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

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 : $0, 1, 2, 3, \ldots$
- 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)
  - ...

Enoncé paramétré et quantification

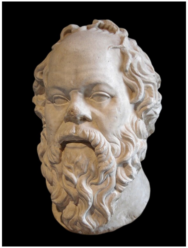
- Un énoncé logique *paramétré* représente une propriété (vraie ou fausse) d'un "*objet*" quelconque
- exemple : propriété pour un étudiant de *valider le cours de logique*
- "*valider le cours de logique*" sera vrai ou faux pour chaque étudiant considéré
  - *Toto valide le cours de logique*
- on peut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

La logique, chemin vers la vérité

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

Exemples de raisonnement

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



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

Chaos de la logique

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

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

Démonstration

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

Principes

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

Fondement des mathématiques

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

Meta-mathématique

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

Logique et informatique

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

Compétences logiques attendues

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

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

Compétences générales attendues

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

Plan du cours

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

Bibliographie

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

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

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

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

Gilles Dowek.
La logique.
Le Pommier, 2015.

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

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

Informations sur le cours

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

Déroulé du cours

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

L'équipe

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

Introduction au cours de logique

Jeu du Sudoku

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

Règles du jeu :

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

Jeu du Sudoku

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

Règles du jeu :

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

Solution :

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

Modélisation propositionnelle

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

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

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

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

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

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

Exemple, complèt (?)

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

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

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

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

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

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

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

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

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

Trouver une solution propositionnelle

- les formules peuvent être “simplifiées”
  - Ensemble de clauses (disjonctions)

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

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

Mise en œuvre

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

1. Davis–Putnam–Logemann–Loveland

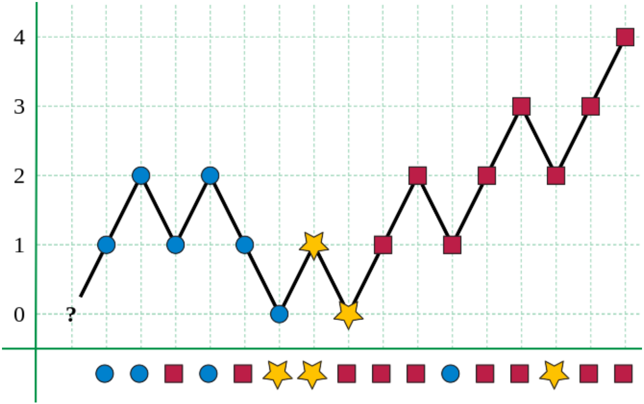
Modéliser, prouver

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

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

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

Principe de l'algorithme



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

Modélisation logique

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

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 nbcneq : forall  x:int. forall t i.
      0<=i<length t -> t[i]<>x -> nbc x t (i+1) = nbc x t i
function nb (x:int) (t:array int) : int = nbc x t (length t)

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

Preuve de programme en Why3 : programme annoté

Logique de Hoare (voir cours de GLA)

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

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


That's all Folks!

1–Maitriser le langage logique

1–Maitriser le langage logique

Langage formel

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

Introduction au cours de logique

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

Termes

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

Signature, termes

Definition (Signature, arité)

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

Definition (Terme)

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

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

Exemples de signature

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

Introduction au cours de logique

Formules atomiques

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

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

Exemples de signature : systèmes d'information

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

    $$\mathtt{ligne}(\mathtt{91-06}), \mathtt{arret}(\mathtt{Massy}), \mathtt{arret}(\mathtt{Saclay}), \mathtt{horaire}(\mathtt{06h00})\ldots$$
  - table qui tient à jour les rotations de bus avec le numéro de ligne, l'arret de départ, celui d'arrivée ainsi que l'horaire de départ : predicat *trajet* d'arité 4.

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

Choix des symboles de prédicat

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

Syntaxe et sémantique

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

Formules complexes

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

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

Et les quantificateurs du premier ordre :

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

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

Sens intuitif des connecteurs et quantificateurs

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

Logique et formules booléennes en programmation

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

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

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

Exercice

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

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

Exemples de formules complexes

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

Différentes catégories syntaxiques : termes

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

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

Différentes catégories syntaxiques : formules

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

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

Notations infixes

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

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

les parenthèses évitent les ambiguités

Notations, règles de parenthésage

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

Calcul des prédicats et calcul propositionnel

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

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

Exercice de traduction

Langage

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

Que signifient les formules suivantes en langage courant ?

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

Exemple de traduction

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

A savoir faire

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


That's all Folks!

La dernière fois

Definition (Signature, arité)

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

Definition (Terme)

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

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

La dernière fois

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

- $\top$ (top) : la propriété toujours vraie, tautologie ($0 = 0$)
- $\perp$ (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

La dernière fois

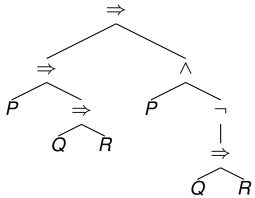
- On appelle logique du premier ordre (ou calcul des prédicats) le langage logique défini par une signature (symboles de fonctions et de prédicats) et qui n'utilise que les connecteurs logiques et les quantificateurs sur les variables de termes tels que définis précédemment.
- Le calcul propositionnel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

1–Maitriser le langage logique

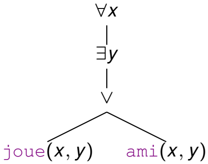
1. Définition du langage
2. Structure des formules
   - Représentation par des arbres
   - Règles de parenthésage
   - Variables libres et liées
3. Formules vraies
4. Théorie et modélisation
5. Définition récursive sur les formules

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



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



Exercice

représenter sous forme d'arbre les formules

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

Règles de précédence

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

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

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

Exercice : Règles de précédence

1. $P \wedge R \wedge \neg Q \Rightarrow P \vee Q \wedge R$
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 ambiguité

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

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

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

Variables liées

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

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

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

Variables libres

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

Variables libres

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

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

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

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

  - cette formule est-elle close ?

Variables libres

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

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

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

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

  - cette formule est-elle close ?

Formules syntaxiquement égales

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

Exercice : Formules syntaxiquement égales

Dire quelles formules sont égales ou différentes

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

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

Comparer les représentations sous forme d'arbre !

A savoir faire

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

1–Maitriser le langage logique

Valeurs de vérité

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

Formules (closes) vraies

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

Formules (closes) valides

- La valeur de vérité d'une formule (vrai ou faux) est définie par rapport à une interprétation des symboles qu'elle contient
- On ne peut pas dire *a priori* si une formule est vraie ou fausse
  - Par abus de langage, on dit parfois qu'une formule est vraie (resp. fausse) si elle est tout le temps vraie (resp. fausse)
- Une formule (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 résoud l'une des trois questions résoud les autres.

Proposition

Les trois propriétés suivantes sont équivalentes

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

Preuve:

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

Modèle et valeur de vérité

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

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

Exercice

Les formules suivantes sont-elles satisfiables ? valides ?

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

Modélisation propositionnelle de problèmes

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

Exercice : Enigme

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

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

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

Formules avec quantificateurs

- les *variables d'objets* représentent des inconnues dans un univers a priori indéterminé

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

Interprétation des formules avec quantificateurs

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

Vérité d'une formule quantifiée

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

Interprétation des symboles

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

Validité, satisfiabilité

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

Formulles valides

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

Remarque

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

Modélisation en logique du premier ordre I

Trouver l'erreur

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

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

A savoir faire

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

Ces notions seront approfondies dans le chapitre 2

Pourquoi des théories ?

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

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

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

Definition (Théorie)

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

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

Axiomes pour les groupes I

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

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

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

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

Axiomes pour la géométrie

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

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

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

Théorie de l'égalité

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

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

Exemples

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

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

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

Egalité

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

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

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

Théorie arithmétique (Peano)

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

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

Illustration

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

Application de la théorie de l'arithmétique

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

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

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


That's all Folks!

La dernière fois : théories

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 infixe.

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

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 $\mathtt{pos}(i, j, k)$ vrai lorsque le chiffre k est à la position (i, j) et l'égalité
- Au plus un seul chiffre k par case (i, j) :

$$\forall i\, j\, k_1\, k_2, \mathtt{pos}(i, j, k_1) \land \mathtt{pos}(i, j, k_2) \Rightarrow k_1 \neq k_2$$

- 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 ?

Savoir faire

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

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 : $\top$, $\bot$, 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 :

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

- 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 :

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

- 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 :

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

- 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 récursivement $\texttt{Vars}(P)$ ensemble des variables propositionnelles qui apparaissent dans P

$$
\begin{aligned}
\texttt{Vars}(X) &= & & X \in \mathcal{V}_p \\
\texttt{Vars}(\bot) &= & & \\
\texttt{Vars}(\top) &= & & \\
\texttt{Vars}(\neg P) &= & & \\
\texttt{Vars}(P_1 \circ P_2) &= & & \circ \in \{\wedge, \vee, \Rightarrow\}
\end{aligned}
$$

Restriction au calcul propositionnel

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

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

Substitution

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

$$
\begin{array}{lll}
X[X \leftarrow Q] & = & \quad X \in \mathcal{V}_p \\
P[X \leftarrow Q] & = & \quad P \text{ est atomique } P \neq X \\
(\neg A)[X \leftarrow Q] & = & \\
(A \circ B)[X \leftarrow Q] & = & \quad \circ \in \{\wedge, \vee, \Rightarrow\}
\end{array}
$$

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 :

$$\begin{array}{lll}
X[X\leftarrow Q] & = Q & X \in \mathcal{V}_p \\
P[X\leftarrow Q] & = P & P \text{ est atomique } P \neq X \\
(\neg A)[X\leftarrow Q] & = \neg(A[X\leftarrow Q]) & \\
(A \circ B)[X\leftarrow Q] & = (A[X\leftarrow Q] \circ B[X\leftarrow Q]) & \circ \in \{\wedge, \vee, \Rightarrow\}
\end{array}$$

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 $I \in \mathcal{V}_p \to \mathbb{B}$, et on définit $\mathtt{val}(I, P) \in \mathbb{B}$

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

Exemple

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

Propriété

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

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

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 $\phi(\bot)$ 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.

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

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

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

Preuve par récurrence

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

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

     - 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 $\mathtt{nbatom}(P) \leq 1 + \mathtt{nbsymbp}(P)$.

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 $X \notin \texttt{Vars}(P)$, on a $P[X \leftarrow Q] = P$
- La propriété $\phi(P)$ à montrer par récurrence structurelle sur P :
  si $X \notin \texttt{Vars}(P)$ alors $P[X \leftarrow Q] = P$
- Examen de chacun des cas possibles pour la formule P

Preuve détaillée

1. P formule atomique :
   Comme $X \notin \texttt{Vars}(P)$, 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 $X \notin \texttt{Vars}(A)$ alors $A[X \leftarrow Q] = A$
   - Comme $X \notin \texttt{Vars}(P)$ et $\texttt{Vars}(\neg A) = \texttt{Vars}(A)$, on a que $X \notin \texttt{Vars}(A)$.
   - 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 $X \notin \texttt{Vars}(A)$ alors $A[X \leftarrow Q] = A$ et de même pour B, si $X \notin \texttt{Vars}(B)$ alors $B[X \leftarrow Q] = B$
   - Comme $X \notin \texttt{Vars}(P)$ et $\texttt{Vars}(A \circ B) = \texttt{Vars}(A) \cup \texttt{Vars}(B)$, on a que $X \notin \texttt{Vars}(A)$ et $X \notin \texttt{Vars}(B)$.
   - 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 $X \notin \texttt{Vars}(P)$ alors $P[X \leftarrow Q] = P$


That's all Folks!

La dernière fois

- 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 :

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

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

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

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

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

La dernière fois : preuve par récurrence

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

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

     - 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 $\mathtt{nbatom}(P) \leq 1 + \mathtt{nbsymbp}(P)$.

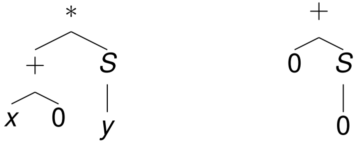
Exercice TD : Sous-formules

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 apparait 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.

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

Exemple

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

$$
\begin{aligned}
\texttt{clos}(x) &= \texttt{faux si } x \texttt{ est une variable} \\
\texttt{clos}(\texttt{c}) &= \texttt{vrai} \\
\texttt{clos}(\texttt{f}(t)) &= \texttt{clos}(t) \\
\texttt{clos}(\texttt{g}(t, u)) &= \texttt{clos}(t) \texttt{ et } \texttt{clos}(u)
\end{aligned}
$$

Définition récursive sur les termes

Definition

$\mathcal{F}$ signature, $\mathcal{X}$ ensemble de variables et $\mathcal{D}$ un ensemble quelconque.
Pour définir une application $G \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \to \mathcal{D}$, on se donne :

1. Une application V dans $\mathcal{X} \to \mathcal{D}$;
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

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

.

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

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

Fonctions génériques utiles

Exemple (Taille d'un terme)

$\mathtt{size}$ *compte le nombre de symboles dans un terme.*

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

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

Exemple (Hauteur d'un terme)

$\mathtt{ht}$ *: compte le nombre maximal de symboles imbriqués dans un terme.*

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

$\mathtt{ht}(t) = 3$.

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

Substitution sur les termes

- remplacement simultanée de plusieurs variables par des termes.
- application $\sigma \in \mathcal{X} \to \mathcal{T}(\mathcal{F}, \mathcal{X})$, appelée substitution qui associe un terme à chaque variable.
- On note $\{x_1 \leftarrow u_1; \ldots; x_n \leftarrow u_n\}$ la substitution σ 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 σ est de la forme $\{x_1 \leftarrow u_1; \ldots; x_n \leftarrow u_n\}$, alors le terme $t[\sigma]$ sera noté $t[x_1 \leftarrow u_1; \ldots; x_n \leftarrow u_n]$.

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

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

Propriétés de la substitution

- le résultat de $t[\sigma]$ ne dépend que de la valeur de la substitution σ sur les variables de t.
- soit deux substitutions σ_1 et σ_2 et un terme t, si pour toute variable $x \in \mathtt{vars}(t)$ 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.

Récurrence 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 $\phi(t_1) \cdots \phi(t_n)$ 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 $\mathtt{ht}(t) \leq \mathtt{size}(t)$
La preuve se fait par récurrence sur la structure du terme t.

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

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

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

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

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

On en déduit que $\mathtt{ht}(t) \leq \mathtt{size}(t)$ 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 :
    - $\bot[x \leftarrow t] = \bot$
    - $\top[x \leftarrow t] = \top$
    - $R(t_1, \ldots, t_n)[x \leftarrow t] = R(t_1[x \leftarrow t], \ldots, t_n[x \leftarrow t])$
  - $(\neg A)[x \leftarrow t] = \neg(A[x \leftarrow t])$
  - $\circ \in \{\wedge, \vee, \Rightarrow\}$: $(A \circ B)[x \leftarrow t] = (A[x \leftarrow t]) \circ (B[x \leftarrow t])$
  - $(\forall y, Q)[x \leftarrow t]$:
    - 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 $y \notin \mathtt{vars}(t)$ 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
      définition partielle
    - $(\exists y, Q)[x \leftarrow t]$: 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. $(\forall y, R(x, y))[y \leftarrow x]$

Remarques

- 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 faire

- 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

Introduction

- 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 $f_I : \mathcal{D}^n \to \mathcal{D}$: pour chaque $v_1, \ldots, v_n \in \mathcal{D}$, on a $f_I(v_1, \ldots, v_n) \in \mathcal{D}$) ;
- 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 $(v_1, \ldots, v_n)$ 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.

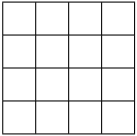
Exemple : interprétation

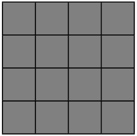
Signature : prédicat binaire P.

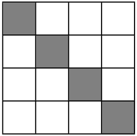
Domaine $D \stackrel{\text{def}}{=} \{1, 2, 3, 4\}$.
L'interprétation de P donnée par un tableau 4×4.
La formule atomique $P(t, u)$ est vraie si t vaut i et u vaut j et que la case sur la ligne i et la colonne j est grisée.
Soient les formules :

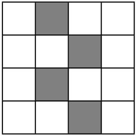
1. $\exists x, \neg P(x, x)$
2. $\exists x, \forall y, P(x, y)$
3. $\forall x, \exists y, P(x, y)$
4. $\forall x\, y, P(x, y) \Rightarrow P(y, x)$

Donner la valeur des formules pour chaque interprétation de P :


(1)


(2)


(3)


(4)

Exemple des rationnels

- signature pour manipuler des rationnels
  - constantes $\mathtt{0}$, $\mathtt{1}$ et $\mathtt{-1}$,
  - une opération binaire de construction `frac`,
  - des opérations binaires $+$ et $\star$
  - 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

$$
\begin{aligned}
\mathtt{0}_{\mathbb{Q}} &\stackrel{\text{def}}{=} (0,1) \\
\mathtt{1}_{\mathbb{Q}} &\stackrel{\text{def}}{=} (1,1) \\
\mathtt{-1}_{\mathbb{Q}} &\stackrel{\text{def}}{=} (-1,1) \\
\mathtt{frac}_{\mathbb{Q}}((p_1,q_1),(p_2,q_2)) &\stackrel{\text{def}}{=} (p_1q_2, q_1p_2) && \text{si } p_2 > 0 \\
\mathtt{frac}_{\mathbb{Q}}((p_1,q_1),(p_2,q_2)) &\stackrel{\text{def}}{=} (0,1) && \text{si } p_2 = 0 \\
\mathtt{frac}_{\mathbb{Q}}((p_1,q_1),(p_2,q_2)) &\stackrel{\text{def}}{=} (-p_1q_2, -q_1p_2) && \text{si } p_2 < 0 \\
+_{\mathbb{Q}}((p_1,q_1),(p_2,q_2)) &\stackrel{\text{def}}{=} (p_1q_2 + p_2q_1, q_1q_2) \\
*_{\mathbb{Q}}((p_1,q_1),(p_2,q_2)) &\stackrel{\text{def}}{=} (p_1p_2, q_1q_2) \\
=_{\mathbb{Q}}((p_1,q_1),(p_2,q_2)) &\stackrel{\text{def}}{=} (p_1q_2 = q_1p_2)
\end{aligned}
$$

Exemple base de données

- 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.

