $\perp[x \leftarrow t] = \perp$
    - $\top[x \leftarrow t] = \top$
    - $R(t_1, \dots, t_n)[x \leftarrow t] = R(t_1[x \leftarrow t], \dots, t_n[x \leftarrow t])$
  - $(\neg A)[x \leftarrow t] = \neg(A[x \leftarrow t])$
  - $\circ \in \{\wedge, \vee, \Rightarrow\} : (A \circ B)[x \leftarrow t] = (A[x \leftarrow t]) \circ (B[x \leftarrow t])$
  - $(\forall y, Q)[x \leftarrow t]$  :
    - si  $y = x$  alors  $(\forall y, Q) = \forall x, Q$  et comme  $x$  n'est pas libre dans  $\forall x, Q$  on a  $(\forall y, Q)[x \leftarrow t] = (\forall x, Q)[x \leftarrow t] = \forall x, Q$
    - si  $y \neq x$  et  $y \notin \text{vars}(t)$  alors  $(\forall y, Q)[x \leftarrow t] = \forall y, (Q[x \leftarrow t])$   
pas de risque de capture puisque la variable liée  $y$  n'apparaît pas dans  $t$
  - $(\exists y, Q)[x \leftarrow t]$  : traitement analogue à  $(\forall y, Q)[x \leftarrow t]$

# Exemples

- 1  $(\forall y, R(x, y))[x \leftarrow f(x)]$
- 2  $(\forall y, R(x, y))[x \leftarrow f(y)]$
- 3  $(\forall y, R(x, y))[y \leftarrow x]$

- La définition est *partielle*, elle n'est pas définie dans le cas des formules avec quantificateurs si une variable liée dans la formule apparaît aussi dans le terme que l'on veut substituer.
- En procédant à un renommage des variables liées dans les quantificateurs, on se ramène à une situation dans laquelle la substitution sera possible.



- Savoir construire des fonctions sur les termes et les formules en utilisant un système d'équations récursives.
- Faire des raisonnements simples par récurrence sur la structure des termes ou des formules.
- Savoir calculer la **substitution** d'une variable (libre) par un terme dans une formule en évitant les problèmes de capture.

## 2—Donner du sens aux formules

- 1 Interprétations et vérité
- 2 Validité et satisfiabilité
- 3 Conséquence logique, équivalence
- 4 Modèles particuliers

- Chapitre 1 : Les bases de la logique du premier ordre :
  - structure des termes et des formules
  - comment utiliser ce langage pour modéliser des situations ou problèmes
  - la vérité d'une formule logique dépend du monde dans lequel on interprète les symboles de la signature
- Chapitre 2 :
  - retour sur la notion d'interprétation de manière plus détaillée
  - définition de la notion de conséquence logique et d'équivalence.
  - étude de quelques modèles particuliers.

# Interprétations et vérité

## signature $(\mathcal{F}, \mathcal{R})$

- $\mathcal{F}$  l'ensemble des symboles de fonctions (termes)
- $\mathcal{R}$  l'ensemble des symboles de prédicat (formules atomiques).

## Definition (Interprétation)

Une **interprétation**  $I$  de la signature  $(\mathcal{F}, \mathcal{R})$  est donnée par

- un ensemble  $\mathcal{D}$  *non vide* appelé **domaine** de l'interprétation ;
- pour chaque symbole de fonction  $f \in \mathcal{F}_n$  d'arité  $n$ , une fonction  $f_I$   $n$ -aire sur  $\mathcal{D}$  (c'est à dire  $f_I : \mathcal{D}^n \rightarrow \mathcal{D}$  : pour chaque  $v_1, \dots, v_n \in \mathcal{D}$ , on a  $f_I(v_1, \dots, v_n) \in \mathcal{D}$ ) ;
- pour chaque symbole de prédicat  $R \in \mathcal{R}_n$  d'arité  $n$ , une relation  $n$ -aire  $R_I$  sur  $\mathcal{D}$  ( $R_I \subseteq \mathcal{D}^n$ ).  
 $R_I$  ensemble de  $n$ -uplets  $(v_1, \dots, v_n)$  avec  $v_i \in \mathcal{D}$  qui correspondent au cas où, *dans cette interprétation*, la relation  $R$  est vraie.

On utilise également la terminologie de **modèle** ou de **structure** pour parler de l'interprétation d'une signature.

# Exemple : interprétation

Signature : prédicat binaire  $P$ .

Domaine  $D \stackrel{\text{def}}{=} \{1, 2, 3, 4\}$ .

L'interprétation de  $P$  donnée par un tableau  $4 \times 4$ .

La formule atomique  $P(t, u)$  est vraie si  $t$  vaut  $i$  et  $u$  vaut  $j$  et que la case sur la ligne  $i$  et la colonne  $j$  est grisée.

Soient les formules :

- ①  $\exists x, \neg P(x, x)$
- ②  $\exists x, \forall y, P(x, y)$
- ③  $\forall x, \exists y, P(x, y)$
- ④  $\forall x y, P(x, y) \Rightarrow P(y, x)$

Donner la valeur des formules pour chaque interprétation de  $P$  :


(1)


(2)


(3)


(4)

## Exemple des rationnels

- signature pour manipuler des rationnels
  - constantes 0, 1 et -1,
  - une opération binaire de construction `frac`,
  - des opérations binaires `+` et `*`
  - un symbole de prédicat binaire pour l'égalité.
- interprétation de cette signature : représentation d'un rationnel
  - un couple  $(p, q) \in \mathbb{Z} \times \mathbb{N} \setminus \{0\}$  formé d'un entier relatif en numérateur et d'un entier naturel non nul en dénominateur.
  - Le domaine de l'interprétation est donc  $\mathcal{D} \stackrel{\text{def}}{=} \mathbb{Z} \times \mathbb{N} \setminus \{0\}$ .
- Autres domaines possibles
  - imposer que la fraction soit réduite (pas de diviseur commun du dénominateur et du numérateur),
  - autoriser un entier relatif en dénominateur

# Interprétation de chaque symbole

$$\begin{aligned}0_{\mathbb{Q}} &\stackrel{\text{def}}{=} (0, 1) \\1_{\mathbb{Q}} &\stackrel{\text{def}}{=} (1, 1) \\-1_{\mathbb{Q}} &\stackrel{\text{def}}{=} (-1, 1) \\\text{frac}_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (p_1 q_2, q_1 p_2) && \text{si } p_2 > 0 \\\text{frac}_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (0, 1) && \text{si } p_2 = 0 \\\text{frac}_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (-p_1 q_2, -q_1 p_2) && \text{si } p_2 < 0 \\+_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (p_1 q_2 + p_2 q_1, q_1 q_2) \\*_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (p_1 p_2, q_1 q_2) \\=_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (p_1 q_2 = q_1 p_2)\end{aligned}$$

- Aux tables primitives de la BD correspondent des symboles de prédicat (l'arité est le nombre de colonnes)
- Chaque état de la base de données correspond à une interprétation particulière des symboles.



# Cas propositionnel

- Dans le cas propositionnel (sans quantificateur), il n'y a pas de symboles de fonctions et tous les symboles de prédicats sont d'arité 0, donc des variables propositionnelles.
- Une interprétation revient donc à fixer une valeur booléenne pour chacune de ces variables.
- S'il y a  $n$  variables propositionnelles alors il y a  $2^n$  interprétations possibles.
- On peut les énumérer toutes et calculer la valeur de vérité de la formule propositionnelle pour chacune de ces interprétations, on retrouve ainsi les tables de vérité.

# Interprétation des symboles de prédicats

- Symbole de prédicat d'arité 0 : vrai, ou faux (*barrière\_ouverte*)
- Symbole de prédicat d'arité 1 : sous-ensemble de  $\mathcal{D}$  (les objets qui vérifient le prédicat).
- Symbole  $R$  de prédicat d'arité 2 : relation binaire sur  $\mathcal{D}$ .  
Représentation par un graphe (fini ou infini) : sommets éléments de  $\mathcal{D}$ , arête entre deux sommets  $a$  et  $b$  ssi  $R_I(a, b)$  est vraie.  
Représentation par une matrice carrée sur les sommets à valeur dans  $\{0, 1\}$ .
- Prédicat  $R$  d'arité plus grande, par exemple 4 alors l'interprétation est un ensemble de quadruplets  $(a, b, c, d)$ .  
Si l'ensemble est fini, on peut utiliser une table avec 4 colonnes.  
Les lignes de la table contiennent les quadruplets  $(a, b, c, d)$  pour lesquels l'interprétation  $R_I(a, b, c, d)$  est vraie.

La vérité d'une formule qui contient des variables libres dépend de la *valeur des variables*

**Exemple** Dans le modèle usuel des entiers :  $\exists n, \text{pair}(3 \times n + 1)$  est vrai,  $\forall n, \text{pair}(3 \times n + 1)$  est faux mais la valeur de vérité de  $\text{pair}(3 \times n + 1)$  dépend de la valeur de  $n$ .

## Definition (Environnement)

Soit  $\mathcal{X}$  l'ensemble des variables d'objets. Soit  $I$  une interprétation de la signature  $(\mathcal{F}, \mathcal{R})$  dont le domaine est  $\mathcal{D}$ , un **environnement** est une application  $\rho$  qui associe une valeur du domaine  $\mathcal{D}$  à chaque variable de  $\mathcal{X}$  (c'est à dire  $\rho : \mathcal{X} \rightarrow \mathcal{D}$ ).

Soit  $\rho$  un environnement, si  $x \in \mathcal{X}$  et  $d \in \mathcal{D}$ , on note  $\rho + \{x \mapsto d\}$  l'environnement qui vaut  $d$  pour la variable  $x$  et  $\rho(y)$  pour toutes les variables  $y \neq x$ .

# Valeur d'un terme dans une interprétation

- une signature  $(\mathcal{F}, \mathcal{R})$ , un ensemble de variables  $\mathcal{X}$
- une interprétation  $I \stackrel{\text{def}}{=} (\mathcal{D}, (f_I)_{f \in \mathcal{F}}, (R_I)_{R \in \mathcal{R}})$

## Definition (Valeur d'un terme)

Soit  $\rho$  un environnement,  $\rho \in \mathcal{X} \rightarrow \mathcal{D}$ .

On définit la valeur  $\text{val}_I(\rho, t)$  d'un terme  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  dans l'interprétation  $I$  et l'environnement  $\rho$ , c'est un élément du domaine  $\mathcal{D}$  :

$$\text{val}_I(\rho, x) = \rho(x) \quad \text{val}_I(\rho, f(t_1, \dots, t_n)) = f_I(\text{val}_I(\rho, t_1), \dots, \text{val}_I(\rho, t_n))$$

# Valeur d'une formule dans une interprétation

## Definition (Valeur d'une formule)

Soit  $\rho$  un environnement,  $\rho : \mathcal{X} \rightarrow \mathcal{D}$ .

On définit la valeur  $\text{val}_I(\rho, P)$  d'une formule  $P$  dans l'interprétation  $I$  et l'environnement  $\rho$ , c'est une valeur de vérité dans  $\{V, F\}$  :

$\text{val}_I(\rho, \perp)$	$= F$
$\text{val}_I(\rho, \top)$	$= V$
$\text{val}_I(\rho, R(t_1, \dots, t_n))$	$= \text{si } R_I(\text{val}_I(\rho, t_1), \dots, \text{val}_I(\rho, t_n)) \text{ alors } V \text{ sinon } F$
$\text{val}_I(\rho, \neg A)$	$= \text{si } \text{val}_I(\rho, A) = V \text{ alors } F \text{ sinon } V$
$\text{val}_I(\rho, A \wedge B)$	$= \text{si } \text{val}_I(\rho, A) = V \text{ alors } \text{val}_I(\rho, B) \text{ sinon } F$
$\text{val}_I(\rho, A \vee B)$	$= \text{si } \text{val}_I(\rho, A) = V \text{ alors } V \text{ sinon } \text{val}_I(\rho, B)$
$\text{val}_I(\rho, A \Rightarrow B)$	$= \text{si } \text{val}_I(\rho, A) = V \text{ alors } \text{val}_I(\rho, B) \text{ sinon } V$
$\text{val}_I(\rho, \forall x, A)$	$= \text{si pour tout } d \in \mathcal{D}, \text{val}_I(\rho + \{x \mapsto d\}, A) = V$ $\text{alors } V \text{ sinon } F$
$\text{val}_I(\rho, \exists x, A)$	$= \text{si il existe } d \in \mathcal{D} \text{ tel que } \text{val}_I(\rho + \{x \mapsto d\}, A) = V$ $\text{alors } V \text{ sinon } F$



*That's all Folks!*

# La dernière fois : Interprétations

## signature $(\mathcal{F}, \mathcal{R})$

- $\mathcal{F}$  l'ensemble des symboles de fonctions (termes)
- $\mathcal{R}$  l'ensemble des symboles de prédicat (formules atomiques).

## Definition (Interprétation)

Une **interprétation**  $I$  de la signature  $(\mathcal{F}, \mathcal{R})$  est donnée par

- un ensemble  $\mathcal{D}$  *non vide* appelé **domaine** de l'interprétation ;
- pour chaque symbole de fonction  $f \in \mathcal{F}_n$  d'arité  $n$ , une fonction  $f_I$   $n$ -aire sur  $\mathcal{D}$  (c'est à dire  $f_I : \mathcal{D}^n \rightarrow \mathcal{D}$  : pour chaque  $v_1, \dots, v_n \in \mathcal{D}$ , on a  $f_I(v_1, \dots, v_n) \in \mathcal{D}$ ) ;
- pour chaque symbole de prédicat  $R \in \mathcal{R}_n$  d'arité  $n$ , une relation  $n$ -aire  $R_I$  sur  $\mathcal{D}$  ( $R_I \subseteq \mathcal{D}^n$ ).  
 $R_I$  ensemble de  $n$ -uplets  $(v_1, \dots, v_n)$  avec  $v_i \in \mathcal{D}$  qui correspondent au cas où, *dans cette interprétation*, la relation  $R$  est vraie.

On utilise également la terminologie de **modèle** ou de **structure** pour parler de l'interprétation d'une signature.

# La dernière fois : Exemple des rationnels

- signature pour manipuler des rationnels
  - constantes 0, 1 et -1,
  - une opération binaire de construction `frac`,
  - des opérations binaires `+` et `*`
  - un symbole de prédicat binaire pour l'égalité.
- interprétation de cette signature : représentation d'un rationnel
  - un couple  $(p, q) \in \mathbb{Z} \times \mathbb{N} \setminus \{0\}$  formé d'un entier relatif en numérateur et d'un entier naturel non nul en dénominateur.
  - Le domaine de l'interprétation est donc  $\mathcal{D} \stackrel{\text{def}}{=} \mathbb{Z} \times \mathbb{N} \setminus \{0\}$ .
- Autres domaines possibles
  - imposer que la fraction soit réduite (pas de diviseur commun du dénominateur et du numérateur),
  - autoriser un entier relatif en dénominateur



# La dernière fois : Interprétation de chaque symbole

$$\begin{aligned}0_{\mathbb{Q}} &\stackrel{\text{def}}{=} (0, 1) \\1_{\mathbb{Q}} &\stackrel{\text{def}}{=} (1, 1) \\-1_{\mathbb{Q}} &\stackrel{\text{def}}{=} (-1, 1) \\ \text{frac}_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (p_1 q_2, q_1 p_2) && \text{si } p_2 > 0 \\ \text{frac}_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (0, 1) && \text{si } p_2 = 0 \\ \text{frac}_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (-p_1 q_2, -q_1 p_2) && \text{si } p_2 < 0 \\ +_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (p_1 q_2 + p_2 q_1, q_1 q_2) \\ *_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (p_1 p_2, q_1 q_2) \\ =_{\mathbb{Q}}((p_1, q_1), (p_2, q_2)) &\stackrel{\text{def}}{=} (p_1 q_2 = q_1 p_2)\end{aligned}$$

# La dernière fois : Environnement et évaluations

## Definition (Environnement)

Soit  $\mathcal{X}$  l'ensemble des variables d'objets. Soit  $I$  une interprétation de la signature  $(\mathcal{F}, \mathcal{R})$  dont le domaine est  $\mathcal{D}$ , un **environnement** est une application  $\rho$  qui associe une valeur du domaine  $\mathcal{D}$  à chaque variable de  $\mathcal{X}$  (c'est à dire  $\rho : \mathcal{X} \rightarrow \mathcal{D}$ ).

Soit  $\rho$  un environnement, si  $x \in \mathcal{X}$  et  $d \in \mathcal{D}$ , on note  $\rho + \{x \mapsto d\}$  l'environnement qui vaut  $d$  pour la variable  $x$  et  $\rho(y)$  pour toutes les variables  $y \neq x$ .

## Definition (Valeur d'un terme)

Soit  $\rho$  un environnement,  $\rho \in \mathcal{X} \rightarrow \mathcal{D}$ .

On définit la valeur  $\text{val}_I(\rho, t)$  d'un terme  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  dans l'interprétation  $I$  et l'environnement  $\rho$ , c'est un élément du domaine  $\mathcal{D}$  :

$$\text{val}_I(\rho, x) = \rho(x) \quad \text{val}_I(\rho, f(t_1, \dots, t_n)) = f_I(\text{val}_I(\rho, t_1), \dots, \text{val}_I(\rho, t_n))$$

## Definition (Valeur d'une formule)

Soit  $\rho$  un environnement,  $\rho : \mathcal{X} \rightarrow \mathcal{D}$ .

On définit la valeur  $\text{val}_I(\rho, P)$  d'une formule  $P$  dans l'interprétation  $I$  et l'environnement  $\rho$ , c'est une valeur de vérité dans  $\{V, F\}$  :

$\text{val}_I(\rho, \perp)$	$= F$
$\text{val}_I(\rho, \top)$	$= V$
$\text{val}_I(\rho, R(t_1, \dots, t_n))$	$= \text{si } R_I(\text{val}_I(\rho, t_1), \dots, \text{val}_I(\rho, t_n)) \text{ alors } V \text{ sinon } F$
$\text{val}_I(\rho, \neg A)$	$= \text{si } \text{val}_I(\rho, A) = V \text{ alors } F \text{ sinon } V$
$\text{val}_I(\rho, A \wedge B)$	$= \text{si } \text{val}_I(\rho, A) = V \text{ alors } \text{val}_I(\rho, B) \text{ sinon } F$
$\text{val}_I(\rho, A \vee B)$	$= \text{si } \text{val}_I(\rho, A) = V \text{ alors } V \text{ sinon } \text{val}_I(\rho, B)$
$\text{val}_I(\rho, A \Rightarrow B)$	$= \text{si } \text{val}_I(\rho, A) = V \text{ alors } \text{val}_I(\rho, B) \text{ sinon } V$
$\text{val}_I(\rho, \forall x, A)$	$= \text{si pour tout } d \in \mathcal{D}, \text{val}_I(\rho + \{x \mapsto d\}, A) = V$ $\text{alors } V \text{ sinon } F$
$\text{val}_I(\rho, \exists x, A)$	$= \text{si il existe } d \in \mathcal{D} \text{ tel que } \text{val}_I(\rho + \{x \mapsto d\}, A) = V$ $\text{alors } V \text{ sinon } F$

# Notation $I, \rho \models A$

Une notation plus intuitive :

- On note  $I, \rho \models A$  lorsque la formule  $A$  est vraie dans l'interprétation  $I$  et l'environnement  $\rho$ , c'est-à-dire lorsque  $\text{val}_I(\rho, A) = V$ .

Répetons :  $I, \rho \models A \stackrel{\text{def}}{=} \text{val}_I(\rho, A) = V$

- On note  $I, \rho \not\models A$  dans le cas contraire lorsque la formule  $A$  est fausse dans l'interprétation  $I$  et l'environnement  $\rho$ , c'est-à-dire lorsque  $\text{val}_I(\rho, A) = F$ .
- Si la formule  $A$  est **close**, alors sa valeur dans une interprétation ne dépend pas de l'environnement et on écrira simplement  $I \models A$  pour indiquer que  $A$  est vraie dans l'interprétation  $I$  (et n'importe quel environnement).

# Propriétés de $I, \rho \models A$

## Proposition

Soit  $I$  une interprétation  $I$  et  $\rho$  un environnement.

- $I, \rho \not\models \perp$
- $I, \rho \models \top$
- $I, \rho \models R(t_1, \dots, t_n)$  si et seulement si  $(val_I(\rho, t_1), \dots, val_I(\rho, t_n))$  appartient à l'interprétation de  $R$
- $I, \rho \models \neg A$  si et seulement si  $I, \rho \not\models A$
- $I, \rho \models A \wedge B$  si et seulement si  $I, \rho \models A$  et  $I, \rho \models B$
- $I, \rho \models A \vee B$  si et seulement si  $I, \rho \models A$  ou  $I, \rho \models B$
- $I, \rho \models A \Rightarrow B$  si et seulement si  $I, \rho \not\models A$  ou  $I, \rho \models B$
- $I, \rho \models \forall x, A$  si et seulement si pour tout  $d \in \mathcal{D}$ , on a  $I, \rho + \{x \mapsto d\} \models A$
- $I, \rho \models \exists x, A$  si et seulement s'il existe  $d \in \mathcal{D}$  tel que  $I, \rho + \{x \mapsto d\} \models A$

**Preuve:** Reformulation de la définition de la valeur d'une formule et du fait que  $I, \rho \models A$  est défini comme  $val_I(\rho, A) = V$  □

# Exercice

Soient les formules suivantes sans variable libre. Sont-elles vraies si on les interprète dans  $\mathbb{N}$ ,  $\mathbb{Z}$ ,  $\mathbb{Q}$  avec les conventions usuelles pour l'interprétation des opérations et des relations ?

①  $\forall x, \exists y, x < y$

②  $\forall x, \exists y, y < x$

③  $\forall x y, x < y \Rightarrow x + 1 \leq y$

④  $\forall x y z, x \leq y \Rightarrow x \times z \leq y \times z$

- des propriétés vraies ou fausses dans une interprétation ne le sont pas forcément dans une autre, même si les interprétations se correspondent sur les formules atomiques.
- les quantifications ne font pas référence aux mêmes ensembles sous-jacents.
- En mathématiques “usuelles” (implicitement la théorie des ensembles) on distingue les formules en *relativisant* les quantificateurs.  
Par exemple  $\forall x \in \mathbb{N}, \exists y \in \mathbb{N}, y < x$  (qui est faux) versus  $\forall x \in \mathbb{Z}, \exists y \in \mathbb{Z}, y < x$  qui est vrai.

# Propriétés de la valeur d'une formule

- La définition de la valeur de vérité d'une formule ne dépend pas de l'opération de renommage des variables liées dans les formules :  
Si  $y \notin \text{vl}(P)$  et  $P[x \leftarrow y]$  est bien définie alors  
 $\text{val}_I(\rho, (\forall x, P)) = \text{val}_I(\rho, (\forall y, P[x \leftarrow y]))$  et  
 $\text{val}_I(\rho, (\exists x, P)) = \text{val}_I(\rho, (\exists y, P[x \leftarrow y]))$
- La **valeur d'un terme** ne dépend que de la valeur de l'environnement sur les variables de ce terme et de l'interprétation des symboles de fonction et des constantes qui apparaissent dans ce terme.
- La **valeur de vérité d'une formule** ne dépend que de la valeur de l'environnement sur les **variables libres** de cette formule et de l'interprétation des symboles de fonction, constantes et relations qui apparaissent dans la formule.  
**Preuve:** La preuve se fait sans difficulté par récurrence sur la structure du terme puis de la formule. □
- Pour un terme clos ou une formule close, la valeur est indépendante de l'environnement :  
on écrira  $\text{val}_I(P)$  au lieu de  $\text{val}_I(\rho, P)$  et  $I \models P$  au lieu de  $I, \rho \models P$ .

## 2—Donner du sens aux formules

- 1 Interprétations et vérité
  - Lien entre substitution et valeur de vérité
- 2 Validité et satisfiabilité
- 3 Conséquence logique, équivalence
- 4 Modèles particuliers



# Substitution versus environnement



Il ne faut pas confondre une **substitution** qui associe un terme *syntaxique* à une variable avec un **environnement** qui associe à une variable une valeur *sémantique* du domaine d'interprétation.

- la substitution est analogue à une opération de remplacement que l'on peut faire dans un éditeur,
- l'environnement correspond à l'état de la mémoire pour chaque variable du programme au moment de son exécution

# Lien entre substitution et valeur de vérité

- formule  $P$  qui contient une variable libre  $x$  et un terme  $t$ .
  - 1 remplacer  $x$  par  $t$  dans  $P$  puis calculer la valeur de  $P[x \leftarrow t]$
  - 2 évaluer le terme  $t$  en une valeur  $v$  puis calculer la valeur de  $P$  (qui contient la variable  $x$ ) dans un environnement où  $x$  a la valeur  $v$ .