Environnement

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

Exemple Dans le modèle usuel des entiers : $\exists n, \texttt{pair}(3 \times n + 1)$ est vrai, $\forall n, \texttt{pair}(3 \times n + 1)$ est faux mais la valeur de vérité de $\texttt{pair}(3 \times n + 1)$ 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 ρ qui associe une valeur du domaine $\mathcal{D}$ à chaque variable de $\mathcal{X}$ (c'est à dire $\rho : \mathcal{X} \to \mathcal{D}$).

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

$$\mathtt{val}_I(\rho, x) = \rho(x) \qquad \mathtt{val}_I(\rho, f(t_1, \ldots, t_n)) = f_I(\mathtt{val}_I(\rho, t_1), \ldots, \mathtt{val}_I(\rho, t_n))$$

Valeur d'une formule dans une interprétation

Definition (Valeur d'une formule)

Soit ρ un environnement, $\rho : \mathcal{X} \to \mathcal{D}$.
On définit la valeur $\mathtt{val}_I(\rho, P)$ d'une formule P dans l'interprétation I et l'environnement ρ, c'est une valeur de vérité dans $\{V, F\}$:

$$
\begin{aligned}
\mathtt{val}_I(\rho, \bot) &= F \\
\mathtt{val}_I(\rho, \top) &= V \\
\mathtt{val}_I(\rho, R(t_1, \ldots, t_n)) &= \textbf{si } R_I(\mathtt{val}_I(\rho, t_1), \ldots, \mathtt{val}_I(\rho, t_n)) \textbf{ alors } V \textbf{ sinon } F \\
\mathtt{val}_I(\rho, \neg A) &= \textbf{si } \mathtt{val}_I(\rho, A) = V \textbf{ alors } F \textbf{ sinon } V \\
\mathtt{val}_I(\rho, A \wedge B) &= \textbf{si } \mathtt{val}_I(\rho, A) = V \textbf{ alors } \mathtt{val}_I(\rho, B) \textbf{ sinon } F \\
\mathtt{val}_I(\rho, A \vee B) &= \textbf{si } \mathtt{val}_I(\rho, A) = V \textbf{ alors } V \textbf{ sinon } \mathtt{val}_I(\rho, B) \\
\mathtt{val}_I(\rho, A \Rightarrow B) &= \textbf{si } \mathtt{val}_I(\rho, A) = V \textbf{ alors } \mathtt{val}_I(\rho, B) \textbf{ sinon } V \\
\mathtt{val}_I(\rho, \forall x, A) &= \textbf{si } \text{pour tout } d \in \mathcal{D}, \mathtt{val}_I(\rho + \{x \mapsto d\}, A) = V \\
&\quad\ \textbf{alors } V \textbf{ sinon } F \\
\mathtt{val}_I(\rho, \exists x, A) &= \textbf{si } \text{il existe } d \in \mathcal{D} \text{ tel que } \mathtt{val}_I(\rho + \{x \mapsto d\}, A) = V \\
&\quad\ \textbf{alors } V \textbf{ sinon } F
\end{aligned}
$$


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 $f_I : \mathcal{D}^n \to \mathcal{D}$: 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 $\star$
  - un symbole de prédicat binaire pour l'égalité.
- interprétation de cette signature : représentation d'un rationnel
  - un couple $(p, q) \in \mathbb{Z} \times \mathbb{N} \backslash \{0\}$ formé d'un entier relatif en numérateur et d'un entier naturel non nul en dénominateur.
  - Le domaine de l'interprétation est donc $\mathcal{D} \stackrel{\text{def}}{=} \mathbb{Z} \times \mathbb{N} \backslash \{0\}$.
- 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

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

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 ρ qui associe une valeur du domaine $\mathcal{D}$ à chaque variable de $\mathcal{X}$ (c'est à dire $\rho : \mathcal{X} \to \mathcal{D}$).
Soit ρ 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 ρ un environnement, $\rho \in \mathcal{X} \to \mathcal{D}$.
On définit la valeur $\mathtt{val}_I(\rho, t)$ d'un terme $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ dans l'interprétation I et l'environnement ρ, c'est un élément du domaine $\mathcal{D}$:

$$\mathtt{val}_I(\rho, x) = \rho(x) \qquad \mathtt{val}_I(\rho, f(t_1, \ldots, t_n)) = f_I(\mathtt{val}_I(\rho, t_1), \ldots, \mathtt{val}_I(\rho, t_n))$$

La dernière fois : Environnement et évaluations, bis

Definition (Valeur d'une formule)

Soit ρ un environnement, $\rho : \mathcal{X} \to \mathcal{D}$.
On définit la valeur $\mathtt{val}_I(\rho, P)$ d'une formule P dans l'interprétation I et l'environnement ρ, c'est une valeur de vérité dans $\{V, F\}$:

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

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 ρ, c'est-à-dire lorsque $\mathtt{val}_I(\rho, A) = V$.

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

- On note $I, \rho \not\models A$ dans le cas contraire lorsque la formule A est fausse dans l'interprétation I et l'environnement ρ, c'est-à-dire lorsque $\mathtt{val}_I(\rho, A) = F$.
- 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 ρ un environnement.

- $I, \rho \not\models \bot$
- $I, \rho \models \top$
- $I, \rho \models R(t_1, \dots, t_n)$ *si et seulement si* $(\mathtt{val}_I(\rho, t_1), \dots, \mathtt{val}_I(\rho, t_n))$ *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* $\mathtt{val}_I(\rho, A) = V$ □

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 $y \notin \mathrm{vl}(P)$ et $P[x \leftarrow y]$ est bien définie alors
  $\mathtt{val}_I(\rho, (\forall x, P)) = \mathtt{val}_I(\rho, (\forall y, P[x \leftarrow y]))$ et
  $\mathtt{val}_I(\rho, (\exists x, P)) = \mathtt{val}_I(\rho, (\exists y, P[x \leftarrow y]))$
- 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 $\mathtt{val}_I(P)$ au lieu de $\mathtt{val}_I(\rho, P)$ et $I \models P$ au lieu de $I, \rho \models P$.

2–Donner du sens aux formules

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 ρ :

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

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

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

Preuve I

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

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

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

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

$$\begin{aligned}
&\mathtt{val}_I(\rho, f(u_1[x \leftarrow t], \ldots, u_n[x \leftarrow t])) \\
&= f_I(\mathtt{val}_I(\rho, u_1[x \leftarrow t]), \ldots, \mathtt{val}_I(\rho, u_n[x \leftarrow t])) && (\mathtt{val}_I) \\
&= f_I(\mathtt{val}_I(\rho + \{x \mapsto \mathtt{val}_I(\rho, t)\}, u_1), \ldots, \mathtt{val}_I(\rho + \{x \mapsto \mathtt{val}_I(\rho, t)\}, u_n)) && (\mathrm{HR}) \\
&= \mathtt{val}_I(\rho + \{x \mapsto \mathtt{val}_I(\rho, t)\}, f(u_1, \ldots, u_n)) && (\mathtt{val}_I)
\end{aligned}$$

Preuve II

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

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

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

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

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

  $\mathtt{val}_I(\rho + \{z \mapsto d\}, A[x \leftarrow t]) = V$ ssi
  $\mathtt{val}_I((\rho + \{x \mapsto \mathtt{val}_I(\rho, t)\}) + \{z \mapsto d\}, A) = V$
  donc $\mathtt{val}_I(\rho, (\forall z, (A[x \leftarrow t]))) = V$ ssi
  $\mathtt{val}_I(\rho + \{x \mapsto \mathtt{val}_I(\rho, t)\}, (\forall z, A)) = V$, qui est le résultat souhaité.
  l'HR est appliquée à $\rho + \{z \mapsto d\}$ (et pas ρ). La propriété à montrer par récurrence sur P doit permettre de faire varier l'environnement.

2–Donner du sens aux formules

Validité et satisfiabilité

- 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 environnnement 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.

Définitions

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 ρ, 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 ρ 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 ρ 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 ρ, 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 $\mathcal{E}$).

Exercice

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. $\{(\forall x, P(x)) \Rightarrow q, \forall x, (P(x) \wedge \neg q)\}$

Validité et variables libres

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.* $\square$

Propriétés

- 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 $P_1 \land \dots \land P_n$
- 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

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 $\mathcal{E}[x \leftarrow t] \stackrel{\text{def}}{=} \{P[x \leftarrow t] | P \in \mathcal{E}\}$ est valide (resp. insatisfiable).*

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

Remplacer une variable par un terme

Les réciproques sont fausses. Soit un prédicat unaire Q et une constante $\mathtt{c}$.

- La formule $Q(x) \Rightarrow Q(\mathtt{c})$ n'est pas valide.
  - interprétation dont le domaine contient deux éléments $\{0, 1\}$, on interprète la constante $\mathtt{c}$ 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 $Q(x) \Rightarrow Q(\mathtt{c})[x \leftarrow c]$ qui est égale à $Q(\mathtt{c}) \Rightarrow Q(\mathtt{c})$ est valide.
- La formule $Q(x) \wedge \neg Q(\mathtt{c})$ est satisfiable (prendre la même interprétation que précédemment)
- La formule $Q(x) \wedge \neg Q(\mathtt{c})[x \leftarrow c]$ qui est égale à $Q(\mathtt{c}) \wedge \neg Q(\mathtt{c})$ 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 ρ 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). $\square$

Remplacer un prédicat par une formule paramétrée

- On généralise au replacement 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 $x_1, \ldots, x_n$. On remplace dans la formule P toutes les sous-formules atomiques de la forme $R(t_1, \ldots, t_n)$ par la formule $Q[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$.
- On note le résultat de cette opération $P[R(x_1, \ldots, x_n) \leftarrow Q]$.

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

Preuve: *(validité)* I *interprétation des symboles de* $P[R(x_1, \ldots, x_n) \leftarrow Q]$ *et* ρ *un environnement. On veut montrer que* $I, \rho \models P[R(x_1, \ldots, x_n) \leftarrow Q]$
Tous les symboles et les variables qui apparaissent dans P *apparaissent aussi dans* $P[R(x_1, \ldots, x_n) \leftarrow Q]$ *à l'exception peut-être de* R. *On construit une interprétation* I' *qui est définie comme* I *sauf pour* R *:*
on pose $R_{I'}(d_1, \ldots, d_n)$ *est vrai ssi* $I, (\rho + \{x_1 \mapsto d_1, \ldots, x_n \mapsto d_n\}) \models Q$.
On a $I', \rho \models P$ *ssi* $I, \rho \models P[R(x_1, \ldots, x_n) \leftarrow Q]$ *(preuve par récurrence sur la structure de* P*).*
Si P *est valide, on a* $I', \rho \models P$. *On en déduit que* $I, \rho \models P[R(x_1, \ldots, x_n) \leftarrow Q]$ *et comme cela vaut pour tout* I, ρ, *on en déduit* $\models P[R(x_1, \ldots, x_n) \leftarrow Q]$. $\square$

A savoir

- 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

La dernière fois

- 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 ρ, c'est-à-dire lorsque $\mathtt{val}_I(\rho, A) = V$.

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

- On note $I, \rho \not\models A$ dans le cas contraire lorsque la formule A est fausse dans l'interprétation I et l'environnement ρ, c'est-à-dire lorsque $\mathtt{val}_I(\rho, A) = F$.
- 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 ρ 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 ρ 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 ρ, 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

Conséquence logique, équivalence

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

Soient $\mathcal{E}$ un ensemble de formules, I une interprétation et ρ 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 ρ 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 ρ, on a $I, \rho \models A$ si et seulement si $I, \rho \models B$.

Propriétés

Proposition

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

1. $A_1, \ldots, A_n \models P$ *ssi* $A_1 \land \ldots \land A_n \Rightarrow P$ *est valide*
2. *si* $\mathcal{E}$ *est insatisfiable alors pour toute formule* P, $\mathcal{E} \models P$
3. $\mathcal{E}$ *est insatisfiable ssi* $\mathcal{E} \models \bot$
4. $\mathcal{E} \models P$ *ssi* $\mathcal{E} \cup \{\neg P\}$ *est insatisfiable.*
5. $\mathcal{E}, P \models Q$ *ssi* $\mathcal{E} \models P \Rightarrow Q$

Vérité et substitution

La relation de conséquence logique est stable par substitution et remplacement. On étend les notations $P[x \leftarrow t]$ et $P[R(x_1, \dots, x_n) \leftarrow Q]$ a 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
$A_1, \dots, A_n \models P$	
vrai	$A_1[x \leftarrow Q], \dots, A_n[x \leftarrow Q] \models P[x \leftarrow Q]$
	vrai

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'apparait 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

Rappel Équivalence

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 :

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

- 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 $\bot$ sont des éléments neutres ou absorbants :

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

Certains connecteurs peuvent se définir en fonction d'autres :

$$\neg P \equiv P \Rightarrow \bot \qquad P \Rightarrow Q \equiv \neg P \vee Q$$
$$P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P) \equiv (P \wedge Q) \vee (\neg P \wedge \neg Q)$$

Exercice

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

Exercice

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

1. $\bot \Rightarrow P \equiv \top$
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 :

$$\begin{aligned} \neg\bot &\equiv \top & \qquad \neg(P \wedge Q) &\equiv \neg P \vee \neg Q \\ \neg\top &\equiv \bot & \neg(P \vee Q) &\equiv \neg P \wedge \neg Q \\ \neg\neg P &\equiv P \end{aligned}$$

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

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'apparait 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 $x \notin \mathrm{vl}(A)$ alors $\forall x, A \equiv \exists x, A \equiv A$.
- "sortir" d'un quantificateur une sous-formule qui n'utilise pas la variable.
  Si $x \notin \mathrm{vl}(A)$ 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 $x \notin \mathrm{vl}(A)$ 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 $x \notin \mathrm{vl}(A)$ 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 $x\ \texttt{boit}$ et pour A la formule $\forall x, x\ \texttt{boit}$

Exercice : contre-exemples

Trouver des interprétations qui justifient les résultats suivants :

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

A savoir

- 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

La dernière fois

- 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

Introduction

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.

Modèle fini

On s'intéresse à un domaine fini particulier $\mathcal{D} = \{d_1; \ldots; d_n\}$.
On supposera (quitte à ajouter des constantes) que l'on a dans le langage des termes sans variable $\{c_1; \ldots; c_n\}$ tels que $\texttt{val}(c_i) = d_i$.
Dans une telle interprétation, pour toute formule P et toute variable libre x dans P, les deux formules suivantes sont vraies.

$$(\forall x, P) \Leftrightarrow (P[x \leftarrow c_1] \wedge \ldots \wedge P[x \leftarrow c_n])$$
$$(\exists x, P) \Leftrightarrow (P[x \leftarrow c_1] \vee \ldots \vee P[x \leftarrow c_n])$$

Cette technique permet d'éliminer les quantificateurs d'une formule close et d'obtenir une formule propositionnelle sans variable.
On remplace chaque formule atomique $R(t_1, \ldots, t_p)$ par $R(c_{i_1}, \ldots, c_{i_p})$ si $\texttt{val}(t_k) = d_{i_k}$.
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 $\{c_1; \ldots; c_n\}$ 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 $\mathtt{val}_I(\rho, t) = \mathtt{val}_I(\rho, u)$.

1. Que peut-on dire du cardinal du domaine d'une interprétation qui rend vraie la formule $\forall x, x = c_1 \lor \ldots \lor x = c_n$?
2. Que peut-on dire du cardinal du domaine d'une interprétation qui rend vraie la formule $c_1 \neq c_2 \land \ldots c_1 \neq c_n \land c_2 \neq c_3 \land \ldots \land c_{n-1} \neq c_n$?
3. Même question pour la formule $\exists x_1 \ldots x_n, x_1 \neq x_2 \land \ldots x_1 \neq x_n \land x_2 \neq x_3 \land \ldots \land x_{n-1} \neq x_n$?
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
   - $\mathcal{E}, (\forall x, x = c_1 \lor \ldots \lor x = c_n), (P(c_1) \land \ldots \land P(c_n)) \models \forall x, P(x)$
   - $\mathcal{E}, (\forall x, x = c_1 \lor \ldots \lor x = c_n), (\exists x, P(x)) \models P(c_1) \lor \ldots \lor P(c_n)$

2–Donner du sens aux formules

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.

Modèle des termes clos

- 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))`,. . ., $\mathtt{f}^n(\mathtt{a})$,. . .

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 interpréte 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 :

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

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 $\mathtt{val}(H, t) = t$

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

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 $\mathtt{a}$*, un symbole de fonction* $\mathtt{f}$*, un symbole de prédicat unaire* P *et un symbole de prédicat binaire* R*, la base de Herbrand est composé des formules atomiques*
$P(\mathtt{a})$,$P(\mathtt{f(a)})\dots P(\mathtt{f}^n(\mathtt{a}))$,…,$R(\mathtt{a},\mathtt{a})$,$R(\mathtt{f}(\mathtt{a}),\mathtt{a})$,…,$R(\mathtt{f}^n(\mathtt{a}),\mathtt{f}^p(\mathtt{a}))$,…

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.