## Proposition (Vérité et substitution)

*Soit une formule  $P$  qui contient une variable libre  $x$  et soit  $t$  un terme.  
On suppose que la formule substituée  $P[x \leftarrow t]$  est définie (pas de capture).  
Pour tout environnement  $\rho$  :*

$$val_I(\rho, P[x \leftarrow t]) = val_I(\rho + \{x \mapsto val_I(\rho, t)\}, P)$$

$$I, \rho \models P[x \leftarrow t] \text{ ssi } I, (\rho + \{x \mapsto val_I(\rho, t)\}) \models P$$

Analogie avec les liaisons locales : **let  $x = t$  in  $A$**

On montre un résultat analogue pour les termes. Soit  $u$  un terme, on a :

$$\text{val}_I(\rho, u[x \leftarrow t]) = \text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, u)$$

(propriété  $\phi(u)$  montrée par récurrence sur la structure du terme  $u$ )

- variable  $x$  : alors  $x[x \leftarrow t] = t$   
 $\text{val}_I(\rho, t) = (\rho + \{x \mapsto \text{val}_I(\rho, t)\})(x) = \text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, x)$
- variable  $y \neq x$  :  $y[x \leftarrow t] = y$   
 $\text{val}_I(\rho, y) = \rho(y) = \text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, y)$
- constante  $c$  : alors  $c[x \leftarrow t] = c$   
 $\text{val}_I(\rho, c) = c = \text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, c)$
- $f(u_1, \dots, u_n)$  : alors  $f(u_1, \dots, u_n)[x \leftarrow t] = f(u_1[x \leftarrow t], \dots, u_n[x \leftarrow t])$   
 $\text{val}_I(\rho, f(u_1[x \leftarrow t], \dots, u_n[x \leftarrow t]))$   
 $= f_I(\text{val}_I(\rho, u_1[x \leftarrow t]), \dots, \text{val}_I(\rho, u_n[x \leftarrow t])) \quad (\text{val}_I)$   
 $= f_I(\text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, u_1), \dots, \text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, u_n)) \quad (\text{HR})$   
 $= \text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, f(u_1, \dots, u_n)) \quad (\text{val}_I)$

# Preuve II

Preuve par récurrence sur la structure de la formule  $P$  de l'énoncé  $(\phi(P))$  :  
pour tout environnement  $\rho$ ,  $\text{val}_I(\rho, P[x \leftarrow t]) = \text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, P)$

- formules atomiques  $R(u_1, \dots, u_n)$  :

$$\begin{aligned} & \text{val}_I(\rho, R(u_1, \dots, u_n)[x \leftarrow t]) \\ &= \text{val}_I(\rho, R(u_1[x \leftarrow t], \dots, u_n[x \leftarrow t])) && (\text{subst}) \\ &= R_I(\text{val}_I(\rho, u_1[x \leftarrow t]), \dots, \text{val}_I(\rho, u_n[x \leftarrow t])) && (\text{val}_I) \\ &= R_I(\text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, u_1), \dots, \text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, u_n)) && (\text{termes}) \\ &= \text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, R(u_1, \dots, u_n)) && (\text{val}_I) \end{aligned}$$

- formule  $\forall z, A$  : avec  $z \neq x$  et  $z \notin \text{vars}(t)$  :  $(\forall z, A)[x \leftarrow t] = \forall z, (A[x \leftarrow t])$ .  
soit  $d \in \mathcal{D}$ ,

$$\begin{aligned} & \text{val}_I(\rho + \{z \mapsto d\}, A[x \leftarrow t]) \\ &= \text{val}_I((\rho + \{z \mapsto d\}) + \{x \mapsto \text{val}_I(\rho + \{z \mapsto d\}, t)\}, A) && (\text{HR}) \\ &= \text{val}_I((\rho + \{x \mapsto \text{val}_I(\rho, t)\}) + \{z \mapsto d\}, A) && (x \neq z, z \notin \text{vars}(t)) \end{aligned}$$

$$\text{val}_I(\rho + \{z \mapsto d\}, A[x \leftarrow t]) = V \text{ ssi}$$

$$\text{val}_I((\rho + \{x \mapsto \text{val}_I(\rho, t)\}) + \{z \mapsto d\}, A) = V$$

$$\text{donc } \text{val}_I(\rho, (\forall z, (A[x \leftarrow t]))) = V \text{ ssi}$$

$$\text{val}_I(\rho + \{x \mapsto \text{val}_I(\rho, t)\}, (\forall z, A)) = V, \text{ qui est le résultat souhaité.}$$

l'HR est appliquée à  $\rho + \{z \mapsto d\}$  (et pas  $\rho$ ). La propriété à montrer par récurrence sur  $P$  doit permettre de faire varier l'environnement.

## 2–Donner du sens aux formules

- 1 Interprétations et vérité
- 2 Validité et satisfiabilité**
  - Définitions
  - Substitution et validité
- 3 Conséquence logique, équivalence
- 4 Modèles particuliers

- La vérité d'une formule se définit par rapport à une **interprétation** qui fixe le sens des symboles de fonction et de prédicats et un **environnement** qui détermine la valeur des variables libres.
- Lorsqu'on a juste une formule, elle peut être :
  - vraie dans toutes les interprétations et tous les environnements possibles,
  - vraie dans aucune interprétation et aucun environnement (toujours fausse)
  - ou bien être vraie dans certaines interprétations et certains environnements et fausse dans d'autres.
- Si la formule est close, sa vérité est la même quelque soit l'environnement et ne dépend que de l'interprétation des symboles.

## Definition (Validité, satisfiabilité, modèle)

Soit  $A$ , une formule du calcul des prédicats sur une signature  $(\mathcal{F}, \mathcal{R})$ .

- La formule est dite **valide** (on dit aussi que c'est une **tautologie**) si sa valeur de vérité est vraie pour toute interprétation de la signature et tout environnement.

On note  $\models P$  pour représenter le fait que  $P$  est une tautologie, c'est-à-dire que pour toute interprétation  $I$  et environnement  $\rho$ , on a  $I, \rho \models P$

- La formule est dite **satisfiable** si sa valeur de vérité est vraie pour au moins une interprétation de la signature et un environnement.  
Une interprétation et un environnement qui rendent vraie la formule forment un **modèle de la formule**.
- La formule est dite **insatisfiable** (on dit aussi **contradictoire**) si sa valeur de vérité est fausse pour toute interprétation de la signature et tout environnement.

# Définitions pour un ensemble de formules

On étend ces notions à un ensemble  $\mathcal{E}$  *fini ou infini* de formules :

## Definition (Validité, satisfiabilité, modèle)

- L'ensemble  $\mathcal{E}$  est **valide** si pour toute interprétation  $I$ , tout environnement  $\rho$  et toute formule  $P \in \mathcal{E}$  on a  $I, \rho \models P$  (toutes les formules sont vraies dans toutes les interprétations). On notera  $\models \mathcal{E}$  cette propriété.
- L'ensemble  $\mathcal{E}$  est **satisfiable** s'il existe une interprétation  $I$  et un environnement  $\rho$  tels que pour toute formule  $P \in \mathcal{E}$  on a  $I, \rho \models P$  (il existe une interprétation qui rend vraies toutes les formules de  $\mathcal{E}$ , appelée **modèle** de  $\mathcal{E}$ ).
- L'ensemble  $\mathcal{E}$  est **insatisfiable** si pour toute interprétation  $I$  et tout environnement  $\rho$ , il existe une formule  $P \in \mathcal{E}$  telle que  $I, \rho \not\models P$  (il n'existe pas d'interprétation qui rend vraies toutes les formules ou de manière équivalente, toute interprétation rend fausse au moins une formule de  $\mathcal{E}$ ).



*Dire si les ensembles de formules suivants sont valides, satisfiables, insatisfiables :*

- 1  $\{\neg(x \wedge y) \Rightarrow \neg y\}$
- 2  $\{x \vee y, y \Rightarrow x, \neg y\}$
- 3  $\{x \vee y, x \Rightarrow y, \neg y\}$
- 4  $\{(\exists x, P(x)) \vee q, (\forall x, \neg P(x)) \Rightarrow \neg q\}$
- 5  $\{(p \Rightarrow (q \Rightarrow p)), ((p \Rightarrow q) \Rightarrow p) \Rightarrow p\}$
- 6  $\{(\forall x, P(x)) \Rightarrow q, \forall x, (P(x) \wedge \neg q)\}$

## Proposition

- Une formule  $A$  avec une variable libre  $x$  est valide si et seulement si la formule  $\forall x, A$  est valide.
- Une formule  $A$  avec une variable libre  $x$  est satisfiable si et seulement si la formule  $\exists x, A$  est satisfiable.
- Une formule  $A$  avec une variable libre  $x$  est insatisfiable si et seulement si la formule  $\exists x, A$  est insatisfiable.

**Preuve:** Cela découle directement des définitions de validité, satisfiabilité et de la valeur des formules quantifiées.  $\square$

- La validité, satisfiabilité d'une formule  $P$  est celle de  $\{P\}$
- La validité, satisfiabilité d'un ensemble fini de formules  $\{P_1, \dots, P_n\}$  est celle de  $P_1 \wedge \dots \wedge P_n$
- Liens entre propriétés de  $P$ ,  $Q$  et  $\{P, Q\}$

$P$	$Q$	$\{P, Q\}$
valides		
		valide
satisfiables		
		satisfiable
insatisfiables		
		insatisfiable

## 2–Donner du sens aux formules

- 1 Interprétations et vérité
- 2 Validité et satisfiabilité**
  - Définitions
  - Substitution et validité**
- 3 Conséquence logique, équivalence
- 4 Modèles particuliers

# Substitution et validité

La validité et l'insatisfiabilité qui sont des propriétés de toutes les interprétations ne changent pas lorsque l'on effectue une substitution dans une formule ou un ensemble de formules.

## Proposition

Soit  $P$  une formule,  $x$  une variable et  $t$  un terme

- Si une formule  $P$  est valide (resp. insatisfiable) alors  $P[x \leftarrow t]$  est valide (resp. insatisfiable).
- Si un ensemble de formules  $\mathcal{E}$  est valide (resp. insatisfiable) alors  $\mathcal{E}[x \leftarrow t] \stackrel{\text{def}}{=} \{P[x \leftarrow t] \mid P \in \mathcal{E}\}$  est valide (resp. insatisfiable).

**Preuve:** Ce résultat est une conséquence du fait que  $I, \rho \models P[x \leftarrow t]$  ssi  $I, (\rho + \{x \mapsto \text{val}_I(\rho, t)\}) \models P$



# Remplacer une variable par un terme

Les réciproques sont fausses. Soit un prédicat unaire  $Q$  et une constante  $c$ .

- La formule  $Q(x) \Rightarrow Q(c)$  n'est pas valide.
  - interprétation dont le domaine contient deux éléments  $\{0, 1\}$ , on interprète la constante  $c$  par la valeur  $0$ . On choisit d'avoir  $Q_i(1)$  vrai et  $Q_i(0)$  faux.
  - environnement dans lequel  $x$  a la valeur  $1$
  - La formule est fausse dans cette interprétation et cet environnement, donc elle est non valide
- La formule  $Q(x) \Rightarrow Q(c)[x \leftarrow c]$  qui est égale à  $Q(c) \Rightarrow Q(c)$  est valide.
- La formule  $Q(x) \wedge \neg Q(c)$  est satisfiable (prendre la même interprétation que précédemment)
- La formule  $Q(x) \wedge \neg Q(c)[x \leftarrow c]$  qui est égale à  $Q(c) \wedge \neg Q(c)$  est insatisfiable.
- La satisfiabilité simple n'est pas préservée, au contraire si  $P[x \leftarrow t]$  est satisfiable alors  $P$  est satisfiable mais le contraire est faux (voir exemple précédent).

# Remplacer un prédicat par une formule

Deux manières différentes de lire la formule  $A \vee \neg A$

- le symbole  $A$  peut correspondre à une variable mathématique qui représente n'importe quelle formule de la logique,
- le symbole  $A$  peut être une variable propositionnelle (symbole de prédicat d'arité 0), auquel cas on a exactement une formule syntaxique qui est différente d'autres formules de même format comme  $\perp \vee \neg \perp$ .

dans le cas de formules valides, les deux points de vue coïncident

# Remplacer une variable propositionnelle par une formule

On définit le remplacement d'une variable propositionnelle  $X$  par une formule  $Q$  par des équations récursives sur la structure de la formule.

## Definition ( $P[X \leftarrow Q]$ )

Soit  $P$  et  $Q$  des formules et  $X$  une variable propositionnelle, le remplacement de  $X$  par  $Q$  dans  $P$  est une formule notée  $P[X \leftarrow Q]$  définie récursivement sur la structure de  $P$ .

- Si  $P$  est une formule atomique : si  $P$  est la variable propositionnelle  $X$  alors  $P[X \leftarrow Q] = Q$  sinon  $P[X \leftarrow Q] = P$
- Si  $P$  est de la forme  $\neg A$  alors  $P[X \leftarrow Q] = \neg(A[X \leftarrow Q])$
- Si  $P$  est de la forme  $A \circ B$  avec  $\circ$  l'une des connecteurs propositionnels alors  $P[X \leftarrow Q] = (A[X \leftarrow Q]) \circ (B[X \leftarrow Q])$
- Si  $P$  est de la forme  $\forall x, A$  (resp.  $\exists x, A$ ) avec la variable  $x$  non libre dans la formule  $Q$ , alors  $P[X \leftarrow Q] = \forall x, (A[X \leftarrow Q])$  (resp.  $P[X \leftarrow Q] = \exists x, (A[X \leftarrow Q])$ )



# Cas des variables propositionnelles

On montre que si on remplace la variable  $X$  par n'importe quelle formule  $Q$  dans une formule valide, alors la formule obtenue reste valide.

## Proposition

*Soit  $P$  et  $Q$  des formules et  $X$  une variable propositionnelle, si  $P$  est une formule valide (resp. insatisfiable) alors il en est de même de la formule  $P[X \leftarrow Q]$  obtenue en remplaçant  $X$  par  $Q$  dans  $P$ .*

**Preuve:** (cas de la validité) Le résultat découle du fait que dans une interprétation  $I$  et un environnement  $\rho$  quelconques, la valeur de  $P[X \leftarrow Q]$  est égale à la valeur de  $P$  dans un environnement dans lequel la variable  $X$  a pour interprétation la valeur de  $Q$ . Ce résultat s'obtient facilement par récurrence sur la structure de  $P$ .

Maintenant si une formule est valide, elle est vraie tout le temps donc en particulier si on fixe une valeur pour une de ses variables (ici  $X$ ).



# Remplacer un prédicat par une formule paramétrée

- On généralise au remplacement d'un symbole de prédicat par une formule.
- Le symbole de prédicat est utilisé associé à des termes, il va être remplacé par une formule paramétrée par des variables qui seront substituées par les arguments du symbole de prédicat.
- Soit une formule  $P$  construite sur une signature qui comporte un symbole de prédicat  $n$ -aire  $R$ .
- Soit une formule  $Q$  et  $n$  variables  $x_1, \dots, x_n$ . On remplace dans la formule  $P$  toutes les sous-formules atomiques de la forme  $R(t_1, \dots, t_n)$  par la formule  $Q[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n]$ .
- On note le résultat de cette opération  $P[R(x_1, \dots, x_n) \leftarrow Q]$ .

## Exemple

*Signature : constante  $c$ , symbole de prédicat unaire  $Q$  et égalité binaire.*

- Soit  $A \stackrel{\text{def}}{=} P(c) \Rightarrow \exists x, P(x)$
- $A[P(z) \leftarrow z = z]$  est défini comme la formule :  $(c = c) \Rightarrow \exists x, x = x$

# Remplacer un symbole de prédicat par une formule

## Definition ( $P[R(x_1, \dots, x_n) \leftarrow Q]$ )

Soient  $P, Q$  des formules,  $R$  un symbole de prédicat d'arité  $n$  et  $x_1, \dots, x_n, n$  variables d'objet.

Le remplacement de  $R$  par  $Q$  paramétré par  $x_1, \dots, x_n$  dans  $P$ , est une formule  $P[R(x_1, \dots, x_n) \leftarrow Q]$  définie récursivement sur la structure de  $P$ .

- Si  $P$  est atomique alors si  $P$  est de la forme  $R(t_1, \dots, t_n)$  on a  
 $P[R(x_1, \dots, x_n) \leftarrow Q] = Q[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n]$   
sinon  $P[R(x_1, \dots, x_n) \leftarrow Q] = P$
- Si  $P$  est  $\neg A$  alors  $P[R(x_1, \dots, x_n) \leftarrow Q] = \neg(A[R(x_1, \dots, x_n) \leftarrow Q])$
- Si  $P$  est  $A \circ B$  avec  $\circ$  l'un des connecteurs propositionnels alors  
 $P[R(x_1, \dots, x_n) \leftarrow Q] = (A[R(x_1, \dots, x_n) \leftarrow Q]) \circ (B[R(x_1, \dots, x_n) \leftarrow Q])$
- Si  $P$  est  $\forall x, A$  (resp.  $\exists x, A$ ) et  $x$  n'est pas libre dans  $Q$ , alors  
 $P[R(x_1, \dots, x_n) \leftarrow Q] = \forall x, (A[R(x_1, \dots, x_n) \leftarrow Q])$   
(resp.  $\exists x, (A[R(x_1, \dots, x_n) \leftarrow Q])$ )

# Préservation de la validité par remplacement

## Proposition

Soit une formule  $P$  et un symbole de prédicat  $n$ -aire  $R$ . Soit une formule  $Q$  et  $n$  variables  $x_1, \dots, x_n$ .

Si  $P$  est valide (resp. insatisfiable) alors il en est de même de la formule  $P[R(x_1, \dots, x_n) \leftarrow Q]$ .

**Preuve:** (validité)  $I$  interprétation des symboles de  $P[R(x_1, \dots, x_n) \leftarrow Q]$  et  $\rho$  un environnement. On veut montrer que  $I, \rho \models P[R(x_1, \dots, x_n) \leftarrow Q]$

Tous les symboles et les variables qui apparaissent dans  $P$  apparaissent aussi dans  $P[R(x_1, \dots, x_n) \leftarrow Q]$  à l'exception peut-être de  $R$ . On construit une interprétation  $I'$  qui est définie comme  $I$  sauf pour  $R$  :

on pose  $R_{I'}(d_1, \dots, d_n)$  est vrai ssi  $I, (\rho + \{x_1 \mapsto d_1, \dots, x_n \mapsto d_n\}) \models Q$ .

On a  $I', \rho \models P$  ssi  $I, \rho \models P[R(x_1, \dots, x_n) \leftarrow Q]$  (preuve par récurrence sur la structure de  $P$ ).

Si  $P$  est valide, on a  $I', \rho \models P$ . On en déduit que  $I, \rho \models P[R(x_1, \dots, x_n) \leftarrow Q]$  et comme cela vaut pour tout  $I, \rho$ , on en déduit  $\models P[R(x_1, \dots, x_n) \leftarrow Q]$ .  $\square$

- Notation  $I, \rho \models P$
- Définitions de la validité et satisfiabilité pour un ensemble de formules
- Modèle d'un ensemble de formules
- Stabilité de la validité par substitution
- Stabilité de la validité par remplacement d'un symbole de prédicat par une formule paramétrée



*That's all Folks!*

## 2—Donner du sens aux formules

- 1 Interprétations et vérité
- 2 Validité et satisfiabilité
- 3 Conséquence logique, équivalence
  - Propriétés de la conséquence logique
  - Equivalences remarquables
- 4 Modèles particuliers

- Notation  $I, \rho \models P$
- Définitions de la validité et satisfiabilité pour un ensemble de formules
- Modèle d'un ensemble de formules
- Stabilité de la validité par substitution
- Stabilité de la validité par remplacement d'un symbole de prédicat par une formule paramétrée



# La dernière fois : Notation $I, \rho \models A$

Une notation plus intuitive :

- On note  $I, \rho \models A$  lorsque la formule  $A$  est vraie dans l'interprétation  $I$  et l'environnement  $\rho$ , c'est-à-dire lorsque  $\text{val}_I(\rho, A) = V$ .

Répetons :  $I, \rho \models A \stackrel{\text{def}}{=} \text{val}_I(\rho, A) = V$

- On note  $I, \rho \not\models A$  dans le cas contraire lorsque la formule  $A$  est fausse dans l'interprétation  $I$  et l'environnement  $\rho$ , c'est-à-dire lorsque  $\text{val}_I(\rho, A) = F$ .
- Si la formule  $A$  est **close**, alors sa valeur dans une interprétation ne dépend pas de l'environnement et on écrira simplement  $I \models A$  pour indiquer que  $A$  est vraie dans l'interprétation  $I$  (et n'importe quel environnement).

# La dernière fois : Définitions pour un ensemble de formules

On étend ces notions à un ensemble  $\mathcal{E}$  *fini ou infini* de formules :

## Definition (Validité, satisfiabilité, modèle)

- L'ensemble  $\mathcal{E}$  est **valide** si pour toute interprétation  $I$ , tout environnement  $\rho$  et toute formule  $P \in \mathcal{E}$  on a  $I, \rho \models P$  (toutes les formules sont vraies dans toutes les interprétations). On notera  $\models \mathcal{E}$  cette propriété.
- L'ensemble  $\mathcal{E}$  est **satisfiable** s'il existe une interprétation  $I$  et un environnement  $\rho$  tels que pour toute formule  $P \in \mathcal{E}$  on a  $I, \rho \models P$  (il existe une interprétation qui rend vraies toutes les formules de  $\mathcal{E}$ , appelée **modèle** de  $\mathcal{E}$ ).
- L'ensemble  $\mathcal{E}$  est **insatisfiable** si pour toute interprétation  $I$  et tout environnement  $\rho$ , il existe une formule  $P \in \mathcal{E}$  telle que  $I, \rho \not\models P$  (il n'existe pas d'interprétation qui rend vraies toutes les formules ou de manière équivalente, toute interprétation rend fausse au moins une formule de  $\mathcal{E}$ ).

# La dernière fois : Stabilité de la validité

## Proposition

Soit  $P$  une formule,  $x$  une variable et  $t$  un terme. Si  $P$  est valide (resp. insatisfiable) alors  $P[x \leftarrow t]$  est valide (resp. insatisfiable).

## Proposition

Soit  $P$  et  $Q$  des formules et  $X$  une variable propositionnelle. Si  $P$  est valide (resp. insatisfiable) alors il en est de même de  $P[X \leftarrow Q]$ .

## Proposition

Soit une formule  $P$  et un symbole de prédicat  $n$ -aire  $R$ . Soit une formule  $Q$  et  $n$  variables  $x_1, \dots, x_n$ . Si  $P$  est valide (resp. insatisfiable) alors il en est de même de  $P[R(x_1, \dots, x_n) \leftarrow Q]$ .

- Les réciproques sont fausses.
- Mêmes propriétés pour *ensembles* de formules.

## 2–Donner du sens aux formules

- 1 Interprétations et vérité
- 2 Validité et satisfiabilité
- 3 Conséquence logique, équivalence
  - Propriétés de la conséquence logique
  - Equivalences remarquables
- 4 Modèles particuliers

# Conséquence logique, équivalence

## Definition ( $I, \rho \models \mathcal{E}$ )

Soient  $\mathcal{E}$  un ensemble de formules,  $I$  une interprétation et  $\rho$  un environnement, on note  $I, \rho \models \mathcal{E}$  le fait que  $I, \rho \models P$ , pour toute formule  $P \in \mathcal{E}$ .

## Definition (Conséquence logique $\mathcal{E} \models A$ , équivalence $A \equiv B$ )

Si  $\mathcal{E}$  est un ensemble de formules et  $A$  une formule, on dit que  $A$  est **conséquence logique** de  $\mathcal{E}$  et on note  $\mathcal{E} \models A$  si pour toute interprétation  $I$  et environnement  $\rho$  on a : si  $I, \rho \models \mathcal{E}$  alors  $I, \rho \models A$

On dit que  $A$  et  $B$  sont des **formules équivalentes** et on écrit  $A \equiv B$  si  $A \models B$  et  $B \models A$ .

C'est-à-dire que pour toute interprétation  $I$  et environnement  $\rho$ , on a  $I, \rho \models A$  si et seulement si  $I, \rho \models B$ .

## Proposition

Pour toutes formules  $A_1, \dots, A_n, P$  et ensemble de formules  $\mathcal{E}$  :

- ①  $A_1, \dots, A_n \models P$  ssi  $A_1 \wedge \dots \wedge A_n \Rightarrow P$  est valide
- ② si  $\mathcal{E}$  est insatisfiable alors pour toute formule  $P$ ,  $\mathcal{E} \models P$
- ③  $\mathcal{E}$  est insatisfiable ssi  $\mathcal{E} \models \perp$
- ④  $\mathcal{E} \models P$  ssi  $\mathcal{E} \cup \{\neg P\}$  est insatisfiable.
- ⑤  $\mathcal{E}, P \models Q$  ssi  $\mathcal{E} \models P \Rightarrow Q$

La relation de conséquence logique est stable par substitution et remplacement. On étend les notations  $P[x \leftarrow t]$  et  $P[R(x_1, \dots, x_n) \leftarrow Q]$  à un ensemble de formules  $\mathcal{E}$  en appliquant la transformation à chacune des formules de l'ensemble.

## Proposition

Soit  $P$  une formule,  $\mathcal{E}$  un ensemble de formules tel que  $\mathcal{E} \models P$ .

- si  $x$  est une variable d'objet et  $t$  un terme alors  $\mathcal{E}[x \leftarrow t] \models P[x \leftarrow t]$
- si  $R$  est un symbole de prédicat  $n$ -aire,  $x_1, \dots, x_n$   $n$  variables et  $Q$  une formule alors  $\mathcal{E}[R(x_1, \dots, x_n) \leftarrow Q] \models P[R(x_1, \dots, x_n) \leftarrow Q]$

On en déduit les mêmes propriétés pour l'équivalence :

Soit  $P_1, P_2$  deux formules telles que  $P_1 \equiv P_2$ . On a  $P_1[x \leftarrow t] \equiv P_2[x \leftarrow t]$  et  $P_1[R(x_1, \dots, x_n) \leftarrow Q] \equiv P_2[R(x_1, \dots, x_n) \leftarrow Q]$ .

# Conséquence logique et remplacement

$P$	$P[x \leftarrow Q]$
valide	
	valide
insatisfiable	
	insatisfiable
$A_1, \dots, A_n \models P$	
<i>vrai</i>	$A_1[x \leftarrow Q], \dots, A_n[x \leftarrow Q] \models P[x \leftarrow Q]$
	<i>vrai</i>



# Conséquence et opérateurs logiques

La propriété de conséquence logique se combine avec les connecteurs et les quantificateurs.