Interprétation de Herbrand : prédicats

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. $\forall x\, y, R(x, a) \land \neg R(b, y)$
2. $\forall x\, y, R(x, a) \Rightarrow R(y, f(y))$

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

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. $\forall x\, y, R(x, a) \land \neg R(b, y)$
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 : $B_H = \{R(a, a), R(a, f(a)), R(f(a), a), R(f(a), f(a)), \dots\}$

Cas des formules sans quantificateur

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* $\{R(t_1, \dots, t_n) | I \models R(t_1, \dots, t_n)\}$.
- *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*

$$\mathtt{val}_H(P) = \mathtt{val}_I(P) = V$$

Satisfiabilité d'une formule universelle

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.

Preuve

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 $I \models \forall x_1 \ldots x_n, A$.
On construit le modèle de Herbrand H tel que $H \models R(t_1, \ldots, t_n)$ ssi $I \models R(t_1, \ldots, t_n)$.
On doit montrer $H \models \forall x_1 \ldots x_n, A$.
Soit $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F})$, $\iota \stackrel{\text{def}}{=} \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, montrons $H, \iota \models A$.
Or $\mathtt{val}_H(\iota, A) = \mathtt{val}_H(A[x_1 \leftarrow t_1; \ldots, x_n \leftarrow t_n])$ 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 ($\mathtt{val}_H(t) = t$).
H et I ont même valeur sur les formules atomiques, $A[x_1 \leftarrow t_1; \ldots, x_n \leftarrow t_n]$ est une formule propositionnelle close, et $I \models \forall x_1 \ldots x_n, A$, on a donc :

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

Satisfiabilité et instances closes

Definition (Instance close d'une formule)

Soit une formule A dont les variables libres sont $\{x_1, \ldots, x_n\}$.
Une formule de la forme $A[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$ avec $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F})$ est appelée instance close de A.

Proposition (Théorème de Herbrand)

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

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

Preuve

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* $\{A[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n] | t_1, \dots, t_n \in \mathcal{T}(\mathcal{F})\}$.
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 assemblant 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* $t_1, \dots, t_n \in \mathcal{T}(F)$ *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.*

Exercice

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 ?*

Exercice

- *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 ?*

Conséquences

- 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

Conclusion

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 syntaxique.

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.

A savoir faire

- 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

Suite

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.


That's all Folks!

3–Manipuler les formules

1. Formes normales
2. Représentations alternatives des formules
3. Systèmes de déduction

Introduction

- aborder la logique et les formules sur le plan *syntaxique*
- 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 clausale).
  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
   - Skolémisation
2. Représentations alternatives des formules
3. Systèmes de déduction

Formes normales

- 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 ($A \Rightarrow B \equiv \neg A \lor B$)
  - on peut aussi tout faire avec $\Rightarrow$ et $\bot$
  - … ou avec le connecteur de *Sheffer*
- Une forme normale est une manière *standardisée* de représenter une formule
  - la forme normale est *syntaxiquement* 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…)
    - $$\begin{aligned} x+3-2+x^2+x &= x^2+2x+1 \\ &= (x+1)^2 \\ &= (x+2)x+1 \end{aligned}$$

Littéraux

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 $\bot$.
- 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 $\top, \bot, \vee$ 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 :

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

Forme normale de négation : algorithme

$$
\begin{aligned}
\mathtt{fnn}(p) &= && \text{si } p \text{ atomique}\\
\mathtt{fnn}(\neg A) &= \mathtt{neg}(A) &&\\
\mathtt{fnn}(A \circ B) &= && \circ \in \{\vee, \wedge\}\\
\mathtt{fnn}(A \Rightarrow B) &= &&\\
\mathtt{fnn}(\forall x, A) &= &&\\
\mathtt{fnn}(\exists x, A) &= &&
\end{aligned}
$$

$$
\begin{aligned}
\mathtt{neg}(\top) &= &&\\
\mathtt{neg}(\bot) &= &&\\
\mathtt{neg}(R(t_1, \dots, t_n)) &= && R \text{ symbole de prédicat}\\
\mathtt{neg}(\neg A) &= &&\\
\mathtt{neg}(A \wedge B) &= &&\\
\mathtt{neg}(A \vee B) &= &&\\
\mathtt{neg}(A \Rightarrow B) &= &&\\
\mathtt{neg}(\forall x, A) &= &&\\
\mathtt{neg}(\exists x, A) &= &&
\end{aligned}
$$

Forme normale de négation : algorithme

$$\begin{array}{lll}
\texttt{fnn}(p) & = p & \text{si } p \text{ atomique} \\
\texttt{fnn}(\neg A) & = \texttt{neg}(A) & \\
\texttt{fnn}(A \circ B) & = \texttt{fnn}(A) \circ \texttt{fnn}(B) & \circ \in \{\vee, \wedge\} \\
\texttt{fnn}(A \Rightarrow B) & = \texttt{neg}(A) \vee \texttt{fnn}(B) & \\
\texttt{fnn}(\forall x, A) & = \forall x, \texttt{fnn}(A) & \\
\texttt{fnn}(\exists x, A) & = \exists x, \texttt{fnn}(A) & \\
\end{array}$$

$$\begin{array}{lll}
\texttt{neg}(\top) & = \bot & \\
\texttt{neg}(\bot) & = \top & \\
\texttt{neg}(R(t_1, \dots, t_n)) & = \neg R(t_1, \dots, t_n) & R \text{ symbole de prédicat} \\
\texttt{neg}(\neg A) & = \texttt{fnn}(A) & \\
\texttt{neg}(A \wedge B) & = \texttt{neg}(A) \vee \texttt{neg}(B) & \\
\texttt{neg}(A \vee B) & = \texttt{neg}(A) \wedge \texttt{neg}(B) & \\
\texttt{neg}(A \Rightarrow B) & = \texttt{fnn}(A) \wedge \texttt{neg}(B) & \\
\texttt{neg}(\forall x, A) & = \exists x, \texttt{neg}(A) & \\
\texttt{neg}(\exists x, A) & = \forall x, \texttt{neg}(A) & \\
\end{array}$$

Preuve de l'algorithme

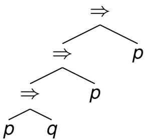
Preuve:

- *On montre par récurrence structurelle sur la formule P que $\mathtt{fnn}(P)$ et $\mathtt{neg}(P)$ sont en forme normale de négation et que de plus $\mathtt{fnn}(P) \equiv P$ et $\mathtt{neg}(P) \equiv \neg P$.*
- *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 $\mathtt{neg}$ appliquée à une instance de prédicat.*
    - *formule $a_0 \Rightarrow a_1 \ldots \Rightarrow a_n$ (n connecteurs logiques) : la forme normale de négation $\neg a_0 \vee \neg a_1 \ldots \vee a_n$ en a $2n$.*
    - *c'est le pire cas : si P a n connecteurs, le nombre de connecteurs de $\mathtt{fnn}(P)$ est au plus $2n$ et le nombre de connecteurs de $\mathtt{neg}(P)$ est au plus $2n + 1$*
    - *un littéral est souvent traité "directement", sans plus de complexité qu'une formule atomique.*

$\square$

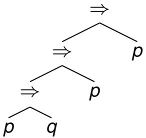
Exemple

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

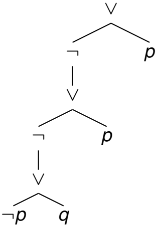


- suppression de $\Rightarrow$:

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



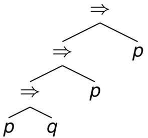
- suppression de $\Rightarrow$:



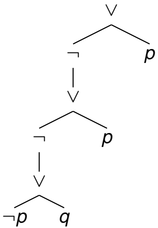
- forme normale de négation :

Exemple

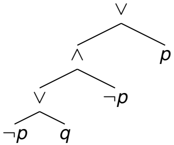
- $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$



- suppression de $\Rightarrow$:



- forme normale de négation :



3–Manipuler les formules

Clauses

Definition (Clause)

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

Un litteral tout seul est une clause dans laquelle il n'y a pas de disjonction.
$\top$ et $\bot$ sont des clauses.
Etre une clause est une propriété syntaxique des formules. Une formule peut être équivalente à une clause sans être une clause elle-même.

Exemple

Soi p, q des variables propositionnelles.

- $\bot, p, \top, \neg p, p \vee q, p \vee q \vee \neg p$ *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 $P \vee P \equiv P$, $P \vee \neg P \equiv \top$, $P \vee \top \equiv \top$, $P \vee \bot \equiv P$.
- Une clause peut donc soit être de la forme $\bot$ ou $\top$, soit s'écrire $l_1 \vee l_2 \vee \ldots \vee l_n$ avec l_i un littéral autre que $\bot$ ou $\top$.
- représentation d'une clause : un ensemble de littéraux tous distincts
  - La clause $\bot$ 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 $R(t_1, \ldots, t_n)$ apparaîsse au plus une fois sous forme positive ou négative ($\neg R(t_1, \ldots, t_n)$).

Clause simplifiée

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. $\square$

Clauses

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

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.

$$\begin{array}{llll} \texttt{clauseset}(s) = & \textbf{si } s = \emptyset \textbf{ alors } \bot & & \\ & \textbf{sinon} & \textbf{soit } (p, b) \in s, s' = s \backslash (p, b) & \\ & & \textbf{dans} & \textbf{soit } A = \textbf{si } b \textbf{ alors } p \textbf{ sinon } \neg p \\ & & & \textbf{dans} \quad \textbf{si } s' = \emptyset \textbf{ alors } A \\ & & & \qquad\quad\;\; \textbf{sinon } A \vee \texttt{clauseset}(s') \end{array}$$

Nombre de clauses différentes

- 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 ?

Nombre de clauses différentes

- 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 l 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 l 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 ρ un environnement.

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

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 $l \in L_1$ tel que $l \notin L_2$.*
  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 $l \in L_1$ peut apparaître dans L_2 mais avec un signe opposé (équivalent à $\neg l$). On a que $J \not\models \neg l$ donc $J \models l$. Si l'instance de prédicat sous-jacente à l n'apparait pas dans L_2 alors on peut étendre J pour que le littéral l soit vrai. Comme $l \in L_1$, 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 $\bot$
- 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

Exemple (Forme normale conjonctive)

- *Les formules* $\bot$, $\top$, $p \vee \neg q$, $p \wedge \neg q$, $(p \vee q) \wedge \neg q$ *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.*

Mise 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.

Fonctions booléennes

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 $\mathbb{B}^n \to \mathbb{B}$ notée $[P]_{\mathbb{B}}$.
Elle est définie par $[P]_{\mathbb{B}}(b_1, \dots, b_n) = \mathtt{val}(I_{\{b_1,\dots,b_n\}}, P)$ 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 $[P]_{\mathbb{B}}(b_1, \dots, b_n) =$ pour tout $b_1 \dots b_n \in \mathbb{B}$
- P est insatisfiable ssi $[P]_{\mathbb{B}}(b_1, \dots, b_n) =$ pour tout $b_1 \dots b_n \in \mathbb{B}$
- $P \equiv Q$ ssi $[P]_{\mathbb{B}}$? $[Q]_{\mathbb{B}}$

Fonctions booléennes

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 $\mathbb{B}^n \to \mathbb{B}$ notée $[P]_{\mathbb{B}}$.
Elle est définie par $[P]_{\mathbb{B}}(b_1, \dots, b_n) = \mathtt{val}(I_{\{b_1,\dots,b_n\}}, P)$ 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
—	—	V	V
V	—	F	F
F	V	F	F
F	F	F	V

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

Représentation d'une fonction booléenne

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

- soit $f \in \mathbb{B}^n \to \mathbb{B}$
- 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. $\vee, \neg$: $\neg(\neg(\neg p \vee \neg q) \vee \neg(p \vee q))$

Existence de la FNC

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 :* $Z \stackrel{\text{def}}{=} \{(b_1, \ldots, b_n) \mid f(b_1, \ldots, b_n) = F\}$
- *si cet ensemble est vide alors $f = [\top]_{\mathbb{B}}$*
- *formule en FNC sur $p_1, \ldots, p_n$ qui exclut les n-uplets de Z.*
    - *Pour $(b_1, \ldots, b_n) \in \mathbb{B}^n$: $l_1 \vee l_2 \vee \ldots \vee l_n$ 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 $(b_1, \ldots, b_n)$.*
    - *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!

La dernière fois : 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

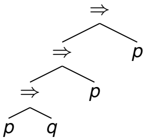
Toute formule est équivalente à une formule en forme normale de négation, c'est-à-dire qui ne comporte pas de connecteur $\neg$ sauf dans un littéral et n'utilise que les connecteurs $\top, \bot, \vee$ et $\wedge$ et les quantificateurs $\forall$ et $\exists$.

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

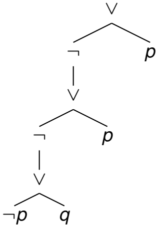
- $\mathtt{fnn}$: $\mathtt{fnn}(P)$ formule en forme normale de négation équivalente à P
- $\mathtt{neg}$: $\mathtt{neg}(P)$ formule en forme normale de négation équivalente à $\neg P$.

La dernière fois : Exemple

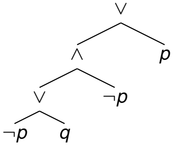
- $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$



- suppression de $\Rightarrow$:



- forme normale de négation :



La dernière fois : Forme normale conjonctive

Definition (Clause)

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

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

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 des FNC : Vrai ou Faux

Soient p, q, et r des variables propositionnelles

1. $p \wedge \neg q$ est une clause
2. la clause formée des littéraux p et $\neg p$ est toujours vraie
3. $p \vee \neg q \vee r$ est en forme normale conjonctive
4. $\neg p \vee q \wedge r$ est en forme normale conjonctive

La dernière fois : Mise 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.

Exercice :

Donner la forme normale conjonctive de la formule P dont la table de vérité est la suivante :

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

Méthode syntaxique

Construire la table de vérité d'une formule est trop complexe en général.
La formule $(a \wedge b) \vee c$ n'est pas en forme normale conjonctive car il y a une disjonction qui porte sur une formule qui n'est pas un littéral.
La règle de distributivité inverse les conjonctions et les disjonctions.

$$
\begin{aligned}
(a \wedge b) \vee c &\equiv (a \vee c) \wedge (b \vee c) \\
(a_1 \wedge \ldots \wedge a_n) \vee (b_1 \wedge \ldots \wedge b_p) &\equiv (a_1 \vee b_1) \wedge \ldots \wedge (a_1 \vee b_p) \\
&\quad \wedge \ldots \wedge (a_n \vee b_1) \wedge \ldots \wedge (a_n \vee b_p) \\
&= \bigwedge_{(i=1..n, j=1..p)} (a_i \vee b_j) \\
(a_1 \wedge b_1) \vee (a_2 \wedge b_2) \vee (a_3 \wedge b_3) &\equiv (a_1 \vee a_2 \vee a_3) \wedge (a_1 \vee a_2 \vee b_3) \\
&\quad \wedge (a_1 \vee b_2 \vee a_3) \wedge (a_1 \vee b_2 \vee b_3) \\
&\quad \wedge (b_1 \vee a_2 \vee a_3) \wedge (b_1 \vee a_2 \vee b_3) \\
&\quad \wedge (b_1 \vee b_2 \vee a_3) \wedge (b_1 \vee b_2 \vee b_3) \\
(a_1 \wedge b_1) \vee \ldots \vee (a_n \wedge b_n) &\equiv \bigwedge_{c_i \in \{a_i, b_i\}} c_1 \vee \ldots \vee c_n
\end{aligned}
$$

Algorithme de mise en FNC

- fonction $\texttt{fnc}(P)$: calcule un ensemble de clauses équivalent à P
- fonction auxiliaire $\texttt{fnc-neg}(P)$: construit un ensemble de clauses équivalent à $\neg P$.
- fonction utilitaire $\texttt{sh}$, prend en argument deux ensembles de clauses E et E' et construit un nouvel ensemble de clauses avec toutes les combinaisons possibles $C \vee C'$ avec $C \in E$ et $C' \in E'$,

$$\texttt{sh}(E, E') = \{C \vee C' | C \in E, C' \in E'\}$$

On note $\bigwedge(E)$ la conjonction des clauses de E. On a par distributivité :

$$\bigwedge(\texttt{sh}(E, E')) \equiv (\bigwedge(E)) \vee (\bigwedge(E'))$$

Si E contient n clauses et si E' contient p clauses alors $\texttt{sh}(E, E')$ peut contenir jusqu'à $n \times p$ clauses (moins si certaines clauses se simplifient).