## Proposition

Soit  $\mathcal{E}$  un ensemble de formules,  $A_1, A_2, B_1$  et  $B_2$  des formules. On suppose que  $\mathcal{E}, A_1 \models B_1$  et  $\mathcal{E}, A_2 \models B_2$ , on a alors

- $\mathcal{E}, \neg B_1 \models \neg A_1$
- $\mathcal{E}, A_1 \wedge A_2 \models B_1 \wedge B_2$
- $\mathcal{E}, A_1 \vee A_2 \models B_1 \vee B_2$
- $\mathcal{E}, B_1 \Rightarrow A_2 \models A_1 \Rightarrow B_2$
- si de plus la variable  $x$  n'apparaît pas libre dans  $\mathcal{E}$ , alors  
 $\mathcal{E}, (\forall x, A_1) \models \forall x, B_1$  et  $\mathcal{E}, (\exists x, A_1) \models \exists x, B_1$

## 2–Donner du sens aux formules

- 1 Interprétations et vérité
- 2 Validité et satisfiabilité
- 3 Conséquence logique, équivalence**
  - Propriétés de la conséquence logique
  - **Equivalences remarquables**
- 4 Modèles particuliers

## Definition (equivalence)

$P$  et  $Q$  sont **équivalentes** (noté  $P \equiv Q$ ) ssi  $P \models Q$  et  $Q \models P$

- deux formules équivalentes sont *vraies en même temps* (pour les mêmes interprétations)
- c'est une *relation d'équivalence* : chaque formule appartient à exactement une classe d'équivalence
- les formules n'ont pas la même syntaxe mais représentent la même *vérité*

Les équivalences de la suite de ce chapitre sont des propriétés de base de la logique qui pourront être utilisées sans justification.

# Lois algébriques

L'ensemble des booléens avec les opérations de conjonction et de disjonction forme ce que l'on appelle une **Algèbre de Boole**.

- la conjonction et la disjonction sont associatifs et commutatifs :

$$\begin{aligned}P \wedge Q &\equiv Q \wedge P & P \vee Q &\equiv Q \vee P \\(P \wedge Q) \wedge R &\equiv P \wedge (Q \wedge R) & (P \vee Q) \vee R &\equiv P \vee (Q \vee R)\end{aligned}$$

- distributivité entre conjonction et disjonction

$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R) \quad P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$$

- $\top$  et  $\perp$  sont des éléments neutres ou absorbants :

$$P \wedge \top \equiv P \quad P \vee \top \equiv \top \quad P \vee \perp \equiv P \quad P \wedge \perp \equiv \perp$$

Certains connecteurs peuvent se définir en fonction d'autres :

$$\begin{aligned}\neg P &\equiv P \Rightarrow \perp & P \Rightarrow Q &\equiv \neg P \vee Q \\P \Leftrightarrow Q &\equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P) &&\equiv (P \wedge Q) \vee (\neg P \wedge \neg Q)\end{aligned}$$

# Exercice

*Donner des formules équivalentes pour  $\perp \Rightarrow P$ ,  $\top \Rightarrow P$ ,  $P \Rightarrow \top$*

# Exercice

Donner des formules équivalentes pour  $\perp \Rightarrow P$ ,  $\top \Rightarrow P$ ,  $P \Rightarrow \top$

1  $\perp \Rightarrow P \equiv \top$

2  $\top \Rightarrow P \equiv P$

3  $P \Rightarrow \top \equiv \top$

# Lois de de Morgan

Les lois de de Morgan établissent le comportement de la négation par rapport aux autres connecteurs :

$$\begin{array}{lll} \neg \perp & \equiv \top & \neg(P \wedge Q) \equiv \neg P \vee \neg Q \\ \neg \top & \equiv \perp & \neg(P \vee Q) \equiv \neg P \wedge \neg Q \\ \neg \neg P & \equiv P & \end{array}$$

$$\begin{array}{ll} \neg(P \Rightarrow Q) & \equiv P \wedge \neg Q \\ \neg \exists x, P(x) & \equiv \forall x, \neg P(x) \\ \neg \forall x, P(x) & \equiv \exists x, \neg P(x) \end{array}$$

# Forme normale de négation

En utilisant les lois de de Morgan et les formules équivalentes pour l'implication, on peut associer à toute formule, une formule équivalente qui ne contient pas le symbole  $\Rightarrow$  et dont les négations ne portent que sur les formules atomiques  $R(t_1, \dots, t_n)$ .

## Definition (Forme normale de négation)

Une formule est dite en forme normale de négation si elle ne contient que les connecteurs logiques  $\wedge, \vee$ , les quantificateurs  $\forall, \exists$  et que le symbole de négation n'apparaît que devant une formule atomique  $R(t_1, \dots, t_n)$ .

- Voir TD exo 5.2!



# Equivalence et quantificateurs

Symboles de prédicat :  $H$  et  $G$  unaires et  $R$  binaires.

Les équivalences suivantes sont vérifiées :

- permuter deux mêmes quantificateurs :

$$\forall x, \forall y, R(x, y) \equiv \forall y, \forall x, R(x, y) \quad \exists x, \exists y, R(x, y) \equiv \exists y, \exists x, R(x, y)$$

- réarranger quantification universelle et conjonction , quantification existentielle et disjonction :

$$\forall x, (G(x) \wedge H(x)) \equiv (\forall x, G(x)) \wedge (\forall x, H(x))$$

$$\exists x, (G(x) \vee H(x)) \equiv (\exists x, G(x)) \vee (\exists x, H(x))$$

- éliminer un quantificateur sur une variable non utilisée :

Si  $x \notin \text{vl}(A)$  alors  $\forall x, A \equiv \exists x, A \equiv A$ .

- “sortir” d’un quantificateur une sous-formule qui n’utilise pas la variable.

Si  $x \notin \text{vl}(A)$  alors :

$$\forall x, (A \wedge G(x)) \equiv A \wedge \forall x, G(x) \quad \forall x, (A \vee G(x)) \equiv A \vee \forall x, G(x)$$

$$\exists x, (A \wedge G(x)) \equiv A \wedge \exists x, G(x) \quad \exists x, (A \vee G(x)) \equiv A \vee \exists x, G(x)$$

# Exercice : forme prénexe

En utilisant les équivalences précédentes, montrer les équivalences suivantes :

- si  $x \notin \text{vl}(A)$  alors

$$\forall x, (A \Rightarrow H(x)) \equiv A \Rightarrow \forall x, H(x) \quad \exists x, (A \Rightarrow H(x)) \equiv A \Rightarrow \exists x, H(x)$$

- si  $x \notin \text{vl}(A)$  alors

$$\forall x, (G(x) \Rightarrow A) \equiv (\exists x, G(x)) \Rightarrow A \quad \exists x, (G(x) \Rightarrow A) \equiv (\forall x, G(x)) \Rightarrow A$$

Une application de la dernière équivalence est le principe du “buveur” : dans un bar quelconque, il existe une personne telle que si cette personne boit alors tout le monde boit.

Il suffit de prendre pour  $G(x)$  la formule  $x \text{ boit}$  et pour  $A$  la formule  $\forall x, x \text{ boit}$

# Exercice : contre-exemples

Trouver des interprétations qui justifient les résultats suivants :

- $\forall x, \exists y, R(x, y) \not\equiv \exists y, \forall x, R(x, y)$
- $\forall x, (G(x) \vee H(x)) \not\equiv (\forall x, G(x)) \vee (\forall x, H(x))$
- $\exists x, (G(x) \wedge H(x)) \not\equiv (\exists x, G(x)) \wedge (\exists x, H(x))$

- La définition de la relation de conséquence logique et ses propriétés de stabilité par substitution, remplacement et connecteurs
- Les équivalences remarquables en particulier les lois de de Morgan pour les quantificateurs et l'implication



*That's all Folks!*

## 2—Donner du sens aux formules

- 1 Interprétations et vérité
- 2 Validité et satisfiabilité
- 3 Conséquence logique, équivalence
- 4 Modèles particuliers
  - Modèle fini
  - Modèle de Herbrand

- La définition de la relation de conséquence logique et ses propriétés de stabilité par substitution, remplacement et connecteurs
- Les équivalences remarquables en particulier les lois de de Morgan pour les quantificateurs et l'implication

## 2—Donner du sens aux formules

1 Interprétations et vérité

2 Validité et satisfiabilité

3 Conséquence logique, équivalence

**4 Modèles particuliers**

- Modèle fini
- Modèle de Herbrand



Certaines propriétés comme la validité ou l'insatisfiabilité s'appuient sur le caractère vrai ou faux d'une formule dans *toutes les interprétations*, ce qui en général représente une infinité de situations.

Par contre la satisfiabilité demande juste d'*exhiber un seul modèle de la formule*, il suffit alors de bien choisir.

Nous allons dans ce chapitre étudier quelques classes d'interprétations qui peuvent être utiles.

On s'intéresse à un domaine fini particulier  $\mathcal{D} = \{d_1; \dots; d_n\}$ .

On supposera (quitte à ajouter des constantes) que l'on a dans le langage des termes sans variable  $\{c_1; \dots; c_n\}$  tels que  $\text{val}(c_i) = d_i$ .

Dans une telle interprétation, pour toute formule  $P$  et toute variable libre  $x$  dans  $P$ , les deux formules suivantes sont vraies.

$$(\forall x, P) \Leftrightarrow (P[x \leftarrow c_1] \wedge \dots \wedge P[x \leftarrow c_n])$$

$$(\exists x, P) \Leftrightarrow (P[x \leftarrow c_1] \vee \dots \vee P[x \leftarrow c_n])$$

Cette technique permet d'éliminer les quantificateurs d'une formule close et d'obtenir une formule propositionnelle sans variable.

On remplace chaque formule atomique  $R(t_1, \dots, t_p)$  par  $R(c_{i_1}, \dots, c_{i_p})$  si  $\text{val}(t_k) = d_{i_k}$ .

On introduit une variable propositionnelle pour chaque sous-formule atomique et un solveur propositionnel peut répondre à la question de la satisfiabilité.

# Exercice

On se place dans un langage qui contient  $n$  constantes  $\{c_1; \dots; c_n\}$  et un symbole de prédicat binaire pour l'égalité.

On s'intéresse uniquement aux interprétations dans lesquelles le symbole d'égalité est interprété comme l'égalité sur le domaine :  $I, \rho \models t = u$  ssi  $\text{val}_I(\rho, t) = \text{val}_I(\rho, u)$ .

- ① Que peut-on dire du cardinal du domaine d'une interprétation qui rend vraie la formule  $\forall x, x = c_1 \vee \dots \vee x = c_n$ ?
- ② Que peut-on dire du cardinal du domaine d'une interprétation qui rend vraie la formule  $c_1 \neq c_2 \wedge \dots \wedge c_1 \neq c_n \wedge c_2 \neq c_3 \wedge \dots \wedge c_{n-1} \neq c_n$ ?
- ③ Même question pour la formule  $\exists x_1 \dots x_n, x_1 \neq x_2 \wedge \dots \wedge x_1 \neq x_n \wedge x_2 \neq x_3 \wedge \dots \wedge x_{n-1} \neq x_n$ ?
- ④  $P$  symbole de prédicat unaire,  $\mathcal{E}$  les axiomes de la théorie de l'égalité, (en particulier  $\forall x y, x = y \Rightarrow (P(x) \Rightarrow P(y))$ ). Montrer
  - $\mathcal{E}, (\forall x, x = c_1 \vee \dots \vee x = c_n), (P(c_1) \wedge \dots \wedge P(c_n)) \models \forall x, P(x)$
  - $\mathcal{E}, (\forall x, x = c_1 \vee \dots \vee x = c_n), (\exists x, P(x)) \models P(c_1) \vee \dots \vee P(c_n)$

## 2—Donner du sens aux formules

- 1 Interprétations et vérité
- 2 Validité et satisfiabilité
- 3 Conséquence logique, équivalence
- 4 Modèles particuliers**
  - Modèle fini
  - **Modèle de Herbrand**

- Pour établir qu'une formule est valide, il faut a priori raisonner sur tous les modèles.
  - $\forall x y, x < y \Rightarrow x + 1 \leq y$
  - $\forall x y, x < y \Rightarrow \exists z, x < z < y$
- Le logicien Jacques Herbrand (1908-1931) a montré qu'on peut se limiter à regarder des modèles sur *un seul* domaine, on parle de **modèle de Herbrand** qui est un *modèle syntaxique*.
- L'idée de base est d'interpréter les termes par eux-mêmes.

- Un langage du premier ordre est défini par une signature formée d'un ensemble  $\mathcal{F}$  de symboles de fonctions (dont les constantes) et d'un ensemble  $\mathcal{R}$  de symboles de prédicats.
- les termes clos sont ceux formés sur la signature  $\mathcal{F}$  qui ne contiennent pas de variable, on note  $\mathcal{T}(\mathcal{F})$  l'ensemble des termes clos.
- S'il n'y a pas de constante dans l'ensemble  $\mathcal{F}$  alors l'ensemble des termes clos est vide.
- Comme un domaine doit être un ensemble non vide, dans toute la suite, on supposera que la signature  $\mathcal{F}$  contient au moins une constante et on l'ajoutera si nécessaire à la signature.

## Definition (Domaine de Herbrand)

Le **domaine de Herbrand** d'un langage logique de signature  $(\mathcal{F}, \mathcal{R})$  est l'ensemble  $\mathcal{T}(\mathcal{F})$  des termes clos formés à partir des symboles de  $\mathcal{F}$ .