Fonction associée

p instance de prédicat

$$
\begin{aligned}
\texttt{fnc}(\top) &= \\
\texttt{fnc}(\bot) &= \\
\texttt{fnc}(p) &= \\
\texttt{fnc}(\neg P) &= \\
\texttt{fnc}(P \wedge Q) &= \\
\texttt{fnc}(P \vee Q) &= \\
\texttt{fnc}(P \Rightarrow Q) &=
\end{aligned}
\qquad
\begin{aligned}
\texttt{fnc-neg}(\bot) &= \\
\texttt{fnc-neg}(\top) &= \\
\texttt{fnc-neg}(p) &= \\
\texttt{fnc-neg}(\neg P) &= \\
\texttt{fnc-neg}(P \vee Q) &= \\
\texttt{fnc-neg}(P \wedge Q) &= \\
\texttt{fnc-neg}(P \Rightarrow Q) &=
\end{aligned}
$$

$$
\begin{aligned}
\texttt{fnc}(\top) &= \emptyset \\
\texttt{fnc}(\bot) &= \{\bot\} \\
\texttt{fnc}(p) &= \{p\} \\
\texttt{fnc}(\neg P) &= \texttt{fnc-neg}(P) \\
\texttt{fnc}(P \land Q) &= \texttt{fnc}(P) \cup \texttt{fnc}(Q) \\
\texttt{fnc}(P \lor Q) &= \texttt{sh}(\texttt{fnc}(P), \texttt{fnc}(Q)) \\
\texttt{fnc}(P \Rightarrow Q) &= \texttt{sh}(\texttt{fnc-neg}(P), \texttt{fnc}(Q)) \\
\texttt{fnc-neg}(\bot) &= \emptyset \\
\texttt{fnc-neg}(\top) &= \{\bot\} \\
\texttt{fnc-neg}(p) &= \{\neg p\} \\
\texttt{fnc-neg}(\neg P) &= \texttt{fnc}(P) \\
\texttt{fnc-neg}(P \lor Q) &= \texttt{fnc-neg}(P) \cup \texttt{fnc-neg}(Q) \\
\texttt{fnc-neg}(P \land Q) &= \texttt{sh}(\texttt{fnc-neg}(P), \texttt{fnc-neg}(Q)) \\
\texttt{fnc-neg}(P \Rightarrow Q) &= \texttt{fnc}(P) \cup \texttt{fnc-neg}(Q)
\end{aligned}
$$

Forme normale disjonctive

On définit de manière duale la notion de forme normale disjonctive.

Definition (Forme normale disjonctive)

- Une conjonction élémentaire est une formule uniquement composée de littéraux et de conjonctions.
- Une formule P est en forme normale disjonctive (abrégé en FND, DNF en anglais), si elle s'écrit comme une disjonction de conjonctions élémentaires.

Comme pour les clauses, les conjonctions élémentaires se simplifient pour ne garder au plus qu'une occurrence d'une instance de prédicat : si p et $\neg p$ apparaissent dans la même conjonction alors celle-ci est équivalente à $\perp$. Comme $\perp \vee Q \equiv Q$, on peut simplement supprimer la conjonction.

Exemple : Forme normale disjonctive

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

Existence de la FND

Proposition

Pour toute formule propositionnelle P, il existe une formule Q équivalente en forme normale disjonctive.

Preuve: *On remarque que si P est en forme normale conjonctive, alors la forme normale de négation de $\neg P$ est en forme normale disjonctive.*
Il suffit donc de prendre la forme normale de négation de la négation de la forme normale conjonctive de la formule $\neg P$ pour obtenir une formule en forme normale disjonctive équivalente à P.

$$\mathtt{fnd}(P) = \mathtt{fnn}(\neg \mathtt{fnc}(\neg P))$$

$\square$

FND à partir de la table de vérité

- Comme précédemment, on peut aussi construire directement la FND à partir de la table de vérité de la formule P.
- variables propositionnelles : $\{p_1, \dots, p_n\}$.
- ensemble O des n-uplets $(b_1, \dots, b_n)$, pour lesquels la table de vérité donne la valeur vrai.
- Pour chaque ligne on construit une conjonction élémentaire $l_1 \wedge l_2 \wedge \dots \wedge l_n$ avec $l_i = p_i$ si $b_i = V$ et $l_i = \neg p_i$ si $b_i = F$.
- Il suffit ensuite de prendre la disjonction de ces formules pour trouver une formule équivalente à P.

Exemple

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

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

- les lignes qui nous intéressent sont la deuxième et la troisième
- formule qui dit exactement que l'on est dans l'un de ces deux cas.
- décomposition par ligne
  - sur la ligne 2 : $p_1 \wedge \neg p_2$
  - sur la ligne 3 : $\neg p_1 \wedge p_2$.
- La forme normale disjonctive est : $(p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2)$.

Exercice

Donner les formes normales disjonctives 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

Applications

Exemple

Pour la formule $P \stackrel{\text{def}}{=} (p_1 \Rightarrow p_2) \Rightarrow (\neg p_1 \wedge p_2)$ *on obtient la forme normale disjonctive :*

$$(p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2)$$

- il est facile de trouver un modèle d'une formule en FND :
  - on suppose que les branches de la disjonction sont en forme "simplifiée" c'est-à-dire qu'une variable propositionnelle apparaît dans un seul littéral.
  - il suffit de choisir une des branches de la disjonction.
  - pour chaque variable propositionnelle x dans un littéral l on prend dans le modèle $x \mapsto V$ si $l = x$ et $x \mapsto F$ si $l = \neg x$.
- aucune composante en forme simplifiée : cas où la formule initiale est équivalente à $\perp$: il n'y a pas de modèle, la formule est insatisfiable.

Limitations

- La mise en forme normale n'est pas forcément la meilleure manière de trouver un modèle.
- cette forme normale peut être exponentiellement plus grosse que la formule initiale.
- exemple : formule $(a_1 \vee b_1) \wedge \ldots \wedge (a_n \vee b_n)$ qui a $2n$ variables et $2n - 1$ connecteurs.
  - la forme normale disjonctive s'écrit $\bigvee_{c_i \in \{a_i, b_i\}} (c_1 \wedge \ldots \wedge c_n)$ dans laquelle il y a 2^n disjonctions.

Complexité de la FNC

- Toute FNC qui contient une clause non triviale n'est pas valide
  - on choisit une clause simplifiée de la forme $l_1 \vee \ldots \vee l_n$ que l'on rend fausse en choisissant une interprétation dans laquelle chacun des l_i est faux (possible car ils portent sur des variables propositionnelles différentes).
  - si une des clauses de la forme normale conjonctive est fausse alors il en est de même de la formule.
- La FNC de la formule $(a_1 \wedge b_1) \vee \ldots \vee (a_n \wedge b_n)$ s'écrit $\bigwedge_{c_i \in \{a_i, b_i\}} c_1 \vee \ldots \vee c_n$ qui contient 2^n clauses.
- Cette complexité est intrinsèque car liée à la complexité de décider si une formule propositionnelle est valide
- Si on cherche juste une formule equi-satisfiable (la formule A est satisfiable ssi B l'est) alors on peut trouver des transformations qui restent linéaires.

FNC efficace pour la satisfiabilité

- On a construit une FNC *équivalente* à la formule initiale. Pour la démonstration automatique, cette équivalence n'est pas forcément nécessaire.
- Pour le problème de satisfiabilité, on peut transformer la formule P en une formule Q seulement equisatisfiable. C'est-à-dire que P est satisfiable si et seulement si Q est satisfiable.
- Cela nous permet d'introduire de nouvelles variables propositionnelles dans Q pour éviter l'explosion combinatoire.
- P et Q équivalents : pour toute interprétation I, on a $I \models P$ ssi $I \models Q$
- P et Q equisatisfiables : (il existe une interprétation I telle que $I \models P$) ssi (il existe une interprétation J telle que $J \models Q$).
- Deux formules équivalentes sont equisatisfiables, le contraire n'est pas vrai.

Forme clausale equisatisfiable

Proposition (Forme clausale equisatisfiable)

Pour toute formule propositionnelle P, il existe un ensemble de clauses $\mathtt{clause}(P)$ dont la taille est linéaire par rapport à la taille de P et qui est equisatisfiable. Plus récisément on établit que pour toute interprétation I :

- *si $I \models \mathtt{clause}(P)$ alors $I \models P$ ($\mathtt{clause}(P) \models P$).*
- *si $I \models P$ alors il existe une interprétation J qui coincide avec I sur les symboles de P et telle que $J \models \mathtt{clause}(P)$.*

Algorithme

Construction récursive sur la structure de la formule en fnn (conjonctions, disjonctions et littéraux).

- Si $P = P_1 \wedge P_2$ alors on décompose P_1 et P_2 en clauses et on les rassemble : $\texttt{clause}(P) = \texttt{clause}(P_1) \cup \texttt{clause}(P_2)$
- Si P est un littéral ou une disjonction de littéraux alors P est déjà une clause et $\texttt{clause}(P) = \{P\}$
- Si $P = P_1 \vee P_2$ mais que P n'est pas une disjonction de littéraux, cela signifie que P contient une conjonction. En réarrangeant les disjonctions, la formule P est logiquement équivalente à $l_1 \vee \ldots \vee l_n \vee (R_1 \wedge Q_1) \vee \ldots \vee (R_p \wedge Q_p)$ (on met à plat les disjonctions sur des littéraux l_i puis celles qui ont une conjonction à éliminer).
  - On remplace chaque conjonction $R_i \wedge Q_i$ par une nouvelle variable propositionnelle x_i
    On ajoute des formules qui disent que $x_i \Rightarrow (R_i \wedge Q_i)$.

$$
\begin{aligned}
\texttt{clause}(P) = \; & \{l_1 \vee \ldots \vee l_n \vee x_1 \vee \ldots \vee x_p\} \\
& \cup \texttt{clause}(\neg x_1 \vee R_1) \cup \texttt{clause}(\neg x_1 \vee Q_1) \cup \ldots \\
& \cup \texttt{clause}(\neg x_p \vee R_p) \cup \texttt{clause}(\neg x_p \vee Q_p)
\end{aligned}
$$

Exemple

$P \overset{\text{def}}{=} (a_1 \wedge b_1) \vee (a_2 \wedge \neg b_2) \vee (\neg a_3 \wedge (b_3 \vee c)) \vee a_4 \vee \neg b_4$.

- On introduit x_1, x_2 et x_3 pour représenter respectivement $(a_1 \wedge b_1)$ (x_1), $(a_2 \wedge \neg b_2)$ (x_2), et $(\neg a_3 \wedge (b_3 \vee c))$ pour (x_3).
- $\texttt{clause}(P) = \{x_1 \vee x_2 \vee x_3 \vee a_4 \vee \neg b_4, \neg x_1 \vee a_1, \neg x_1 \vee b_1, \neg x_2 \vee a_2, \neg x_2 \vee \neg b_2, \neg x_3 \vee a_3, \neg x_3 \vee b_3 \vee c\}$
- même nombre de connecteurs $\vee$ et $\wedge$ que dans la formule initiale plus un nombre de variables et de clauses égal au nombre de connecteurs $\wedge$ qui se trouvent sous un connecteur $\vee$ dans la formule initiale (ici 3).

1. Formes normales
   - Forme normale de négation
   - Formes normales conjonctive et disjonctive
   - Skolémisation
2. Représentations alternatives des formules
3. Systèmes de déduction

Traiter les quantificateurs

- Les transformations précédentes traitent de la partie propositionnelle des formules (sans quantificateur)
- Les propriétés de l'équivalence permettent de ramener les quantificateurs au début de la formule (forme prénexe)

$$\forall x_1 \dots x_n, \exists y_1 \dots y_p, \forall \dots \exists \dots, A$$

- Les alternances $\forall$ et $\exists$ sont essentielles
- Si on ne s'intéresse qu'à la satisfiabilité, les quantificateurs existentiels peuvent être remplacés par de nouveaux symboles de fonction

Forme prénexe

Si x n'est pas libre dans la formule A alors :

$$\begin{array}{ll} \forall x, (A \wedge G(x)) \equiv A \wedge \forall x, G(x) & \forall x, (A \vee G(x)) \equiv A \vee \forall x, G(x) \\ \exists x, (A \wedge G(x)) \equiv A \wedge \exists x, G(x) & \exists x, (A \vee G(x)) \equiv A \vee \exists x, G(x) \end{array}$$

la remontée des quantificateurs au travers d'une négation change le quantificateur (loi de de Morgan) de même lorsqu'il est dans la partie gauche d'une implication.
La remontée des quantificateurs se fait *après* avoir mis la formule en forme normale de négation.

la remontée naïve des quantificateurs peut engendrer une *capture* des variables : condition que x n'est pas libre dans la formule A.
Si x est libre dans A (en plus d'être liée dans $\exists x, G(x)$) alors on renomme la variable liée en choisissant une variable z qui n'est pas libre dans A ni dans $\exists x, G(x)$, on a alors $A \vee \exists x, G(x) = A \vee \exists z, G(z) \equiv \exists z, A \vee G(z)$

Forme prénexe

Proposition (Forme prénexe)

Toute formule logique P est équivalente à une formule de la forme

$$\mathcal{Q}x_1, \dots, \mathcal{Q}x_n, A$$

avec $\mathcal{Q}$ l'un des quantificateurs $\forall$ ou $\exists$ et A une formule propositionnelle (sans quantificateur).
Une formule de la forme $\mathcal{Q}x_1, \dots, \mathcal{Q}x_n, A$ est dite en ***forme prénexe****.*

Preuve: *On met la formule P en fnn, on utilise les équivalences rappelées ci-dessus pour remonter les quantificateurs en renommant si nécessaire les variables liées.* $\square$

La forme prénexe n'est pas unique. L'ordre des quantificateurs varie en fonction de l'ordre dans lequel on choisit de les faire remonter.

Elimination des quantificateurs existentiels

- intervertir un quantificateur existentiel et un quantificateur universel ne préserve pas l'équivalence entre les formules.
- si on ne s'intéresse qu'à la satisfiabilité, alors on peut se ramener, grace à une transformation appelée la skolemisation, à une formule prénexe qui n'aura que des quantificateurs universels.
- la skolemisation a été introduite par le logicien norvégien Thoraf Albert Skolem au 20ème siècle
- la skolemisation remplace l'usage de quantifications existentielles par l'introduction de nouveaux symboles dans la signature
- la satisfiabilité est préservée

Skolemisation : intuition

- Soit la formule $\forall x, \exists y, P(x, y)$
- cette formule est satisfaite si pour tout élément d du domaine, on peut trouver un élément e du domaine tel que $I, \{x \mapsto d, y \mapsto e\} \models P(x, y)$.
- la valeur e pour y dépend de la valeur d pour x.
- e est donc une fonction de d
- le problème de trouver un modèle de $\forall x, \exists y, P(x, y)$ est analogue au problème de trouver un modèle pour $\forall x, P(x, f(x))$ dans lequel f est un nouveau symbole de fonction.
- $\forall x, P(x, f(x)) \models \forall x, \exists y, P(x, y)$ (choisir pour y la valeur de $f(x)$)
- à partir d'un modèle de $\forall x, \exists y, P(x, y)$, on peut construire un modèle de la formule $\forall x, P(x, f(x))$ en choisissant la “bonne” interprétation pour la fonction f.

Elimination d'un quantificateur existentiel

Definition (Elimination d'un quantificateur existentiel)

- Soit une formule A en fnn dans laquelle apparaît une sous-formule $\exists y, B$.
- On identifie les variables libres de $\exists y, B$: $\{x_1, \dots, x_n\}$.
- On introduit un *nouveau symbole de fonction* f à n arguments (si $n = 0$ alors f est une constante).
- On introduit la formule A' qui est la formule A dans laquelle on a remplacé la sous-formule $\exists y, B$ par $B[y \leftarrow f(x_1, \dots, x_n)]$.

Exemple

On suppose que l'on a un symbole de prédicat binaire P. On élimine le quantificateur existentiel dans les formules suivantes :

- $\forall x, \exists y, P(x, y)$:

Exemple

On suppose que l'on a un symbole de prédicat binaire P. On élimine le quantificateur existentiel dans les formules suivantes :

- $\forall x, \exists y, P(x, y)$: la sous-formule existentielle à traiter est $\exists y, P(x, y)$. Elle a une seule variable libre x : nouveau symbole de fonction f d'arité 1. On remplace y par $f(x)$. La formule devient $\forall x, P(x, f(x))$
- $\forall x\, y, \exists z, P(x, z) \wedge P(z, y)$:

Exemple

On suppose que l'on a un symbole de prédicat binaire P. On élimine le quantificateur existentiel dans les formules suivantes :

- $\forall x, \exists y, P(x, y)$: la sous-formule existentielle à traiter est $\exists y, P(x, y)$. Elle a une seule variable libre x : nouveau symbole de fonction f d'arité 1. On remplace y par $f(x)$. La formule devient $\forall x, P(x, f(x))$
- $\forall x\, y, \exists z, P(x, z) \wedge P(z, y)$: la sous-formule existentielle à traiter est $\exists z, P(x, z) \wedge P(z, y)$. Elle a deux variables libres x, y : nouveau symbole de fonction g d'arité 2. On remplace z par $g(x, y)$. La formule devient $\forall x\, y, P(x, g(x, y)) \wedge P(g(x, y), y)$
- $\exists x, \forall y, P(x, y)$:

Exemple

On suppose que l'on a un symbole de prédicat binaire P. On élimine le quantificateur existentiel dans les formules suivantes :

- $\forall x, \exists y, P(x, y)$: la sous-formule existentielle à traiter est $\exists y, P(x, y)$. Elle a une seule variable libre x : nouveau symbole de fonction f d'arité 1. On remplace y par $f(x)$. La formule devient $\forall x, P(x, f(x))$
- $\forall x\, y, \exists z, P(x, z) \wedge P(z, y)$: la sous-formule existentielle à traiter est $\exists z, P(x, z) \wedge P(z, y)$. Elle a deux variables libres x, y : nouveau symbole de fonction g d'arité 2. On remplace z par $g(x, y)$. La formule devient $\forall x\, y, P(x, g(x, y)) \wedge P(g(x, y), y)$
- $\exists x, \forall y, P(x, y)$: la sous-formule existentielle à traiter est la formule elle même. Elle n'a pas de variable libre : nouveau symbole de constante a. On remplace x par a. La formule devient $\forall y, P(a, y)$

Méthode alternative

Dans beaucoup d'ouvrages, l'élimination du quantificateur existentiel se fait après avoir mis la formule en forme prénexe. On choisit alors l'arité du symbole de fonction introduit en fonction du nombre de quantifications universelles préalables. L'idée étant que si on a une alternance $\forall x, \exists y$ alors le choix de y peut dépendre de x.