# Exemple

Si la signature est formée d'une constante  $a$  et d'un symbole de fonction unaire  $f$ , le domaine de Herbrand contient les termes  $a, f(a), f(f(a)), \dots, f^n(a), \dots$



Le domaine de Herbrand est formé de termes clos, représentés comme des arbres.

Deux termes syntaxiquement différents correspondent à deux éléments différents.

Pour l'arithmétique avec la constante  $0$ , la fonction successeur unaire  $S$ , la fonction binaire  $+$ , alors les termes clos  $0+S(0)$ ,  $S(0)+0$  et  $S(0)$  sont tous différents. L'interprétation de l'égalité dans les entiers ne correspondra pas à l'égalité du domaine mais à une relation d'équivalence qui identifiera les trois expressions précédentes.

# Interprétation de Herbrand : fonctions

On interprète chaque symbole de fonction dans le domaine de Herbrand par “lui-même”.

## Definition (Interprétation de Herbrand (fonctions))

Une **interprétation de Herbrand** est une interprétation  $H$  construite sur le domaine  $\mathcal{T}(\mathcal{F})$  dans laquelle chaque symbole de fonction  $f$  d'arité  $n$  est interprété par une fonction  $f_H$  définie par :

$$f_H(t_1, \dots, t_n) \stackrel{\text{def}}{=} f(t_1, \dots, t_n) \quad t_i \in \mathcal{T}(\mathcal{F})$$

## Proposition

*Dans n'importe quelle interprétation de Herbrand  $H$ , la valeur d'un terme clos est lui-même : si  $t \in \mathcal{T}(\mathcal{F})$  alors  $\text{val}(H, t) = t$*

**Preuve:** La preuve se fait par simple récurrence sur la structure des termes.  $\square$



# Base de Herbrand

Interpréter le symbole de prédicat  $R$  sur le domaine de Herbrand : ensemble des  $n$ -uplets de termes clos  $(t_1, \dots, t_n)$  tels que  $R(t_1, \dots, t_n)$  est vrai.

## Definition (Base de Herbrand)

La **base de Herbrand** est l'ensemble des formules atomiques closes construites sur le langage.

## Exemple

*Dans un langage avec une constante  $a$ , un symbole de fonction  $f$ , un symbole de prédicat unaire  $P$  et un symbole de prédicat binaire  $R$ , la base de Herbrand est composé des formules atomiques*

*$P(a), P(f(a)) \dots P(f^n(a)), \dots, R(a, a), R(f(a), a), \dots, R(f^n(a), f^p(a)), \dots$*

Il y a en général une infinité de formules atomiques closes. Comme il n'y a qu'un nombre fini de symboles dans une formule, la base de Herbrand associée aux symboles d'une formule sera toujours finie ou dénombrable.

Un modèle de Herbrand fixe l'interprétation des termes mais pas celle des prédicats. Il y a autant de modèles de Herbrand que de manière d'assigner des valeurs de vérité aux formules atomiques closes.

## Definition (Modèle de Herbrand)

Une **modèle de Herbrand** est défini en assignant des valeurs de vérité à chacune des formules de la base de Herbrand (c'est-à-dire à chaque formule atomique close).

# Exemples

Donner les domaines et bases de Herbrand pour les signatures des formules suivantes avec  $a$  et  $b$  des constantes et  $f$  un symbole de fonction :

①  $\forall x y, R(x, a) \wedge \neg R(b, y)$

②  $\forall x y, R(x, a) \Rightarrow R(y, f(y))$

# Exemples

Donner les domaines et bases de Herbrand pour les signatures des formules suivantes avec  $a$  et  $b$  des constantes et  $f$  un symbole de fonction :

①  $\forall x y, R(x, a) \wedge \neg R(b, y)$

②  $\forall x y, R(x, a) \Rightarrow R(y, f(y))$

- ①
- domaine de Herbrand :  $D_H = \{a, b\}$
  - base de Herbrand :  $B_H = \{R(a, a), R(a, b), R(b, a), R(b, b)\}$

Donner les domaines et bases de Herbrand pour les signatures des formules suivantes avec  $a$  et  $b$  des constantes et  $f$  un symbole de fonction :

①  $\forall x y, R(x, a) \wedge \neg R(b, y)$

②  $\forall x y, R(x, a) \Rightarrow R(y, f(y))$

- ①
  - domaine de Herbrand :  $D_H = \{a, b\}$
  - base de Herbrand :  $B_H = \{R(a, a), R(a, b), R(b, a), R(b, b)\}$
- ②
  - domaine de Herbrand :  $D_H = \{a, f(a), f(f(a)), \dots\}$
  - base de Herbrand :  $B_H = \{R(a, a), R(a, f(a)), R(f(a), a), R(f(a), f(a)), \dots\}$

## Proposition

Une formule *close sans quantificateur* est satisfiable si et seulement si il existe une interprétation de Herbrand qui rend vraie la formule.

### Preuve:

- $P$  : formule *close sans quantificateur*
- si une interprétation de Herbrand rend la formule vraie alors  $P$  est satisfiable (par définition de la satisfiabilité)
- soit un modèle  $I$  de la formule ( $I \models P$ ), alors on prend comme interprétation de Herbrand  $\{R(t_1, \dots, t_n) \mid I \models R(t_1, \dots, t_n)\}$ .
- toutes les sous-formules atomiques de  $P$  sont dans la base de Herbrand
- la formule se comporte donc comme une formule propositionnelle la valeur de la formule ne dépend que de la valeur des sous-formules atomiques

$$val_H(P) = val_I(P) = V$$

## Definition (Formule universelle)

Une formule logique est dite **universelle** si elle est de la forme

$$\forall x_1 \dots x_n, A$$

avec **A** une formule sans quantificateur.

## Proposition

*Une formule **universelle close** est satisfiable si et seulement si il existe une interprétation de Herbrand qui rend vraie la formule.*

## Proposition

Une formule *universelle close* est satisfiable si et seulement si il existe une interprétation de Herbrand qui rend vraie la formule.

S'il existe une interprétation de Herbrand qui rend vraie la formule alors celle-ci est satisfiable.

Réciproquement, on suppose  $I \models \forall x_1 \dots x_n, A$ .

On construit le modèle de Herbrand  $H$  tel que  $H \models R(t_1, \dots, t_n)$  ssi

$I \models R(t_1, \dots, t_n)$ .

On doit montrer  $H \models \forall x_1 \dots x_n, A$ .

Soit  $t_1, \dots, t_n \in \mathcal{T}(\mathcal{F})$ ,  $\iota \stackrel{\text{def}}{=} \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ , montrons  $H, \iota \models A$ .

Or  $\text{val}_H(\iota, A) = \text{val}_H(A[x_1 \leftarrow t_1; \dots, x_n \leftarrow t_n])$  en utilisant le lien entre substitution et environnement et le fait que dans un modèle de Herbrand la valeur d'un terme clos est lui-même ( $\text{val}_H(t) = t$ ).

$H$  et  $I$  ont même valeur sur les formules atomiques,  $A[x_1 \leftarrow t_1; \dots, x_n \leftarrow t_n]$  est une formule propositionnelle close, et  $I \models \forall x_1 \dots x_n, A$ , on a donc :

$$\begin{aligned} \text{val}_H(A[x_1 \leftarrow t_1; \dots, x_n \leftarrow t_n]) &= \text{val}_I(A[x_1 \leftarrow t_1; \dots, x_n \leftarrow t_n]) \\ &= \text{val}_I(\{x_1 \mapsto \text{val}_I(t_1), \dots, x_n \mapsto \text{val}_I(t_n)\}, A) \\ &= V \end{aligned}$$



## Definition (Instance close d'une formule)

Soit une formule  $A$  dont les variables libres sont  $\{x_1, \dots, x_n\}$ .

Une formule de la forme  $A[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n]$  avec  $t_1, \dots, t_n \in \mathcal{T}(\mathcal{F})$  est appelée **instance close** de  $A$ .

## Proposition (Théorème de Herbrand)

Soit  $B \stackrel{\text{def}}{=} \forall x_1 \dots x_n, A$  une **formule universelle close** du calcul des prédicats avec  $A$  sans quantificateur.

- la formule  $B$  est satisfiable ssi tout sous-ensemble fini  $\{A_1, \dots, A_p\}$  d'instances closes de  $A$  est satisfiable ;
- de manière équivalente : la formule  $B$  est insatisfiable ssi il existe un ensemble fini  $\{A_1, \dots, A_p\}$  d'instances closes de  $A$  qui est insatisfiable.

**Preuve:** On montre d'abord que la satisfiabilité de la formule  $\forall x_1 \dots x_n, A$  se ramène à la satisfiabilité d'un ensemble dénombrable de formules closes  $\{A[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n] \mid t_1, \dots, t_n \in \mathcal{T}(\mathcal{F})\}$ .

On utilise les mêmes arguments que précédemment,  $\forall x_1 \dots x_n, A$  est vraie dans une interprétation de Herbrand si pour tous les termes  $t_1, \dots, t_n \in \mathcal{T}(\mathcal{F})$ ,  $A[x_1 \leftarrow t_1; \dots, x_n \leftarrow t_n]$  est vraie dans ce modèle.

Et donc en rassemblant les éléments précédents, on a que  $\forall x_1 \dots x_n, A$  est satisfiable si et seulement si l'ensemble (fini ou dénombrable) de toutes les formules  $A[x_1 \leftarrow t_1; \dots, x_n \leftarrow t_n]$  pour  $t_1, \dots, t_n \in \mathcal{T}(\mathcal{F})$  est satisfiable.

On utilise ensuite le théorème de compacité qui dit que tout ensemble de formules insatisfiable possède un sous-ensemble fini insatisfiable pour conclure. □

# Exemple

Pour montrer la satisfiabilité d'une formule close dans le calcul des prédicats, le domaine de Herbrand qui nous intéresse est celui formé de la signature correspondant uniquement aux *symboles qui apparaissent dans la formule A*, auquel on ajoute éventuellement une constante.

## Exemple

Soit la formule  $\neg((\forall x, P(x)) \Rightarrow \exists x, P(x))$  elle est équivalente à  $(\forall x, P(x)) \wedge \forall x, \neg P(x)$  et donc à  $\forall x y, P(x) \wedge \neg P(y)$ .

Le domaine de Herbrand est composé d'une seule constante *a* et la base de Herbrand a une seule formule atomique close  $P(a)$ .

Il y a donc deux modèles de Herbrand. Celui dans lequel  $P(a)$  est vrai et celui dans lequel  $P(a)$  est faux.

Dans les deux modèles, la formule  $P(a) \wedge \neg P(a)$  qui est la seule instance close de  $P(x) \wedge \neg P(y)$  est fausse. On peut en déduire que la formule  $\forall x y, P(x) \wedge \neg P(y)$  est insatisfiable et donc que  $(\forall x, P(x)) \Rightarrow \exists x, P(x)$  est valide.

Soit la formule  $\forall x y, (P(x) \wedge Q(y) \wedge (\neg P(a) \vee \neg Q(a)))$ .

- Quel est le domaine de Herbrand et la base de Herbrand associés ?
- Donner toutes les interprétations de Herbrand possibles.
- La formule est-elle satisfiable ?

- La formule  $A$  définie comme  $(\exists x, P(x)) \Rightarrow \forall x, P(x)$  est-elle valide ?
- Construire la formule  $B$  qui est la négation de  $A$ .
- Donner la base de Herbrand associée à la formule  $B$ . Combien y-a-t-il de modèles de Herbrand ?
- Est-ce qu'il y a un modèle de Herbrand qui rend vraie la formule  $B$  ?
- Peut-on en déduire que  $B$  est insatisfiable ?

- Le théorème de Herbrand nous dit que si une formule  $\forall x, A$  (avec  $A$  sans quantificateur) est tout le temps fausse alors on peut trouver des termes  $t_1, \dots, t_n$  tels que la conjonction finie  $A[x \leftarrow t_1] \wedge \dots \wedge A[x \leftarrow t_n]$  est déjà fausse dans toute interprétation.
  - Pour chaque interprétation  $I$  il y a un terme  $t$  tel que  $I \not\models A[x \leftarrow t]$
  - Il n'existe pas en général de terme  $t$  tel que  $A[x \leftarrow t]$  est fausse dans toutes les interprétations
  - Par contre on peut trouver un ensemble fini de termes dont la conjonction ne sera jamais vraie.
- En utilisant la négation, on peut obtenir un théorème analogue pour la validité des formules existentielles
  - Si une formule existentielle  $\exists x, A$  est valide alors il existe un ensemble fini de termes  $t_1, \dots, t_n$  tels que la disjonction finie  $A[x \leftarrow t_1] \vee \dots \vee A[x \leftarrow t_n]$  est valide

Le théorème de Herbrand ramène un problème **sémantique** général (existence d'un modèle) du premier ordre (avec quantificateurs) à un problème propositionnel que l'on va pouvoir traiter avec des outils de nature **syntactique**.

Même si le calcul propositionnel est décidable, cela ne donne pas une méthode de décision pour le calcul des prédicats.

En effet l'ensemble des formules dont on doit prouver le caractère insatisfiable est infini en général.