Dans l'approche présentée ici, on s'intéresse à la sous-formule $\exists y, A$ et aux dépendances de la condition que doit vérifier y. Si A ne dépend pas de x alors dès qu'on a une valeur pour y qui vérifie A, cette valeur fonctionnera pour toute valeur de x, il n'est donc pas utile de faire dépendre y de la valeur de x.

Exemple

La méthode prénexe est correcte mais peut amener à des dépendances artificielles.
$(\forall x, P(x)) \vee \exists y, \neg P(y)$,

- la formule $\exists y, \neg P(y)$ étant close, le symbole introduit pour remplacer l'existentielle est une constante a et on obtient la formule $(\forall x, P(x)) \vee \neg P(a)$. Le domaine de Herbrand associé est constitué de la seule constante a.
- Si on utilise la forme prénexe $\forall x, \exists y, P(x) \vee \neg P(y)$, par la méthode prénexe, il faut introduire un symbole de fonction unaire f et la formule devient $(\forall x, P(x) \vee \neg P(f(x)))$, le domaine de Herbrand associé est alors infini.

Avoir un domaine de Herbrand plus restreint simplifie la recherche de modèle.

Propriétés

Proposition

Soit A une formule en forme normale de négation. Soit A' obtenue à partir de A en éliminant un quantificateur existentiel suivant la procédure précédente.

- $A' \models A$
- *tout modèle de A peut être transformé en un modèle de A' qui coincide avec celui de A sauf pour l'interprétation du nouveau symbole f.*

Preuve :

- La première propriété est une conséquence de $B[y \leftarrow f(x_1, \dots, x_n)] \models \exists y, B$ et du fait que A est en fnn.
  On utilise la stabilité de $\models$: si $Q \models R$ alors
  $Q \wedge P \models R \wedge P \quad Q \vee P \models R \vee P \quad \forall x, Q \models \forall x, R \quad \exists x, Q \models \exists x, R$

Proposition

Soit A une formule en forme normale de négation. Soit A' obtenue à partir de A en éliminant un quantificateur existentiel suivant la procédure précédente.

- $A' \models A$
- *tout modèle de A peut être transformé en un modèle de A' qui coincide avec celui de A sauf pour l'interprétation du nouveau symbole f.*

- Pour la seconde propriété, soit une interprétation I de domaine $\mathcal{D}$ telle que $I \models A$. B a comme variables libres $x_1 \dots x_n, y$.
  - Soit l'ensemble $\{(d_1, \dots, d_n, e) | I, \{x_1 \mapsto d_1, \dots, x_n \mapsto d_n, y \mapsto e\} \models B\}$.
  - On construit une fonction $f_J \in \mathcal{D}^n \to \mathcal{D}$ telle que $I, \{x_1 \mapsto d_1, \dots, x_n \mapsto d_n, y \mapsto f_J(d_1, \dots, d_n)\} \models B$ s'il existe e tel que $I, \{x_1 \mapsto d_1, \dots, x_n \mapsto d_n, y \mapsto e\} \models B$ et $f_J(d_1, \dots, d_n) = d$ avec d une valeur arbitraire de $\mathcal{D}$ si pour tout e, $I, \{x_1 \mapsto d_1, \dots, x_n \mapsto d_n, y \mapsto e\} \not\models B$.
  - On construit J sur le domaine $\mathcal{D}$, qui étend I en interprétant f par f_J.
  - $J, \rho \models (\exists y, B) \Leftrightarrow B[y \leftarrow f(x_1, \dots, x_n)]$ pour tout environnement ρ
  - donc $J \models \forall x_1 \dots x_n, ((\exists y, B) \Leftrightarrow B[y \leftarrow f(x_1, \dots, x_n)])$.
  - On a aussi $J \models A$ car J étend I et $I \models A$.
  - on montre $\forall x_1 \dots x_n, ((\exists y, B) \Leftrightarrow B[y \leftarrow f(x_1, \dots, x_n)]) \models A \Leftrightarrow A'$ car A' est obtenue à partir de A en remplaçant $\exists y, B$ par $B[y \leftarrow f(x_1, \dots, x_n)]$
  - on en déduit que J est un modèle de A'.

Skolémisation d'une formule

- On itère l'opération précédente pour éliminer tous les quantificateurs existentiels d'une formule.
- Chaque élimination de quantificateur introduit un *nouveau symbole* de fonction.

Si on applique ces transformations à une formule A en forme normale de négation, on obtient une formule B sans quantificateur existentiel telle que

- $B \models A$
- Tout modèle de A peut être étendu en un modèle de B par le choix d'une interprétation des symboles de B qui ne sont pas dans A.

Suppression des existentielles

Proposition

Soit A une formule en forme normale de négation et B une formule obtenue par élimination des quantificateurs existentiels de A.

- *Si B est valide alors A l'est aussi.*
- *Si B est satisfiable alors A l'est aussi.*
- *Si B est insatisfiable alors A l'est aussi.*

Preuve: *Les deux premières propriétés sont des conséquences directes de $B \models A$ alors que la dernière vient du fait que s'il existait un modèle de A alors il y en aurait aussi un de B.* □

Un symbole nouveau par quantificateur

Exemple

Soit la formule $(\exists x, P(x)) \land (\exists z, \neg P(z))$. *La skolemisation (utilisée deux fois) introduit deux nouvelles constantes* a *et* b *et donne la formule* $P(a) \land \neg P(b)$. *Cette formule est satisfiable, par contre si on avait utilisé deux fois la même constante* a *on aboutirait à la formule* $P(a) \land \neg P(a)$ *qui est insatisfiable.*

Skolémisation d'une formule

Proposition (Skolemisation, Forme de Skolem)

Soit A une formule close alors il existe une formule B en forme normale de négation et sans quantificateur telle que si $vl(B) = \{x_1, \ldots, x_n\}$ on a

- $(\forall x_1 \ldots x_n, B) \models A$
- *si A a un modèle alors il en est de même de $\forall x_1 \ldots x_n, B$*

L'opération de transformation s'appelle la skolémisation. On dit que la formule $\forall x_1 \ldots x_n, B$ est la forme de Skolem de A ou encore la formule skolémisée associée à A.

Preuve: *Il suffit de mettre la formule en forme normale de négation. On applique ensuite l'élimination des quantificateurs existentiels suivant la procédure vue précédemment. On fait ensuite remonter les quantificateurs universels (en s'assurant au préalable qu'il n'y en n'a pas deux différents qui portent sur une variable de même nom) et en utilisant les équivalences $A \circ \forall x, B \equiv \forall x, (A \circ B)$ lorsque x n'est pas libre dans A et $\circ \in \{\vee, \wedge\}$.* □

Applications

La skolemisation préserve la satisfiabilité mais fournit également une méthode pour montrer la validité d'une formule en recherchant des modèles de Herbrand d'un ensemble de formules propositionnelles. Il suffit pour cela de considérer la négation de la formule initiale.

- A est valide ssi $\neg A$ est insatisfiable ssi la forme skolémisée de $\neg A$ est insatisfiable.
- Comme la forme skolémisée est une formule universelle, le théorème de Herbrand ramène la satisfiabilité à l'existence d'un modèle de Herbrand.

Exemple

Soit la formule $A \stackrel{\text{def}}{=} (\forall x, (P(x) \Rightarrow Q(x))) \Rightarrow (\forall x, P(x)) \Rightarrow \forall x, Q(x)$.
Pour montrer que A est valide, on montre que $\neg A$ est insatisfiable.
Pour cela, on skolémise $\neg A$ en passant par les étapes suivantes :

1. Forme normale de négation de $\neg A$:
   $(\forall x, (\neg P(x) \vee Q(x))) \wedge (\forall y, P(y)) \wedge \exists z, \neg Q(z)$
2. Elimination du quantificateur existentiel :
   $(\forall x, (\neg P(x) \vee Q(x))) \wedge (\forall y, P(y)) \wedge \neg Q(a)$
3. Mise en forme prénexe : $\forall x\, y, ((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a))$
4. Domaine de Herbrand : $\mathcal{D} = \{a\}$
5. Instances closes : $\{(\neg P(a) \vee Q(a)) \wedge P(a) \wedge \neg Q(a)\}$ insatisfiable

D'après le théorème de Herbrand, la formule
$\forall x\, y, ((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a))$ est insatisfiable et donc il en est de même de la formule $\neg A$ donc A est valide.

Fermeture universelle

Definition (Fermeture universelle)

Soit P une formule dont les variables libres sont $\{x_1, \ldots, x_n\}$. On appelle fermeture universelle de la formule P et on notera $\forall(P)$ la formule $\forall x_1, \ldots, x_n, P$.

Cette formule est close (sans variable libre).

L'opération de permutation de deux quantificateurs universels ne change pas le sens de la formule donc l'ordre dans lequel les variables apparaissent dans les quantificateurs n'a pas d'importance.

Skolemisation d'un ensemble de formules

Si on se donne un ensemble de formules $A_1, \ldots, A_n$ on peut traiter la skolémisation de manière indépendante pour chaque formule (en introduisant évidemment des symboles différents).
On obtient pour chaque A_i une formule B_i sans quantificateurs telle que $\forall(B_i) \models A_i$ et tout modèle de A_i s'étend en un modèle de $\forall(B_i)$.
On en déduit aisément que :

- $\forall(B_1), \ldots, \forall(B_n) \models \{A_1, \ldots, A_n\}$ ($\forall(B_1), \ldots, \forall(B_n) \models A_1 \wedge \ldots \wedge A_n$)
- S'il existe un modèle des formules $A_1, \ldots, A_n$ alors on a aussi un modèle de l'ensemble de formules $\{\forall(B_1), \ldots, \forall(B_n)\}$.

Forme clausale avec quantificateurs

Definition (Forme clausale d'une formule avec quantificateur)

La forme clausale d'une formule P est un ensemble de clauses $\mathcal{C} = \{C_1, \ldots, C_p\}$ tel que

- $\forall(C_1), \ldots, \forall(C_p) \models P$
- S'il existe un modèle de P alors on peut l'étendre en un modèle de l'ensemble de formules $\{\forall(C_1), \ldots, \forall(C_p)\}$.

La forme de skolem de P s'écrit $\forall x_1 \ \ldots \ x_n, A$ avec A formule propositionnelle. L'ensemble de clauses $\mathcal{C} = \{C_1, \ldots, C_p\}$ est obtenu en mettant en FNC la formule A. On a alors

- $C_1, \ldots, C_p \models A$
- tout modèle de A s'étend en un modèle de $\{C_1, \ldots, C_p\}$.

La quantification universelle se comporte de manière *régulière* par rapport à la conjonction (mais pas la disjonction).
On a $\forall x, P(x) \wedge Q(x) \equiv (\forall x, P(x)) \wedge (\forall x, Q(x))$, et donc
$\forall(C_1 \wedge \ldots \wedge C_p) \equiv \forall(C_1) \wedge \ldots \wedge \forall(C_p)$

A savoir

- Mise en forme normale de négation
- Mise en forme clausale syntaxique des formules propositionnelles
- Elimination des quantificateurs existentiels
- Construction de la forme clausale des formules avec quantificateurs
- Distinguer les étapes d'équivalence et d'équisatisfiabilité
- Définition d'une transformation par des équations récursives sur la structure des formules


That's all Folks!

La dernière fois

- Mise en forme normale de négation
- Mise en forme normale de conjonction
- Mise en forme clausale syntaxique des formules propositionnelles
- Elimination des quantificateurs existentiels
- Construction de la forme clausale des formules avec quantificateurs
- Distinguer les étapes d'équivalence et d'équisatisfiabilité

La dernière fois : Forme clausale equisatisfiable

Proposition (Forme clausale equisatisfiable)

Pour toute formule propositionnelle P, il existe un ensemble de clauses $\mathtt{clause}(P)$ dont la taille est linéaire par rapport à la taille de P et qui est equisatisfiable. Plus récisément on établit que pour toute interprétation I :

- *si $I \models \mathtt{clause}(P)$ alors $I \models P$ ($\mathtt{clause}(P) \models P$).*
- *si $I \models P$ alors il existe une interprétation J qui coincide avec I sur les symboles de P et telle que $J \models \mathtt{clause}(P)$.*

La dernière fois : FNC efficace pour la satisfiabilité

Construction récursive sur la structure de la formule en fnn (conjonctions, disjonctions et littéraux).

- Si $P = P_1 \wedge P_2$ alors on décompose P_1 et P_2 en clauses et on les rassemble : $\texttt{clause}(P) = \texttt{clause}(P_1) \cup \texttt{clause}(P_2)$
- Si P est un littéral ou une disjonction de littéraux alors P est déjà une clause et $\texttt{clause}(P) = \{P\}$
- Si $P = P_1 \vee P_2$ mais que P n'est pas une disjonction de littéraux, cela signifie que P contient une conjonction. En réarrangeant les disjonctions, la formule P est logiquement équivalente à $l_1 \vee \ldots \vee l_n \vee (R_1 \wedge Q_1) \vee \ldots \vee (R_p \wedge Q_p)$ (on met à plat les disjonctions sur des littéraux l_i puis celles qui ont une conjonction à éliminer).
  - On remplace chaque conjonction $R_i \wedge Q_i$ par une nouvelle variable propositionnelle x_i
    On ajoute des formules qui disent que $x_i \Rightarrow (R_i \wedge Q_i)$.

$$\begin{aligned}\texttt{clause}(P) = \quad & \{l_1 \vee \ldots \vee l_n \vee x_1 \vee \ldots \vee x_p\} \\ & \cup \texttt{clause}(\neg x_1 \vee R_1) \cup \texttt{clause}(\neg x_1 \vee Q_1) \cup \ldots \\ & \cup \texttt{clause}(\neg x_p \vee R_p) \cup \texttt{clause}(\neg x_p \vee Q_p)\end{aligned}$$

La dernière fois : Skolemisation

Proposition

Soit A une formule en forme normale de négation. Soit A' obtenue à partir de A en éliminant un quantificateur existentiel suivant la procédure précédente.

- $A' \models A$
- *tout modèle de A peut être transformé en un modèle de A' qui coincide avec celui de A sauf pour l'interprétation du nouveau symbole f.*

Definition (Elimination d'un quantificateur existentiel)

- Soit une formule A en fnn dans laquelle apparaît une sous-formule $\exists y, B$.
- On identifie les variables libres de $\exists y, B$: $\{x_1, \dots, x_n\}$.
- On introduit un *nouveau symbole de fonction* f à n arguments (si $n = 0$ alors f est une constante).
- On introduit la formule A' qui est la formule A dans laquelle on a remplacé la sous-formule $\exists y, B$ par $B[y \leftarrow f(x_1, \dots, x_n)]$.

Exercice

On part de la formule $A \stackrel{\text{def}}{=} (\exists x, \forall y, R(x,y)) \Rightarrow (\forall y, \exists x, R(x,y))$*.*

- *Donner la forme normale de négation de la formule* $\neg A$
- *Donner la forme de skolem de la formule* $\neg A$
- *Donner la forme clausale de* $\neg A$
- *la formule* $\neg A$ *est-elle satisfiable ? La formule* A *est-elle valide ?*

$A \stackrel{\text{def}}{=} (\exists x, \forall y, R(x,y)) \Rightarrow (\forall y, \exists x, R(x,y))$

- Forme normale de négation de $\neg A$: $(\exists x, \forall y, R(x,y)) \wedge (\exists y, \forall x, \neg R(x,y))$
- Skolemisation :
  - Deux constantes a, b
  - Elimination des existentielles : $(\forall y, R(a,y)) \wedge (\forall x, \neg R(x,b))$
  - Forme skolémisée : $\forall x\, y, R(a,y) \wedge \neg R(x,b)$
- Forme clausale : $\{R(a,y), \neg R(x,b)\}$
- Satisfiabilité, validité :
  - $\neg A$ est insatisfiable $R(a,b)$ doit être à la fois vrai et faux
  - A est valide

3–Manipuler les formules

Introduction

- alternatives aux connecteurs usuels pour représenter les formules de la logique
- représentation à l'aide de diagrammes de décision binaires
- déjà fait en TD dans certains groupes, plus en détail – ici on voit les grandes lignes

Restrictions sur les connecteurs

- Connecteurs usuels : $\forall, \exists, \wedge, \vee, \neg, \Rightarrow, \bot, \top$
- Connecteurs remplaçables
  - $\bot \equiv p \wedge \neg p$, $\top \equiv p \vee \neg p \equiv p \Rightarrow p$, avec p variable propositionnelle.
  - $(A \Rightarrow B) \equiv (\neg A \vee B)$,
  - forme normale de négation (seules négations dans les littéraux)
  - Lois de de Morgan
    - $(A \vee B) \equiv \neg(\neg A \wedge \neg B)$
    - $(A \wedge B) \equiv \neg(\neg A \vee \neg B)$
    - $(\exists x, A) \equiv \neg(\forall x, \neg A)$
    - $(\forall x, A) \equiv \neg(\exists x, \neg A)$
  - *Logique minimale* avec $\forall, \Rightarrow$ et $\bot$
    - $\top \equiv \bot \Rightarrow \bot$
    - $\neg A \equiv A \Rightarrow \bot$
    - $(A \vee B) \equiv \neg A \Rightarrow B \equiv \neg B \Rightarrow A$,
    - $(A \wedge B) \equiv \neg(B \Rightarrow \neg A) \equiv \neg(A \Rightarrow \neg B)$
    - $(\exists x, A) \equiv \neg(\forall x, \neg A)$
- Formes normales conjonctives et disjonctives : $\forall, \exists, \wedge, \vee$ et des littéraux
  - taille *exponentielle* si on préserve l'équivalence
  - taille *linéaire* si on ne garde que l'équisatisfiabilité.

Restrictions sur les connecteurs : limitations

- on ne peut pas se débarrasser simultanément de la négation et de l'implication.
  - si une formule P n'utilise que les connecteurs $\vee$ et $\wedge$, alors la fonction booléenne correspondant $[P]_{\mathbb{B}}$ est croissante (en considérant l'ordre $F < V$).
  - on ne peut représenter avec juste $\vee$ et $\wedge$ des fonctions booléennes comme la négation qui ne sont pas croissantes.

  La mise en forme normale conjonctive montre que l'on peut représenter toutes les formules avec les deux quantificateurs, les connecteurs $\vee$ et $\wedge$ (dont on peut restreindre l'imbrication) et des négations limitées aux littéraux.

- La skolémisation élimine les quantifications existentielles mais la formule obtenue *n'est pas équivalente* à la formule initiale, elle peut être fausse dans des interprétations qui rendent vraies la formule initiale.

Connecteurs alternatifs

- autres connecteurs propositionnels binaires pour des représentations différentes des formules.
- ou exclusif noté $\oplus$:
  - $x \oplus y \equiv x \vee y \wedge \neg(x \wedge y) \equiv (x \wedge \neg y) \vee (\neg x \wedge y) \equiv \neg(x \Leftrightarrow y)$
  - coder les fonctions booléennes à l'aide des connecteurs $\oplus, \wedge, \top$.
  - $\oplus$ et $\wedge$ munissent $\mathbb{B}$ d'une structure d'*anneau de Boole*.
  - L'opération $\oplus$ sert en cryptographie : codage bit-à-bit d'un message. On utilise $x \oplus x = \bot$ et $x \oplus \bot = x$, et l'associativité de $\oplus$. si $\mathtt{key}$ clé secrète, le message m crypté sera $m \oplus \mathtt{key}$, il est décrypté par la même opération $(m \oplus \mathtt{key}) \oplus \mathtt{key}$
- coder les fonctions booléennes avec des variables propositionnelles et un unique connecteur binaire : la barre de Sheffer $x \mid y$ ($\mathtt{NAND}$)
  - $x \mid y \equiv \neg(x \wedge y)$,
  - on vérifie que $x \mid x \equiv \neg x$, $x \mid \neg x \equiv \top$, $(x \mid y) \mid (x \mid y) \equiv \neg(x \mid y) \equiv x \wedge y$
  - constructions suffisantes pour reconstruire toutes les formules propositionnelles.

3–Manipuler les formules

1. Formes normales

2. Représentations alternatives des formules
   - Restrictions sur les connecteurs
   - Connecteurs alternatifs
   - Diagrammes de décision binaire

3. Systèmes de déduction

Diagrammes de décision binaire

- représentation alternative de la partie propositionnelle des formules : connecteur de conditionnelle à trois arguments **IF**(P, Q, R)
- construction naturelle dans les langages de programmation
- en limitant les conditionnelles aux variables propositionnelles, la représentation des formules propositionnelles devient très efficace
- technique largement utilisée en démonstration automatique.

IF-expressions

- connecteur logique $\mathbf{IF}(P, Q, R)$: correspond à $(P \wedge Q) \vee (\neg P \wedge R)$, ou de manière équivalente à $(P \Rightarrow Q) \wedge (\neg P \Rightarrow R)$
- représentation des fonctions booléennes avec **IF**, $\bot$ et $\top$:

$$\begin{aligned}
\mathbf{IF}(P, \bot, \top) &\equiv \neg P \\
\mathbf{IF}(P, Q, \bot) &\equiv P \wedge Q \\
\mathbf{IF}(P, \top, Q) &\equiv P \vee Q \\
\mathbf{IF}(P, Q, \top) &\equiv P \Rightarrow Q
\end{aligned}$$

IF-expressions : forme simplifiée

- simplifier la partie condition pour ne porter que sur des variables propositionnelles
- équivalences :

$$\mathbf{IF}(\top, P, Q) \equiv P \qquad \mathbf{IF}(\bot, P, Q) \equiv Q$$
$$\mathbf{IF}(\mathbf{IF}(B, P, Q), R, S) \equiv \mathbf{IF}(B, \mathbf{IF}(P, R, S), \mathbf{IF}(Q, R, S))$$

- la forme “simplifiée” peut faire grossir la taille de la formule de manière exponentielle.

IF-expressions : complexité de la simplification

- variables propositionnelles $p, p_1, \dots, p_k, q_1, \dots, q_k$,
- suite de formules $A_0 = p$, $A_{n+1} = \mathbf{IF}(A_n, p_n, q_n)$.
- A_n contient n connecteurs **IF** et chaque variable propositionnelle apparaît exactement une fois.
- simplification de A_{n+1}, si p est une variable propositionnelle, on note $\tilde{p}$ la IF-expression $\mathbf{IF}(p, \top, \bot)$

$$
\begin{aligned}
A_{n+1} &= \mathbf{IF}(A_n, p_n, q_n) \quad = \mathbf{IF}(\mathbf{IF}(A_{n-1}, p_{n-1}, q_{n-1}), p_n, q_n) \\
&\equiv \mathbf{IF}(A_{n-1}, \mathbf{IF}(p_{n-1}, \tilde{p}_n, \tilde{q}_n), \mathbf{IF}(q_{n-1}, \tilde{p}_n, \tilde{q}_n)) \\
&\equiv \mathbf{IF}(A_{n-2}, \quad \mathbf{IF}(p_{n-2}, \mathbf{IF}(p_{n-1}, \tilde{p}_n, \tilde{q}_n), \mathbf{IF}(q_{n-1}, \tilde{p}_n, \tilde{q}_n)), \\
&\qquad\qquad\quad \mathbf{IF}(p_{n-2}, \mathbf{IF}(p_{n-1}, \tilde{p}_n, \tilde{q}_n), \mathbf{IF}(q_{n-1}, \tilde{p}_n, \tilde{q}_n)))
\end{aligned}
$$

- A chaque étape, on diminue de un le nombre de **IF** dans la condition mais on double le nombre de ceux qui apparaissent dans les branches et on en ajoute deux supplémentaires.

IF-expressions : enjeux

- limiter cette explosion par le choix d'une méthode de construction et d'une représentation adéquate.
- le comportement exponentiel dans le pire des cas est de toute manière inévitable lorsqu'on veut résoudre des problèmes généraux du calcul propositionnel.

Arbre de décision binaire

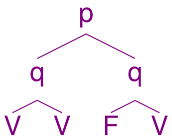
Formules du calcul propositionnel construites sur un ensemble de variables propositionnelles *totalement ordonné*.

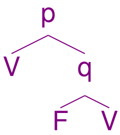
Definition (Arbre de décision binaire, BDT)

- Un arbre de décision binaire (BDT) est un arbre binaire dont les nœuds sont étiquetés par des variables propositionnelles et les feuilles sont formées des constantes booléennes V ou F.
- les variables propositionnelles des fils sont de plus strictement plus grandes que celle du père.
- L'arbre de décision binaire définit une fonction booléenne sur l'ensemble des variables.
  - soit un nœud étiqueté par la variable x, le sous arbre gauche correspond au cas où $x \mapsto V$ et le sous-arbre droit au cas où $x \mapsto F$.
  - Une branche de l'arbre définit une interprétation des variables propositionnelles et la valeur de la fonction est déterminée par le booléen présent à la feuille.
  - toutes les variables ne sont pas forcément présentes dans une branche : valeur pour un ensemble d'interprétations.

Exemple

Si on a deux variables propositionnelles $p < q$, on peut construire les arbres de décision binaire t et t' suivant auquels sont associés la même fonction booléenne dont on donne le tableau :





p	q	t, t'
V	V	V
V	F	V
F	V	F
F	F	V

Valeur

On définit simplement de manière récursive la valeur $\texttt{vald}(I, t)$ d'un arbre de décision binaire t dans une interprétation des variabbles propositionnelles I :

$$\begin{aligned}
\texttt{vald}(I, b) &= b \ \text{ si } b \in \{V, F\}\\
\texttt{vald}(I, \mathbf{IF}(x, P, Q)) &= \texttt{vald}(I, P) \ \text{ si } I(x) = V\\
\texttt{vald}(I, \mathbf{IF}(x, P, Q)) &= \texttt{vald}(I, Q) \ \text{ si } I(x) = F
\end{aligned}$$

Comme pour les formules ordinaires, pour une interprétation I et un arbre t, on notera $I \models t$ la propriété $\texttt{vald}(I, t) = V$.

Diagramme de décision binaire : motivations

Inconvénients des arbres de décision binaires :

- arbres différents pour des formules équivalentes
  (par exemple $\mathbf{IF}(x, P, P) \equiv P$)
- taille parfois exponentielle en le nombre de variables propositionnelles.

Diagramme de décision binaire : définitions

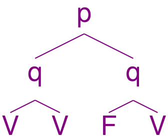
Definition (Diagramme de décision binaire, BDD, OBDD)

Un diagramme de décision binaire (BDD) est obtenu à partir d'un arbre de décision en assurant de plus les conditions suivantes :

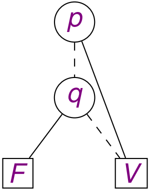
1. *réduction :* aucun nœud n'a deux fils identiques, cela correspond au fait que l'expression $\mathbf{IF}(p, A, A)$ est logiquement équivalente à A et donc on peut remplacer un nœud qui a deux fois le même fils par ce fils
2. *partage :* l'arbre est transformé en *graphe orienté acyclique (DAG)*, c'est-à-dire que deux sous-arbres identiques sont partagés. Dans un graphe, il n'y a plus de notion de sous-arbre gauche et de sous-arbre droit : on étiquette donc les arcs par une valeur de vérité V ou F. Graphiquement on distingue par un *arc plein* le cas où la variable vaut V et par un *arc pointillé* le cas où la variable vaut F
3. *variables ordonnées :* comme dans les BDT, les variables sont ordonnées : la variable d'un fils est toujours strictement supérieure à la variable du père, on parle alors de diagramme de décision binaire ordonné (ou OBDD).

Exemple : diagramme de décision binaire

L'arbre



correspond au diagramme :



Construction d'un OBDD

Pour construire un OBDD à partir d'une formule propositionnelle P :

- partir de la plus petite variable x qui apparaît dans la formule
- construire récursivement l'OBDD P_V correspondant à la formule $P[x \leftarrow \top]$ pour la branche "vrai" et l'OBDD P_F correspondant à la formule $P[x \leftarrow \bot]$ pour la branche "faux".
- Si les deux OBDDs sont les mêmes alors P_V est le résultat attendu, sinon on introduit le nœud x avec un arc plein vers P_V et un arc pointillé vers P_F.

A savoir

- Exprimer des formules logiques sur différents ensembles de connecteurs
- Transformer des formules logiques pour changer de représentation
- Connaître la définition des arbres/diagrammes de décision binaire

1. Formes normales
2. Représentations alternatives des formules
3. Systèmes de déduction
   - Préliminaires (système d'inférence)
   - Système de Hilbert
   - Déduction naturelle
   - Calcul des séquents multi-conclusions

Introduction

- On introduit des méthodes syntaxiques pour prouver qu'une formule P est conséquence logique d'un ensemble de formules $\mathcal{E}$,
(
P est vraie dans toute interprétation qui rend vraies les formules de $\mathcal{E}$).
- les *systèmes de déduction* définissent des relations sur les formules en se donnant des *règles du jeu*, liées à leur structure logique.
- Les relations portent sur les formules elle-mêmes ou bien sur une structure appelée séquent qui se compose d'un ensemble d'hypothèses et d'une ou plusieurs conclusions.
- nous introduisons un outil mathématique : les définitions par système/règles d'inférence.

Système d'inférence : préliminaires

On s'intéresse ici à des outils *mathématiques* généraux et non pas à des formules de la logique

- Les règles d'inférence sont une manière de *définir une relation mathématique* R, très utile en informatique.
- Une relation n-aire R est un sous-ensemble du produit de n ensembles $A_1 \times \ldots \times A_n$.
- On considère une relation sur un ensemble A
- La relation R est donc simplement un sous-ensemble de A.
- On écrira de manière générique $R(t)$ la propriété qui dit que $t \in R$.

Règles d'inférence : principe

- on définit une relation $R \subseteq A$, caractérisée par des conditions que R doit vérifier.
- Les conditions sont représentées par des règles d'inférence.
- Une règle d'inférence se présente comme une fraction :

$$\frac{P_1 \dots P_k}{R(t)}$$

- Le dénominateur (conclusion de la règle) est un cas particulier de la relation R.
- Le numérateur contient un nombre quelconque de conditions (les prémisses de la règle).
- La règle peut utiliser des variables (mathématiques) $x_1 \dots x_p$ (les paramètres).
- La règle se lit : pour toutes les valeurs de $x_1 \dots x_p$, si les prémisses $P_1 \dots P_k$ sont vérifiées alors la conclusion $R(t)$ est vérifiée.
- S'il n'y a aucune condition ($k = 0$), alors $R(t)$ est toujours vrai.

Règles d'inférence : exemple

On définit la relation "être pair" avec les deux règles d'inférences :

$$(p_0)\frac{}{\texttt{pair}(0)} \qquad (p_S)\frac{\texttt{pair}(x)}{\texttt{pair}(x+2)}$$

- Paramètre x dans la règle p_S, : règle valable pour tout entier x : par exemple $\frac{\texttt{pair}(2)}{\texttt{pair}(4)}$ mais aussi $\frac{\texttt{pair}(1)}{\texttt{pair}(3)}$.
- Plusieurs relations vérifient ces propriétés ($\texttt{pair} = \mathbb{N}$).
- Par construction, $\texttt{pair}$ est l'*intersection* de toutes les relations qui vérifient les conditions.
- C'est *la plus petite* relation (au sens de l'inclusion) ayant cette propriété.
- Le fait qu'une telle relation existe et a les bonnes propriétés est un théorème qui nécessite des conditions supplémentaires sur P_i.
  Cas simple : P_i de la forme $R(u)$, ou une condition indépendante de R.
- $\texttt{pair}$ peut se définir sans utiliser de règles d'inférence comme $\exists k, n = 2k$ ou bien $n \ \mathbf{mod}\ 2 = 0$.

Règles d'inférence : définition

Definition (Relation définie par des règles d'inférences)

Soit un ensemble A. Une relation R sur A peut se définir par un système d'inférence qui est un ensemble fini de règles d'inférence.

Chaque règle d'inférence a la forme $\dfrac{R(u_1)\ldots R(u_n)\ P_1\ldots P_k}{R(t)}$

- $t, u_1, \ldots, u_n \in A$ et $P_1, \ldots, P_k$ conditions auxiliaires (indépendantes de R).
- paramètres de la règle : variables mathématiques $x_1 \ldots x_p$.

Chaque règle d'inférence représente une condition sur la relation R :
pour n'importe quelles valeurs de $x_1 \ldots x_p$, si les prémisses $R(u_1)\ldots R(u_n)$ et $P_1 \ldots P_k$ sont vérifiées alors la conclusion $R(t)$ est vérifiée.
La relation R ainsi définie est la plus petite relation qui vérifie l'ensemble des conditions associées aux règles d'inférence.

Règles d'inférence : justification

- Cette définition nécessite une preuve à savoir l'existence d'une plus petite relation qui vérifie les propriétés des règles d'inférence.
- On note $\textsc{Cond}(R)$ l'ensemble des conditions associées aux règles d'inférence de R et on note R_0 l'intersection $\bigcap\{R \subseteq A | \textsc{Cond}(R)\}$ de toutes les relations qui vérifient la condition.
- Cette intersection est bien définie car il y a au moins une relation qui vérifie la condition à savoir la relation qui correspond à l'ensemble A tout entier.
- On montre que R_0 vérifie les conditions (voir preuve dans le polycopié).

Arbre de dérivation : exemple

- on utilise les règles d'inférence pour *justifier* que la relation est vérifiée.
- une preuve de $R(a)$ se construit par étapes comme une séquence de faits (dérivation) de la forme $R(b_1), \dots, R(b_n)$ où chaque $R(b_i)$ s'obtient par une règle dont les prémisses sont dans $R(b_1), \dots, R(b_{i-1})$
- on représente la dérivation comme un arbre dont la racine est la conclusion et chaque nœud correspond à un cas particulier d'une règle d'inférence.

$$
\cfrac{\cfrac{\cfrac{}{\texttt{pair}(0)}\,p_0}{\texttt{pair}(2)}\,p_S}{\texttt{pair}(4)}\,p_S
$$

Arbre de dérivation : définition

- $R \subseteq A$ définie par un système d'inférence
- arbre de dérivation dans ce système :
  - les noeuds de l'arbre portent des instances de la relation de la forme $R(a)$.
  - chaque noeud correspond à une règle d'inférence $\dfrac{R(u_1)\dots R(u_n)\ P_1\dots P_k}{R(t)}$.

    pour certaines valeurs des paramètres $x_1\dots,x_p$: $\dfrac{R(b_1)\dots R(b_n)\ P'_1\dots P'_k}{R(a)}$
  - Pour que la règle soit applicable, il faut que l'ensemble des conditions auxiliaires (instanciées) $P'_1\dots P'_k$ soient vérifiées.
  - Un nœud correspondant à cette règle aura n fils (un fils par prémisse de la forme $R(b_i)$), chaque fils est étiqueté par cette prémisse.

$$\textsc{Nom-Règle}\ \dfrac{\dfrac{\vdots}{R(b_1)}\qquad\dots\qquad\dfrac{\vdots}{R(b_n)}\qquad(\text{vérification } P'_1\dots P'_k)}{R(a)}$$

Arbre de dérivation : vocabulaire

- L'arbre est complet s'il n'y a plus de feuilles (application de règles sans prémisse portant sur R, c'est-à-dire $n = 0$) et il constitue alors une preuve que la racine est dans la relation.
- Un arbre incomplet constitue une preuve du fait que si l'ensemble des propriétés correspondant aux feuilles sont vérifiées alors la propriété à la racine de l'arbre est aussi vérifiée.