- Si l'ensemble est insatisfiable, il y aura un sous-ensemble fini insatisfiable, et on pourra le trouver en énumérant tous les ensembles.
- Par contre si l'ensemble est satisfiable, alors n'importe quel sous-ensemble fini est satisfiable, et donc on ne trouvera pas de sous-ensemble insatisfiable mais on ne pourra pas conclure par une simple énumération.

- Ecrire des formules logiques permettant de contraindre le nombre d'éléments dans les interprétations (au moins/au plus ou exactement  $n$ )
- Définir le domaine de Herbrand associé à une formule et savoir caractériser les interprétations de Herbrand
- Chercher une interprétation de Herbrand qui rend vraie une formule
- Appliquer le théorème de Herbrand pour justifier l'insatisfiabilité d'une formule universelle



Dans le chapitre suivant, nous introduisons des méthodes de transformation de formules qui permettront de se ramener au cas des formules universelles. Nous étudierons ensuite différentes techniques pour prouver la validité ou la satisfiabilité des formules en raisonnant uniquement sur leur forme syntaxique.

# 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

# 3—Manipuler les formules

## 1 Formes normales

- aborder la logique et les formules sur le plan *syntactique*
- établir des propriétés logiques des formules (validité, satisfiabilité) par des transformations sans passer par la notion de modèle.
  - 1 mises en formes canoniques de formules logiques (skolémisation, **mise en forme clause**).
  - 2 représentation alternatives des formules propositionnelles (diagrammes de décisions binaires)
  - 3 systèmes de déduction pour les formules (calcul des séquents).

# 3—Manipuler les formules

## 1 Formes normales

- Forme normale de négation
- Formes normales conjonctive et disjonctive

- La même “vérité” peut s’exprimer de plusieurs manières différentes
- Intérêt informatique d’utiliser des formes “simplifiées” : moins de connecteurs à traiter
  - $\Rightarrow$  est superflu ( $A \Rightarrow B \equiv \neg A \vee B$ )
  - on peut aussi tout faire avec  $\Rightarrow$  et  $\perp$
  - ... ou avec le connecteur de *Sheffer*
- Une **forme normale** est une manière *standardisée* de représenter une formule
  - la forme normale est *syntactiquement* différente mais logiquement *équivalente*
  - facilite le test d’équivalence, la recherche de modèle. ...
  - analogie avec les expressions polynomiales (forme développée en monomes, forme factorisée, forme de Horner. ...)
    - $$\begin{aligned}x + 3 - 2 + x^2 + x &= x^2 + 2x + 1 \\ &= (x + 1)^2 \\ &= (x + 2)x + 1\end{aligned}$$

## Definition (Littéral)

On appelle **littéral** une formule qui est soit une formule atomique, soit de la forme  $\neg R(t_1, \dots, t_n)$  avec  $R$  un symbole de prédicat.

- Une **instance de prédicat** désigne une formule  $R(t_1, \dots, t_n)$  avec  $R$  un symbole de prédicat.
- Une **formule atomique** est soit une instance de prédicat, soit  $\top$  ou  $\perp$ .
- Dans le cas propositionnel, les instances de prédicat sont juste les **variables propositionnelles**.

# Forme normale de négation

Une formule en **forme normale de négation**

- ne contient pas de connecteur  $\Rightarrow$  (ou  $\Leftrightarrow$ )
- les seules négations portent sur des symboles de prédicat.
- utilisation des lois de de Morgan pour propager les négations
- ne change pas (l'ordre de grandeur de) la taille de la formule.

## Proposition

*Toute formule est équivalente à une formule en **forme normale de négation**, c'est-à-dire qui ne comporte pas de connecteur  $\neg$  sauf dans un **littéral** et n'utilise que les connecteurs  $\top$ ,  $\perp$ ,  $\vee$  et  $\wedge$  et les quantificateurs  $\forall$  et  $\exists$ .*

Pour définir un algorithme qui calcule la forme normale de négation d'une formule, on construit deux fonctions de manière récursive :

- $fnn$  :  $fnn(P)$  formule en forme normale de négation équivalente à  $P$
- $neg$  :  $neg(P)$  formule en forme normale de négation équivalente à  $\neg P$ .



# Forme normale de négation : algorithme

$\text{fnn}(p)$	=	si $p$ atomique
$\text{fnn}(\neg A)$	=	$\text{neg}(A)$
$\text{fnn}(A \circ B)$	=	$\circ \in \{\vee, \wedge\}$
$\text{fnn}(A \Rightarrow B)$	=	
$\text{fnn}(\forall x, A)$	=	
$\text{fnn}(\exists x, A)$	=	
$\text{neg}(\top)$	=	
$\text{neg}(\perp)$	=	
$\text{neg}(R(t_1, \dots, t_n))$	=	$R$ symbole de prédicat
$\text{neg}(\neg A)$	=	
$\text{neg}(A \wedge B)$	=	
$\text{neg}(A \vee B)$	=	
$\text{neg}(A \Rightarrow B)$	=	
$\text{neg}(\forall x, A)$	=	
$\text{neg}(\exists x, A)$	=	

# Forme normale de négation : algorithme

$\text{fnn}(p) = p$  si  $p$  atomique

$\text{fnn}(\neg A) = \text{neg}(A)$

$\text{fnn}(A \circ B) = \text{fnn}(A) \circ \text{fnn}(B)$   $\circ \in \{\vee, \wedge\}$

$\text{fnn}(A \Rightarrow B) = \text{neg}(A) \vee \text{fnn}(B)$

$\text{fnn}(\forall x, A) = \forall x, \text{fnn}(A)$

$\text{fnn}(\exists x, A) = \exists x, \text{fnn}(A)$

$\text{neg}(\top) = \perp$

$\text{neg}(\perp) = \top$

$\text{neg}(R(t_1, \dots, t_n)) = \neg R(t_1, \dots, t_n)$   $R$  symbole de prédicat

$\text{neg}(\neg A) = \text{fnn}(A)$

$\text{neg}(A \wedge B) = \text{neg}(A) \vee \text{neg}(B)$

$\text{neg}(A \vee B) = \text{neg}(A) \wedge \text{neg}(B)$

$\text{neg}(A \Rightarrow B) = \text{fnn}(A) \wedge \text{neg}(B)$

$\text{neg}(\forall x, A) = \exists x, \text{neg}(A)$

$\text{neg}(\exists x, A) = \forall x, \text{neg}(A)$

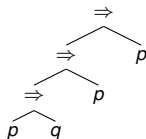
## Preuve:

- On montre par récurrence structurelle sur la formule  $P$  que  $fnn(P)$  et  $neg(P)$  sont en forme normale de négation et que de plus  $fnn(P) \equiv P$  et  $neg(P) \equiv \neg P$ .
- Taille de la formule transformée
  - membre droit d'une équation : même nombre de connecteurs logiques  $\wedge, \vee, \forall, \exists$  que dans le membre gauche.
  - ajout d'un symbole de négation uniquement dans le cas de la fonction  $neg$  appliquée à une instance de prédicat.
    - formule  $a_0 \Rightarrow a_1 \dots \Rightarrow a_n$  ( $n$  connecteurs logiques) : la forme normale de négation  $\neg a_0 \vee \neg a_1 \dots \vee \neg a_n$  en a  $2n$ .
    - c'est le pire cas : si  $P$  a  $n$  connecteurs, le nombre de connecteurs de  $fnn(P)$  est au plus  $2n$  et le nombre de connecteurs de  $neg(P)$  est au plus  $2n + 1$
    - un littéral est souvent traité "directement", sans plus de complexité qu'une formule atomique.



# Exemple

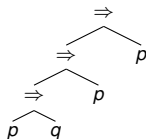
- $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$



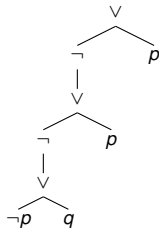
- suppression de  $\Rightarrow$  :

# Exemple

- $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$



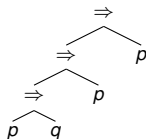
- suppression de  $\Rightarrow$  :



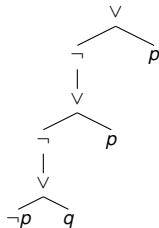
- forme normale de négation :

# Exemple

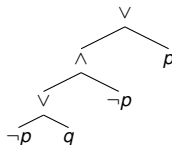
- $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$



- suppression de  $\Rightarrow$  :



- forme normale de négation :



# 3—Manipuler les formules

- 1 Formes normales
  - Forme normale de négation
  - Formes normales conjonctive et disjonctive

## Definition (Clause)

On appelle **clause** une formule qui n'utilise que des littéraux et la disjonction.



Un littéral tout seul est une clause dans laquelle il n'y a pas de disjonction.  
 $\top$  et  $\perp$  sont des clauses.

Etre une clause est une propriété **syntactique** des formules. Une formule peut être équivalente à une clause sans être une clause elle-même.

## Exemple

Soi  $p, q$  des variables propositionnelles.

- $\perp, p, \top, \neg p, p \vee q, p \vee q \vee \neg p$  sont des clauses.
- $p \wedge q, p \wedge \top, p \Rightarrow q$  ne sont pas des clauses.



# Clause sous forme simplifiée

- La disjonction est associative et commutative : on peut réarranger les littéraux librement à l'intérieur de la formule sans changer le sens et en restant syntaxiquement une clause.
- On a également  $P \vee P \equiv P$ ,  $P \vee \neg P \equiv \top$ ,  $P \vee \top \equiv \top$ ,  $P \vee \perp \equiv P$ .
- Une clause peut donc soit être de la forme  $\perp$  ou  $\top$ , soit s'écrire  $l_1 \vee l_2 \vee \dots \vee l_n$  avec  $l_i$  un littéral autre que  $\perp$  ou  $\top$ .
- représentation d'une clause : un ensemble de littéraux tous distincts
  - La clause  $\perp$  est appelée **clause vide** et correspond par convention à un ensemble vide de littéraux.
  - Si une clause contient une instance de prédicat et sa négation alors elle est de la forme  $p \vee \neg p \vee Q$  et se simplifie en  $\top$ .
- On peut toujours réécrire une clause de manière à ce que chaque instance de prédicat  $R(t_1, \dots, t_n)$  apparaisse au plus une fois sous forme positive ou négative ( $\neg R(t_1, \dots, t_n)$ ).

## Proposition

*Une clause est équivalente à  $\top$  ou  $\perp$  ou bien à une disjonction d'instances de prédicats (sous forme positive ou négative) dans laquelle chaque instance apparaît au plus une fois.*

**Preuve:** *Si une instance de prédicat  $p$  apparaît deux fois sous forme positive ou deux fois sous forme négative, alors en utilisant  $p \vee p \equiv p$  on peut retirer une des occurrences sans changer la valeur de la formule.*

*Si une instance de prédicat  $p$  apparaît une fois sous forme positive et une fois sous forme négative alors, en utilisant  $p \vee \neg p \equiv \top$ , on peut remplacer ces deux occurrences par la formule  $\top$ .*

*Une clause qui contient  $\top$  est simplement équivalente à  $\top$  (on dira que la clause est **triviale**).*

*Si une clause est de la forme  $P \vee \perp$  on peut la simplifier en la clause  $P$ .* □

- On peut représenter une clause non triviale par un ensemble d'instances de prédicat associées à des booléens
  - $R(t_1, \dots, t_n), \text{true}$  représente le littéral “positif”  $R(t_1, \dots, t_n)$
  - $R(t_1, \dots, t_n), \text{false}$  représente le littéral “négatif”  $\neg R(t_1, \dots, t_n)$
- **Exemple** : clause  $p \vee \neg q$  représentée par  $\{(p, \text{true}); (q, \text{false})\}$ .
- **Convention** :  $\perp$  est la clause associée à un ensemble vide de littéraux.
- Si  $L = \{l_1, \dots, l_n\}$  est un ensemble fini de littéraux, on notera  $\tilde{L}$  la formule logique correspondant à la clause  $l_1 \vee \dots \vee l_n$ .

# Exercice

Donner les équations récursives d'une fonction `clauseset` qui étant donné un ensemble d'instances de prédicats, chacune associée à un booléen représentant sa polarité, construit la formule correspondante sous forme de clause.

# Exercice

Donner les équations récursives d'une fonction `clauseset` qui étant donné un ensemble d'instances de prédicats, chacune associée à un booléen représentant sa polarité, construit la formule correspondante sous forme de clause.

```
clauseset(s) =  si s =  $\emptyset$  alors  $\perp$   
                sinon soit (p, b)  $\in$  s, s' = s \ (p, b)  
                  dans soit A = si b alors p sinon  $\neg p$   
                    dans si s' =  $\emptyset$  alors A  
                      sinon A  $\vee$  clauseset(s')
```

- Combien y a-t-il de clauses simplifiées non triviales et non équivalentes sur une signature comportant  $n$  variables propositionnelles ?
- Peut-on trouver une clause équivalente à n'importe quelle formule propositionnelle ?