Arbre de dérivation : exemple

- L'arbre se dessine en juxtaposant les règles d'inférence instanciées,
- il est recommandé de nommer chaque règle et d'inscrire le nom en marge de la barre de fraction.
- De même les conditions à vérifier seront rappelées en annotation du nœud.
- On pourra noter les valeurs prises par les paramètres de manière à les reporter uniformément dans les prémisses et la conclusion.

Exercice : Déplacement d'un robot

- On modélise le déplacement d'un robot dans le plan
- Les coordonnées sont des couples d'entiers naturels
- Le robot part de la position $(0, 0)$.
- Les seuls déplacements possibles sont d'une case (x, y) à la case $(x + 1, y + 2)$ ou bien à la case $(x + 2, y + 1)$

Questions :

1. Définir par un système d'inférence la relation $\mathtt{pos}$ sur $\mathbb{N} \times \mathbb{N}$ correspondant à toutes les positions accessibles au robot.
2. Construire un arbre de dérivation pour montrer $\mathtt{pos}(3, 3)$
3. Justifier que $\mathtt{pos}(1, 1)$ implique $\mathtt{pos}(3, 5)$

Preuve par minimalité de la relation

- La relation définie est la *plus petite* qui satisfait les conditions associées aux règles d'inférence
- Schéma de preuve associé puissant :
  - si on se donne une autre relation S qui vérifie les conditions associées aux règles d'inférence que R
  - alors pour tout t, si $R(t)$ on a aussi $S(t)$.
- On appellera ce schéma de preuve, une preuve par minimalité de la définition de R (on parle aussi de preuve par récurrence/induction sur la structure de la définition).

Exemple de principe de minimalité

Dans le cas de la définition de `pair`, cela se traduit par le principe suivant :

Proposition (Principe de minimalité sur les entiers pairs.)

- *Soit $\phi(n)$ une propriété à montrer sur les entiers, si on peut démontrer les deux propositions suivantes :*
  1. *$\phi(0)$ est vérifié ;*
  2. *pour tout n arbitraire tel que $\mathtt{pair}(n)$ et tel que $\phi(n)$ est vérifié, on peut établir que $\phi(n+2)$ est vérifié ;*
- *alors on en déduit que pour tout n tel que $\mathtt{pair}(n)$, on a $\phi(n)$ est vérifié.*

Application : prouver que si n vérifie `pair` alors il existe k tel que $n = 2k$.
On vérifie les conditions pour la propriété $\phi(n)$: “il existe k tel que $n = 2k$” :

1. il existe k tel que $0 = 2k$ (il suffit de prendre $k = 0$)
2. soit n arbitraire et supposons qu'il existe k tel que $n = 2k$, montrons qu'il existe k' tel que $n + 2 = 2k'$. Il suffit de prendre $k' = k + 1$.

Le schéma de minimalité : définition

- schéma de minimalité associé à la définition par un système d'inférence de $R \subseteq A$
- Soit $\phi(x)$ une propriété paramétrée par $x \in A$
- Pour *chaque règle d'inférence* de la relation R,

$$\frac{R(u_1) \ldots R(u_n)\ P_1 \ldots P_k}{R(t)}$$

  paramètrée par $x_1, \ldots, x_p$,
  il faut vérifier que la propriété ϕ est *préservée* par la règle, ce qui s'écrit :
  - pour n'importe quelles valeurs des paramètres $x_1, \ldots, x_p$, telles que les prémisses sont vérifiés (c'est-à-dire $R(u_1) \ldots R(u_n)\ P_1 \ldots P_k$) et telles que $\phi(u_1) \ldots \phi(u_n)$ (hypothèses de récurrence) sont vérifiés,
    alors $\phi(t)$ est vérifié
- Si on a vérifié toutes les conditions précédentes (une par règle d'inférence), alors $\phi(t)$ est vérifié pour tous les éléments t tels que $R(t)$.

Schéma de minimalité : illustration

- On souhaite montrer un résultat de la forme

$$\text{Pour tout } t \in A, \text{ si } R(t) \text{ alors } \phi(t)$$

- $R(t)$ est une relation définie par un système d'inférence
- $\phi(t)$ est une propriété quelconque
- Principe de minimalité
  - pour chaque règle $\dfrac{R(u_1) \dots R(u_n) \; P_1 \dots P_k}{R(t)}$
  - on vérifie pour des paramètres quelconques

$$\frac{R(u_1) \dots R(u_n) \; \phi(u_1) \dots \phi(u_n) \; P_1 \dots P_k}{\phi(t)}$$

Exercice

On reprend la définition des positions du robot.

1. *Enoncer le principe de minimalité associé à cette définition.*
2. *Utiliser ce principe pour montrer que pour tout* $x\ y$, *si* $\mathtt{pos}(x, y)$ *alors* $x + y$ **mod** $3 = 0$
3. *En déduire que* $\mathtt{pos}(1, 1)$ *est faux*

Exercice

Les systèmes d'inférence sont utiles pour définir des relations générales qui ne sont pas des fonctions.

- Relation $\texttt{ami}(x, y)$: x a déclaré y comme ami dans un réseau social
- Définir $\texttt{reseau}(x, y)$: il y a une *chaîne d'amitié* entre x et y.

Caractérisation de la relation `reseau` :

1. Si x et y sont amis alors ils sont dans le même réseau.
2. Si x et y sont dans le même réseau et y et z sont amis alors x et z sont dans le même réseau.
3. Il n'y a pas d'autre manière d'être dans le même réseau : si quelqu'un est dans mon réseau, il est soit mon ami, soit l'ami de quelqu'un dans mon réseau

Donner un système d'inférence correspondant.

Système d'inférence : applications

- Les systèmes d'inférence ont des applications en logique pour définir des systèmes de déduction,
- ils sont également beaucoup utilisés en informatique pour modéliser la sémantique des langages de programmation, par exemple les règles de typage ou les traductions entre langages en particulier dans le domaine de la compilation.
- c'est aussi une technique qui permet de modéliser des systèmes informatiques ayant des comportements non déterministes (systèmes parallèles et communicants)


That's all Folks!

3–Manipuler les formules

La dernière fois : Introduction

- On introduit des méthodes syntaxiques pour prouver qu'une formule P est conséquence logique d'un ensemble de formules $\mathcal{E}$, (P est vraie dans toute interprétation qui rend vraies les formules de $\mathcal{E}$).
- les *systèmes de déduction* définissent des relations sur les formules en se donnant des *règles du jeu*, liées à leur structure logique.
- Les relations portent sur les formules elle-mêmes ou bien sur une structure appelée séquent qui se compose d'un ensemble d'hypothèses et d'une ou plusieurs conclusions.
- nous introduisons un outil mathématique : les définitions par système/règles d'inférence.

La dernière fois : Règles d'inférence

Definition (Relation définie par des règles d'inférences)

Soit un ensemble A. Une relation R sur A peut se définir par un système d'inférence qui est un ensemble fini de règles d'inférence.

Chaque règle d'inférence a la forme $\dfrac{R(u_1)\dots R(u_n)\ P_1\dots P_k}{R(t)}$

- $t, u_1, \dots, u_n \in A$ et $P_1, \dots, P_k$ conditions auxiliaires (indépendantes de R).
- paramètres de la règle : variables mathématiques $x_1 \dots x_p$.

Chaque règle d'inférence représente une condition sur la relation R :
pour n'importe quelles valeurs de $x_1 \dots x_p$, si les prémisses $R(u_1)\dots R(u_n)$ et $P_1 \dots P_k$ sont vérifiées alors la conclusion $R(t)$ est vérifiée.
La relation R ainsi définie est la plus petite relation qui vérifie l'ensemble des conditions associées aux règles d'inférence.

Exemple (GLA TD 8 exo 3)

$$\dfrac{\dfrac{}{\vdash \{x < y\}\ \texttt{x:=x-1}\ \{x \leq y-2\}}\,\text{aff} \qquad \dfrac{}{\vdash \{x \leq y-2\}\ \texttt{y:=y-2}\ \{x \leq y\}}\,\text{aff}}{\dfrac{\vdash \{x \leq y \wedge x \neq y\}\ \texttt{x:=x-1;y:=y-2}\ \{x \leq y\}}{\vdash \{x \leq y\}\ \texttt{WHILE x != y DO x:=x-1;y:=y-2}\ \{x \leq y \wedge x = y\}}\,\text{while}}\,\text{seq}$$

La dernière fois : Schéma de minimalité

- On souhaite montrer un résultat de la forme

  Pour tout $t \in A$, si $R(t)$ alors $\phi(t)$
- $R(t)$ est une relation définie par un système d'inférence
- $\phi(t)$ est une propriété quelconque
- Principe de minimalité
  - pour chaque règle $\dfrac{R(u_1) \dots R(u_n)\ P_1 \dots P_k}{R(t)}$
  - on vérifie pour des paramètres quelconques

$$\frac{R(u_1) \dots R(u_n)\ \phi(u_1) \dots \phi(u_n)\ P_1 \dots P_k}{\phi(t)}$$

- Si on a vérifié toutes les conditions précédentes (une par règle d'inférence), alors $\phi(t)$ est vérifié pour tous les éléments t tels que $R(t)$.

Systèmes de déduction

- systèmes d'inférence pour prouver la validité de formules en utilisant des propriétés syntaxiques : on parlera de *système de déduction logique*.
- un raisonnement travaille de manière générale avec un ensemble d'hypothèses Γ et une formule P.

Definition (Séquent)

- On appelle séquent un couple formé d'un ensemble fini de formules Γ (les hypothèses) et une formule P appelée la conclusion. On note le séquent $\Gamma \vdash P$.
- La formule associée au séquent $P_1, \ldots, P_n \vdash Q$ est définie comme $(P_1 \wedge \ldots \wedge P_n) \Rightarrow Q$
- Un séquent $\Gamma \vdash P$ est valide si P est conséquence logique de Γ ($\Gamma \models P$) ou de manière équivalente si la formule associée au séquent est valide.

Notations associées aux séquents

- l'ordre des formules n'intervient pas dans la définition d'un ensemble. Les ensembles $\{A, B, C\}$ et $\{C, A, B\}$ sont égaux.
- si Γ est un ensemble fini de formules $\{A_1, \dots, A_n\}$, on écrit $A_1, \dots, A_n \vdash P$ pour le séquent $\Gamma \vdash P$.
- si Γ est l'ensemble vide, on note simplement $\vdash P$.
- si Q est une formule, on note Γ, Q l'ensemble Γ auquel on a ajouté la formule Q soit $\Gamma \cup \{Q\}$
- de même si Γ et Δ sont des ensembles finis de formules, on note Γ, Δ l'ensemble $\Gamma \cup \Delta$
- une formule P est valide si et seulement si le séquent $\vdash P$ est valide

Séquents prouvables

- systèmes d'inférence pour capturer l'ensemble des séquents "prouvables".
- traditionnellement, on n'introduit pas de notation supplémentaire.
- ainsi suivant le contexte, la notation $\Gamma \vdash P$ représentera l'objet séquent (un couple formé d'un ensemble d'hypothèses et d'une conclusion) ou bien la propriété qui dit que ce séquent en prouvable dans le système considéré.
- on n'introduira évidemment que des systèmes corrects : tout séquent prouvable est valide
- on cherchera à ce que les systèmes soient complets à savoir que tout séquent valide soit prouvable.

Les règles d'inférence pour faire des preuves reposent sur la syntaxe des formules, ainsi il n'est pas correct de transformer dans un séquent une formule par une autre formule équivalente (sauf si une règle nous autorise explicitement à le faire).

Système de Hilbert

- MIN : ensemble des formules avec juste des variables propositionnelles et le connecteur $\Rightarrow$
- formules prouvables : relation $\vdash P$ avec $P \in \text{MIN}$

$$(K)\frac{}{\vdash p \Rightarrow q \Rightarrow p} \qquad (S)\frac{}{\vdash (p \Rightarrow q \Rightarrow r) \Rightarrow (p \Rightarrow q) \Rightarrow (p \Rightarrow r)}$$

$$(mp)\frac{\vdash p \Rightarrow q \qquad \vdash p}{\vdash q}$$

- arbre de dérivation de $\vdash p \Rightarrow p$.

$$mp\,\dfrac{mp\,\dfrac{S\,\dfrac{}{\vdash (p \Rightarrow (p \Rightarrow p) \Rightarrow p) \Rightarrow (p \Rightarrow p \Rightarrow p) \Rightarrow (p \Rightarrow p)} \quad K\,\dfrac{}{\vdash p \Rightarrow (p \Rightarrow p) \Rightarrow p}}{\vdash (p \Rightarrow p \Rightarrow p) \Rightarrow (p \Rightarrow p)} \quad K\,\dfrac{}{\vdash p \Rightarrow p \Rightarrow p}}{\vdash p \Rightarrow p}$$

Système de Hilbert : correction

- La preuve de “si $\vdash P$ alors $\models P$” se fait en utilisant le principe de minimalité associé à la définition de $\vdash$.
- $\phi(P)$ est défini comme $\models P$
- préservation de ϕ à vérifier pour chaque règle d’inférence pour n’importe quelles formules p, q et r :
  1. $\models p \Rightarrow q \Rightarrow p$
  2. $\models (p \Rightarrow q \Rightarrow r) \Rightarrow (p \Rightarrow q) \Rightarrow (p \Rightarrow r)$
  3. si $\models p \Rightarrow q$ et $\models p$ alors $\models q$

  ce qui se vérifie simplement.

Système de Hilbert : ajout d'hypothèses

- Extension au raisonnement sous hypothèse.
- Les règles intégrent les hypothèses.
- Une règle est ajoutée qui dit que toute formule qui apparaît dans les hypothèses Γ peut être déduite.

$$(\text{hyp})\frac{p \in \Gamma}{\Gamma \vdash p}$$

$$(K)\frac{}{\Gamma \vdash p \Rightarrow q \Rightarrow p} \qquad (S)\frac{}{\Gamma \vdash (p \Rightarrow q \Rightarrow r) \Rightarrow (p \Rightarrow q) \Rightarrow (p \Rightarrow r)}$$

$$(mp)\frac{\Gamma \vdash p \Rightarrow q \quad \Gamma \vdash p}{\Gamma \vdash q}$$

Système de Hilbert : affaiblissement

Proposition (Affaiblissement des hypothèses)

Si $\Gamma \vdash p$*, alors pour tout ensemble* Δ *tel que* $\Gamma \subseteq \Delta$*, on a* $\Delta \vdash p$*.*

Preuve:

- *principe de minimalité associé à la définition.*
- *la propriété* $\phi(p)$ *est* $\Delta \vdash p$*.*
- *seul cas non trivial : règle hypothèse pour laquelle on a* $p \in \Gamma$*, comme* $\Gamma \subseteq \Delta$*, on a aussi* $p \in \Delta$*, on en déduit* $\Delta \vdash p$*.*

$\square$

Proposition (Affaiblissement de la conclusion)

Si $\Gamma \vdash p$ *alors* $\Gamma \vdash q \Rightarrow p$*.*

Preuve: *C'est une simple déduction sous hypothèse :*

$$\dfrac{K\,\dfrac{}{\Gamma \vdash p \Rightarrow q \Rightarrow p} \qquad \Gamma \vdash p}{\Gamma \vdash q \Rightarrow p}\ mp$$

Système de Hilbert : théorème de déduction

Proposition (Théorème de déduction)

Soit Γ un ensemble de formules et p, q deux formules. On a $\Gamma, p \vdash q$ si et seulement si $\Gamma \vdash p \Rightarrow q$.