- Dans une clause simplifiée non triviale, chaque variable propositionnelle peut ou non apparaître et si elle apparaît, cela peut être de manière positive ou négative.
- Deux clauses qui ne correspondent pas au même ensemble de littéraux ne sont pas équivalentes
  - il existe un littéral  $\ell$  qui est dans une des clauses  $C_1$  et pas dans l'autre  $C_2$
  - il existe une interprétation qui rend faux tous les littéraux de  $C_2$  et qui rend vrai  $\ell$  donc qui rend vraie la clause  $C_1$ .
- le nombre de clauses non équivalentes est :  $3^n + 1$
- le nombre de formules non équivalentes est :  $2^{2^n} > 3^n + 1$  pour  $n \geq 2$

# Clauses et vérité

Soit  $L$  un ensemble de littéraux,  $J$  une interprétation et  $\rho$  un environnement.

- $J, \rho \models \tilde{L}$  ssi il existe  $I \in L$  telle que  $J, \rho \models I$
- $J, \rho \not\models \tilde{L}$  ssi pour tout  $I \in L$ , on a  $J, \rho \not\models I$

## Proposition

Soient  $L_1$  et  $L_2$  des ensembles de littéraux clos et simplifiés,  
 $L_1 \subseteq L_2$  ssi  $\tilde{L}_1 \models \tilde{L}_2$ .

### Preuve:

- si  $L_1 \subseteq L_2$  alors soit  $L_3 = L_2 \setminus L_1$ . On a  $L_2 = L_1 \cup L_3$  et  $\tilde{L}_2 \equiv \tilde{L}_1 \vee \tilde{L}_3$ . Or  $A \models A \vee B$ , d'où  $\tilde{L}_1 \models \tilde{L}_2$ .
- si  $\tilde{L}_1 \models \tilde{L}_2$  et  $L_1 \not\subseteq L_2$ , alors il existe un littéral  $I \in L_1$  tel que  $I \notin L_2$ . Comme  $L_2$  porte sur des instances de prédicat closes différentes, on peut choisir une interprétation  $J$  telle que  $J \not\models \tilde{L}_2$ . L'instance de prédicat du littéral  $I \in L_1$  peut apparaître dans  $L_2$  mais avec un signe opposé (équivalent à  $\neg I$ ). On a que  $J \not\models \neg I$  donc  $J \models I$ . Si l'instance de prédicat sous-jacente à  $I$  n'apparaît pas dans  $L_2$  alors on peut étendre  $J$  pour que le littéral  $I$  soit vrai. Comme  $I \in L_1$ , on a que  $J \models \tilde{L}_1$  et donc une contradiction avec le fait que  $\tilde{L}_1 \models \tilde{L}_2$ .



# Formes normales conjonctives et disjonctives

Dans la suite, on ne s'intéresse qu'à la partie propositionnelle du langage c'est-à-dire à des formules qui ne contiennent pas de quantificateur.

## Definition (Forme normale conjonctive, forme clausale)

Une formule est dite en **forme normale conjonctive** (abrégé en **FNC**, CNF en anglais) si elle s'écrit comme une conjonction de clauses.

On peut représenter une formule en forme normale conjonctive par un ensemble de clauses, on parle alors de **forme clausale**.

Par convention, la *conjonction d'un ensemble vide de formules* est définie comme la formule vraie  $\top$ .

# Reconnaître une forme normale conjonctive

Forme normale conjonctive = conjonction de disjonctions de littéraux.



attention aux *cas limites* :

- Conjonction d'un ensemble vide de formules : toujours vrai  $\top$
- Disjonction d'un ensemble vide de formules : toujours faux  $\perp$
- Conjonction ou disjonction d'un ensemble réduit à une formule  $\{A\}$  : la formule elle-même  $A$
- Une formule en **forme normale conjonctive** peut ne comporter aucun symbole  $\vee$  ni symbole  $\wedge$ .
- Une disjonction de littéraux est en forme normale conjonctive, de même pour une conjonction de littéraux.
- une formule est en **forme normale conjonctive** ssi
  - elle ne comporte que les connecteurs logiques  $\vee$ ,  $\wedge$  et des littéraux
  - dans la représentation sous forme d'arbre, il n'y a pas de connecteur  $\wedge$  *en dessous* d'un connecteur  $\vee$ .

## Exemple (Forme normale conjonctive)

- Les formules  $\perp$ ,  $\top$ ,  $p \vee \neg q$ ,  $p \wedge \neg q$ ,  $(p \vee q) \wedge \neg q$  sont en forme normale conjonctive.
- Les formules  $p \Rightarrow q$ ,  $\neg(p \wedge q)$ ,  $(p \wedge q) \vee q$ ,  $(p \wedge (q \vee r)) \vee q$  ne sont pas en forme normale conjonctive.

## Proposition

*Toute formule  $P$  propositionnelle admet une forme normale conjonctive, c'est-à-dire qu'il existe une formule  $Q$  équivalente à  $P$  et qui est en forme normale conjonctive.*

On montre que toute fonction booléenne  $f$  à  $n$  arguments peut être associée à une formule  $Q$  en forme normale conjonctive, telle que la table de vérité de  $Q$  correspond à  $f$ .

## Definition (Fonction booléenne associée à une formule, $[P]_{\mathbb{B}}$ )

Soit  $P$  une formule du calcul propositionnel qui contient  $n$  variables propositionnelles nommées  $p_1, \dots, p_n$ . La fonction booléenne associée à  $P$  est une fonction de  $\mathbb{B}^n \rightarrow \mathbb{B}$  notée  $[P]_{\mathbb{B}}$ .

Elle est définie par  $[P]_{\mathbb{B}}(b_1, \dots, b_n) = \text{val}(I_{\{b_1, \dots, b_n\}}, P)$  avec  $I_{\{b_1, \dots, b_n\}}$  l'interprétation dans laquelle chaque variable  $p_i$  a la valeur  $b_i$ .

La fonction  $[P]_{\mathbb{B}}$  correspond à la table de vérité de la formule  $P$ , dans laquelle les variables sont ordonnées suivant les colonnes et chaque ligne correspond à une entrée de la fonction.

- $P$  est valide ssi  $[P]_{\mathbb{B}}(b_1, \dots, b_n) = 1$  pour tout  $b_1 \dots b_n \in \mathbb{B}$
- $P$  est insatisfiable ssi  $[P]_{\mathbb{B}}(b_1, \dots, b_n) = 0$  pour tout  $b_1 \dots b_n \in \mathbb{B}$
- $P \equiv Q$  ssi  $[P]_{\mathbb{B}} = [Q]_{\mathbb{B}}$

## Definition (Fonction booléenne associée à une formule, $[P]_{\mathbb{B}}$ )

Soit  $P$  une formule du calcul propositionnel qui contient  $n$  variables propositionnelles nommées  $p_1, \dots, p_n$ . La fonction booléenne associée à  $P$  est une fonction de  $\mathbb{B}^n \rightarrow \mathbb{B}$  notée  $[P]_{\mathbb{B}}$ .

Elle est définie par  $[P]_{\mathbb{B}}(b_1, \dots, b_n) = \text{val}(I_{\{b_1, \dots, b_n\}}, P)$  avec  $I_{\{b_1, \dots, b_n\}}$  l'interprétation dans laquelle chaque variable  $p_i$  a la valeur  $b_i$ .

La fonction  $[P]_{\mathbb{B}}$  correspond à la table de vérité de la formule  $P$ , dans laquelle les variables sont ordonnées suivant les colonnes et chaque ligne correspond à une entrée de la fonction.

- $P$  est valide ssi  $[P]_{\mathbb{B}}(b_1, \dots, b_n) = V$  pour tout  $b_1 \dots b_n \in \mathbb{B}$
- $P$  est insatisfiable ssi  $[P]_{\mathbb{B}}(b_1, \dots, b_n) = F$  pour tout  $b_1 \dots b_n \in \mathbb{B}$
- $P \equiv Q$  ssi  $[P]_{\mathbb{B}} = [Q]_{\mathbb{B}}$

# Exemple

Soit la formule  $P \stackrel{\text{def}}{=} p_1 \vee p_2 \Rightarrow p_3$ .

La table de vérité et la fonction associée sont données ci-dessous

$p_1$	$p_2$	$p_3$	$P$
$\neg$	$\neg$	$V$	$V$
$V$	$\neg$	$F$	$F$
$F$	$V$	$F$	$F$
$F$	$F$	$F$	$V$

$$[P]_{\mathbb{B}}(V, V, V) = V$$

$$[P]_{\mathbb{B}}(V, V, F) = F$$

$$[P]_{\mathbb{B}}(V, F, V) = V$$

$$[P]_{\mathbb{B}}(V, F, F) = F$$

$$[P]_{\mathbb{B}}(F, V, V) = V$$

$$[P]_{\mathbb{B}}(F, V, F) = F$$

$$[P]_{\mathbb{B}}(F, F, V) = V$$

$$[P]_{\mathbb{B}}(F, F, F) = V$$

# Représentation d'une fonction booléenne

Question : une formule définit une fonction booléenne, le contraire est-il vrai ?

- soit  $f \in \mathbb{B}^n \rightarrow \mathbb{B}$
- on se donne  $n$  variables propositionnelles  $p_1, \dots, p_n$
- on cherche une formule  $P$  telle que  $[P]_{\mathbb{B}} = f$

Réponse :

- plusieurs formules  $P$  possibles

Nouvelle question :

- Qu'en est-il si on restreint la forme syntaxique de  $P$  ?



# Exemple

$p$	$q$	$f(p, q)$
$V$	$V$	$F$
$V$	$F$	$V$
$F$	$V$	$V$
$F$	$F$	$F$

- 1  $\Leftrightarrow, \neg :$
- 2  $\vee, \wedge, \neg :$
- 3  $\vee, \neg :$

# Exemple

$p$	$q$	$f(p, q)$
$V$	$V$	$F$
$V$	$F$	$V$
$F$	$V$	$V$
$F$	$F$	$F$

- 1  $\Leftrightarrow, \neg : \neg(p \Leftrightarrow q), \neg p \Leftrightarrow q$
- 2  $\vee, \wedge, \neg : (p \wedge \neg q) \vee (\neg p \wedge q), (\neg p \vee \neg q) \wedge (p \vee q)$
- 3  $\vee, \neg : \neg(\neg(\neg p \vee \neg q) \vee \neg(p \vee q))$

## Proposition

Pour toute fonction booléenne  $f$  à  $n$  arguments, on peut trouver une formule  $Q$  en forme normale conjonctive telle que  $f = [Q]_{\mathbb{B}}$ .

Pour une formule  $P$  quelconque, on calcule la FNC  $Q$  de la fonction booléenne  $[P]_{\mathbb{B}}$ . On a  $[P]_{\mathbb{B}} = [Q]_{\mathbb{B}}$ , donc  $P \equiv Q$ .

### Preuve:

- ensemble  $Z$  des  $n$ -uplets pour les lignes fausses de la table de vérité :  
 $Z \stackrel{\text{def}}{=} \{(b_1, \dots, b_n) \mid f(b_1, \dots, b_n) = F\}$
- si cet ensemble est vide alors  $f = [\top]_{\mathbb{B}}$
- formule en FNC sur  $p_1, \dots, p_n$  qui exclut les  $n$ -uplets de  $Z$ .
  - Pour  $(b_1, \dots, b_n) \in \mathbb{B}^n : l_1 \vee l_2 \vee \dots \vee l_n$  avec  $l_i = p_i$  si  $b_i = F$  et  $l_i = \neg p_i$  si  $b_i = V$  est vraie pour toutes les interprétations autre que  $(b_1, \dots, b_n)$ .
  - c'est une clause
  - on prend la conjonction des clauses pour chaque élément de  $Z$ .
  - c'est une formule en FNC qui est vraie exactement pour les interprétations qui ne correspondent pas à des éléments de  $Z$

# Exemple de calcul de FNC

Soit la formule  $P \stackrel{\text{def}}{=} (p_1 \Rightarrow p_2) \Rightarrow p_1$ , on commence par écrire la table de vérité :

$p_1$	$p_2$	$p_1 \Rightarrow p_2$	$(p_1 \Rightarrow p_2) \Rightarrow p_1$
V	V	V	V
V	F	F	V
F	V	V	F
F	F	V	F

- les lignes qui nous intéressent sont les deux dernières
- formule qui dit exactement que l'on n'est pas dans un de ces deux cas.
- décomposition par ligne
  - pas sur la ligne 3 :  $p_1 \vee \neg p_2$
  - pas sur la ligne 4 :  $p_1 \vee p_2$ .
- la forme normale conjonctive est :  $(p_1 \vee \neg p_2) \wedge (p_1 \vee p_2)$  qui est équivalent à  $p_1$ .

# Exercice

Donner les formes normales conjonctives des formules  $P$  et  $Q$  dont les tables de vérité sont les suivantes :

	$a$	$b$	$P$
1	$V$	$V$	$V$
2	$V$	$F$	$F$
3	$F$	$V$	$F$
4	$F$	$F$	$V$

	$a$	$b$	$c$	$Q$
1	$V$	$V$	$V$	$F$
2	$V$	$V$	$F$	$V$
3	$V$	$F$	$V$	$V$
4	$V$	$F$	$F$	$V$
5	$F$	$V$	$V$	$F$
6	$F$	$V$	$F$	$F$
7	$F$	$F$	$V$	$V$
8	$F$	$F$	$F$	$F$



*That's all Folks!*