Preuve: *(sens si)*

- *Si $\Gamma \vdash p \Rightarrow q$ alors*
  - *par le théorème d'affaiblissement des hypothèses, on a aussi $\Gamma, p \vdash p \Rightarrow q$*
  - *par la règle hypothèse, $\Gamma, p \vdash p$*
  - *par la règle modus-ponens, on a donc $\Gamma, p \vdash q$*

□

suite de la preuve

Preuve: *(sens seulement si)*

- *on suppose que* $\Gamma, p \vdash q$
- *on montre* $\Gamma \vdash p \Rightarrow q$ *en utilisant le principe de minimalité associé à la définition de* $\Gamma, p \vdash q$.
- *La propriété* $\phi(q)$ *est* $\Gamma \vdash p \Rightarrow q$
- *On vérifie les cas correspondant aux 4 règles*
  1. *règle hypothèse* $\Gamma, p \vdash q$ *avec* $q \in \Gamma, p$:
     - *soit* $q \in \Gamma$ *et alors on a* $\Gamma \vdash q$ *et donc par affaiblissement de la conclusion on a* $\Gamma \vdash p \Rightarrow q$,
     - *si* $q = p$ *alors on obtient la dérivation de* $\Gamma \vdash p \Rightarrow p$
  2. *les deux règles* S *et* K *sont applicables quelque soit l'ensemble d'hypothèses, on utilise l'affaiblissement de la conclusion*
  3. *la règle modus ponens est* $\dfrac{\Gamma, p \vdash r \Rightarrow q \qquad \Gamma, p \vdash r}{\Gamma, p \vdash q}$

     on a donc $\Gamma \vdash p \Rightarrow r \Rightarrow q$ *et* $\Gamma \vdash p \Rightarrow r$

$$
mp\ \dfrac{mp\ \dfrac{S\ \dfrac{}{\Gamma \vdash (p \Rightarrow r \Rightarrow q) \Rightarrow (p \Rightarrow r) \Rightarrow (p \Rightarrow q)} \qquad \Gamma \vdash p \Rightarrow r \Rightarrow q}{\Gamma \vdash (p \Rightarrow r) \Rightarrow p \Rightarrow q} \qquad \Gamma \vdash p \Rightarrow r}{\Gamma \vdash p \Rightarrow q}
$$

Système de Hilbert : conclusion

- Dans le système de Hilbert, la relation de déduction est définie avec un ensemble fixe d'hypothèses,
- Le théorème de déduction peut se voir comme une règle "dérivée" qui transforme simultanément l'ensemble des hypothèses et la conclusion.
- L'idée des calculs de séquents est d'intégrer dans les règles d'inférence des manipulations sur l'ensemble des hypothèses.
- Il en existe plusieurs variantes.
- Nous présentons ici la déduction naturelle et le calcul des séquents à conclusions multiples.

Déduction naturelle : introduction

- La déduction naturelle est un système dû à Gerhart Gentzen qui est proche du raisonnement mathématique naturel.
- Nous allons définir une relation qui correspond au sous-ensemble des séquents “prouvables en déduction naturelle”.
- Nous présentons un ensemble de règles d'inférence

$$\frac{\Gamma_1 \vdash A_1 \quad \dots \quad \Gamma_n \vdash A_n}{\Gamma \vdash P}$$

qui permet de construire un arbre de preuve dont la racine (en bas) représente le séquent que l'on cherche à justifier et les feuilles sont des conditions suffisantes.

Règles de base

Une première série de règles permet de terminer une preuve dans les cas “élémentaires” :

- Hypothèse : si la conclusion fait partie des hypothèses alors la preuve est terminée.

$$\text{hyp}\,\frac{A \in \Gamma}{\Gamma \vdash A}$$

- Si la conclusion est atomique toujours vraie alors elle est prouvable

$$\text{triv}\,\frac{}{\Gamma \vdash \top}$$

Règles d'introduction

Les *règles d'introduction* ramènent la preuve d'une formule à des preuves de certaines de ses sous-formules

$(\wedge I)$ $$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$$

$(\vee I)$ $$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

Attention l'utilisation de ces règles peut ne pas aboutir ($A \vee B \vdash B \vee A$)

$(\neg I)$ $$\frac{\Gamma, A \vdash \perp}{\Gamma \vdash \neg A}$$

$(\Rightarrow I)$ $$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

$(\forall I)$ $$\frac{\Gamma \vdash P \quad x \notin \mathrm{vl}(\Gamma)}{\Gamma \vdash \forall x, P}$$

Le nom x ne doit pas être utilisé ailleurs (renommer si nécessaire)

$(\exists I)$ $$\frac{\Gamma \vdash P[x \leftarrow t]}{\Gamma \vdash \exists x, P}$$

Attention au choix de l'objet t !

Règles d'introduction : applications

- $A \Rightarrow B \Rightarrow A$

$$\dfrac{\dfrac{\dfrac{}{A, B \vdash A}\ \text{hyp}}{A \vdash B \Rightarrow A}\ \Rightarrow I}{\vdash A \Rightarrow B \Rightarrow A}\ \Rightarrow I$$

- $A \Rightarrow (A \vee B)$

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\ \text{hyp}}{A \vdash A \vee B}\ \vee I}{\vdash A \Rightarrow (A \vee B)}\ \Rightarrow I$$

- $(A \wedge B) \Rightarrow (A \vee B)$: règles insuffisantes

Les règles d'introduction ajoutent des connecteurs dans la conclusion d'autres règles vont permettre d'utiliser les hypothèses en les décomposant.

Règles d'élimination

Les *règles d'élimination* expliquent comment *utiliser* une preuve d'une formule en fonction du connecteur principal de cette formule.

$$(\bot E)\quad \frac{\Gamma \vdash \bot}{\Gamma \vdash C}$$

$$(\neg E)\quad \frac{\Gamma \vdash \neg A \qquad \Gamma \vdash A}{\Gamma \vdash C}$$

$$(\wedge E)\quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad \wedge E\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

$$(\vee E)\quad \frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}$$

$$(\Rightarrow E)\quad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

$$(\forall E)\quad \frac{\Gamma \vdash \forall x, P}{\Gamma \vdash P[x \leftarrow t]}$$

$$(\exists E)\quad \frac{\Gamma \vdash \exists x, P \qquad \Gamma, P \vdash C \qquad x \notin \mathrm{vl}(\Gamma, C)}{\Gamma \vdash C}$$

renommage si nécessaire

Raisonnement par l'absurde

- Les règles précédentes ne suffisent pas à prouver toutes les formules valides (par exemple $A \vee \neg A$),
- il faut y ajouter le schéma de raisonnement *classique* qui capture ce que l'on appelle le *raisonnement par l'absurde*.
- Si on veut prouver A alors on peut supposer $\neg A$ et en déduire une contradiction.

$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}$$

ou de manière équivalente

$$\frac{\Gamma, \neg A \vdash A}{\Gamma \vdash A}$$

Règles dérivées

Proposition (Affaiblissement)

Si $\Gamma \vdash A$ *et* $\Gamma \subseteq \Delta$ *alors* $\Delta \vdash A$

Lorsqu'on construit une preuve de la racine aux feuilles, on peut "effacer"certaines hypothèses
La preuve reste correcte mais ne pas effacer trop d'hypothèses !

Règles dérivées

Règles d'élimination sur des formules issues des hypothèses :

$$(\bot H) \quad \frac{\bot \in \Gamma}{\Gamma \vdash C}$$

$$(\neg H) \quad \frac{\Gamma \vdash A \qquad \neg A \in \Gamma}{\Gamma \vdash C}$$

$$(\wedge H) \quad \frac{\Gamma, A, B \vdash C \qquad A \wedge B \in \Gamma}{\Gamma \vdash C}$$

$$(\vee H) \quad \frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C \qquad A \vee B \in \Gamma}{\Gamma \vdash C}$$

$$(\Rightarrow H) \quad \frac{\Gamma, B \vdash C \qquad \Gamma \vdash A \qquad A \Rightarrow B \in \Gamma}{\Gamma \vdash C}$$

$$(\forall H) \quad \frac{\Gamma, P[x \leftarrow t] \vdash C \qquad \forall x, P \in \Gamma}{\Gamma \vdash C}$$

$$(\exists H) \quad \frac{\Gamma, P \vdash C \qquad x \notin \mathrm{vl}(\Gamma, C) \quad \exists x, P \in \Gamma}{\Gamma \vdash C}$$

Exemple

Dérivation de $\neg A \vee B \Rightarrow A \Rightarrow B$.
Arbre de preuve :

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\neg A \vee B, A, \neg A \vdash A}\,\text{hyp}}{\neg A \vee B, A, \neg A \vdash B}\,\neg H \qquad \dfrac{}{\neg A \vee B, A, B \vdash B}\,\text{hyp}}{\neg A \vee B, A \vdash B}\,\vee H}{\dfrac{\neg A \vee B \vdash A \Rightarrow B}{\vdash \neg A \vee B \Rightarrow A \Rightarrow B}\,\Rightarrow I}\,\Rightarrow I
$$

Exercice : Preuve en déduction naturelle (poly exo 3.9)

Construire des démonstrations en déduction naturelle des formules et séquents suivants :

1. $(A \Rightarrow \neg A) \Rightarrow \neg A$
2. $\exists x, P(x) \vdash \neg \forall x, \neg P(x)$
3. $\neg \exists x, P(x) \vdash \forall y, \neg P(y)$

Exercice : Raisonnement par l'absurde (poly exo 3.10)

Construire des arbres de dérivation pour faire les preuves en déduction naturelle des séquents suivants :

1. $P \vdash \neg\neg P$ et $\neg\neg P \vdash P$ (pour ce séquent, commencer par la règle de raisonnement par l'absurde)
2. $\neg(\neg P \vee P) \vdash \neg P$, en déduire en utilisant le raisonnement par l'absurde une dérivation de $\vdash \neg P \vee P$

3–Manipuler les formules

Limites de la déduction naturelle

- les formules naviguent entre hypothèses et conclusion, la structure des formules en hypothèses ne bouge pas
- règles qui transforment les hypothèses utiles (règles dérivées).
- plusieurs règles non “inversibles” (conclusion vraie sans que les prémisses le soient).

$$\frac{\vdash A}{\vdash A \vee B} \quad \frac{\vdash B}{\vdash A \vee B} \quad \frac{\vdash P[x \leftarrow t]}{\vdash \exists x, P} \quad \frac{\vdash A \wedge B}{\Gamma \vdash A} \quad \frac{\vdash A \wedge B}{\vdash B} \quad \frac{\vdash \forall x, P}{\vdash P[x \leftarrow t]} \quad \frac{\vdash A \Rightarrow B \quad \vdash A}{\vdash B}$$

- règles d'élimination d'hypothèses pour $\wedge$ et $\forall$: pas de perte d'information

$$(\wedge H)\frac{\Gamma, A, B \vdash C \qquad A \wedge B \in \Gamma}{\Gamma \vdash C} \qquad (\forall H)\frac{\Gamma, P[x \leftarrow t] \vdash C \qquad \forall x, P \in \Gamma}{\Gamma \vdash C}$$

  mais reste problème disjonction, existentielle et implication.
- disymétrie entre plusieurs hypothèses et une seule conclusion.

Séquent généralisé à conclusions multiples

calcul des séquents : autoriser plusieurs formules à droite du symbole $\vdash$.

Definition (Séquent à conclusions multiples)

- Un séquent à conclusions multiples est formé de deux ensembles finis de formules Γ (les hypothèses) et Δ (les conclusions). On le note $\Gamma \vdash \Delta$.
- La formule associée à $P_1, \ldots, P_n \vdash Q_1, \ldots, Q_p$ est $(P_1 \wedge \ldots \wedge P_n) \Rightarrow (Q_1 \vee \ldots \vee Q_p)$.
  Si $\Delta = \emptyset$ ($p = 0$) alors $(Q_1 \vee \ldots \vee Q_p)$ est défini comme $\perp$ et la formule associée s'écrit $(P_1 \wedge \ldots \wedge P_n) \Rightarrow \perp$ $(\equiv \neg(P_1 \wedge \ldots \wedge P_n))$
- Un séquent $\Gamma \vdash \Delta$ est valide ssi la formule associée au séquent est valide, ie pour toute interprétation I telle que $I \models \Gamma$, il existe $P \in \Delta$ tel que $I \models P$.

- l'ensemble des hypothèses à gauche de $\vdash$ est interprété comme une conjonction (toutes les hypothèses sont vraies)
- l'ensemble des conclusions à droite du signe $\vdash$ est interprété comme une disjonction (au moins une des conclusions est vraie).

Exercice TD : séquent multi-conclusion

Donner pour chacun des séquents suivants la formule associée et dire si les séquents sont valides.

1. $(p \vee q \Rightarrow r), \neg(p \wedge z) \vdash \neg p, \neg z$
2. $\vdash (r \wedge s), q$
3. $(p \vee q) \Rightarrow r, \neg(p \vee z) \vdash$
4. $\vdash$
5. $\bot \vdash$

Règles du calcul des séquents

- On appelle ce système de déduction, le *Système G*
- Les règles permettent de décomposer les connecteurs soit dans la partie *gauche* du séquent, soit dans la partie *droite*.
- Règles terminales (cas triviaux)

$$(\textsc{Hyp})\frac{}{A, \Gamma \vdash \Delta, A} \qquad (\textsc{Jok})\frac{}{\bot, \Gamma \vdash \Delta} \qquad (\textsc{Triv})\frac{}{\Gamma \vdash \Delta, \top}$$

Règles pour chaque connecteur : on retrouve JOK (règle gauche de $\bot$) et TRIV (règle droite de $\top$).

Règles du système G

$$(\bot g) \quad \frac{}{\bot, \Gamma \vdash \Delta} \qquad (\bot d) \quad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \bot}$$

$$(\top g) \quad \frac{\Gamma \vdash \Delta}{\top, \Gamma \vdash \Delta} \qquad (\top d) \quad \frac{}{\Gamma \vdash \Delta, \top}$$

$$(\neg g) \quad \frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta} \qquad (\neg d) \quad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}$$

$$(\wedge g) \quad \frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta} \qquad (\wedge d) \quad \frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}$$

$$(\vee g) \quad \frac{A, \Gamma \vdash \Delta \quad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta} \qquad (\vee d) \quad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B}$$

$$(\Rightarrow g) \quad \frac{\Gamma \vdash \Delta, A \quad B, \Gamma \vdash \Delta}{A \Rightarrow B, \Gamma \vdash \Delta} \qquad (\Rightarrow d) \quad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B}$$

$$(\forall g) \quad \frac{P[x \leftarrow t], (\forall x, P), \Gamma \vdash \Delta}{(\forall x, P), \Gamma \vdash \Delta} \qquad (\forall d) \quad \frac{\Gamma \vdash \Delta, P \qquad x \notin \mathrm{vl}(\Gamma, \Delta)}{\Gamma \vdash \Delta, (\forall x, P)}$$

$$(\exists g) \quad \frac{P, \Gamma \vdash \Delta \qquad x \notin \mathrm{vl}(\Gamma, \Delta)}{(\exists x, P), \Gamma \vdash \Delta} \qquad (\exists d) \quad \frac{\Gamma \vdash \Delta, (\exists x, P), P[x \leftarrow t]}{\Gamma \vdash \Delta, (\exists x, P)}$$

Exemple

On peut montrer $\neg(A \wedge B) \vdash \neg A \vee \neg B$.

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash \neg B, A}\,\text{hyp}}{\vdash \neg A, \neg B, A}\,\neg d \qquad \dfrac{\dfrac{}{B \vdash \neg A, B}\,\text{hyp}}{\vdash \neg A, \neg B, B}\,\neg d}{\vdash \neg A, \neg B, A \wedge B}\,\wedge d}{\dfrac{\neg(A \wedge B) \vdash \neg A, \neg B}{\neg(A \wedge B) \vdash \neg A \vee \neg B}\,\vee d}\,\neg g
$$

Exemples

Montrer les séquents

1. $\exists x, (p(x) \vee q(x)) \vdash (\exists x, p(x)) \vee (\exists x, q(x))$
2. $\neg \exists y, p(y) \vdash \forall x, \neg p(x)$

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{p(x) \vdash p(x), (\exists x, p(x)), (\exists x, q(x))}\ \text{hyp}}{p(x) \vdash (\exists x, p(x)), (\exists x, q(x))}\ \exists d \qquad \dfrac{\vdots}{q(x) \vdash (\exists x, p(x)), (\exists x, q(x))}}{p(x) \vee q(x) \vdash (\exists x, p(x)), (\exists x, q(x))}\ \vee g}{\dfrac{\exists x, (p(x) \vee q(x)) \vdash (\exists x, p(x)), (\exists x, q(x))}{\exists x, (p(x) \vee q(x)) \vdash (\exists x, p(x)) \vee (\exists x, q(x))}\ \vee d}\ \exists g
$$

$$
\neg g\ \dfrac{\forall d\ \dfrac{\neg d\ \dfrac{\exists d\ \dfrac{\text{hyp}\ \dfrac{}{p(x) \vdash p(x), (\exists y, p(y))}}{p(x) \vdash \exists y, p(y)}}{\vdash (\exists y, p(y)), \neg p(x)}}{\vdash (\exists y, p(y)), (\forall x, \neg p(x))}}{\neg(\exists y, p(y)) \vdash \forall x, \neg p(x)}
$$

Propriétés du système G

Toutes les règles sont *correctes*, elles sont aussi *inversibles* (une interprétation rend vraie le séquent en conclusion d'une règle ssi elle rend vrais les séquents prémisses).

Proposition (Correction du système G)

Les règles du système G sont correctes et inversibles.

Preuve:

- *validité des formules associées ou tables de vérité sur les séquents*
- *un séquent est faux dans une interprétation ssi toutes les hypothèses sont vraies et toutes les conclusions sont fausses*

Cas de la règle gauche de l'implication $\dfrac{\Gamma \vdash \Delta, A \quad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}$

Γ	Δ	A	B	$\Gamma \vdash \Delta, A$	$\Gamma, B \vdash \Delta$	$\Gamma, A \Rightarrow B \vdash \Delta$
F	_	_	_	V	V	V
V	V	_	_	V	V	V
V	F	F	_	F	_	F
V	F	V	V	V	F	F
V	F	V	F	V	V	V

Règles pour les quantificateurs

$$(\forall g)\,\frac{P[x\leftarrow t], (\forall x, P), \Gamma \vdash \Delta}{(\forall x, P), \Gamma \vdash \Delta} \qquad (\exists d)\,\frac{\Gamma \vdash \Delta, (\exists x, P), P[x\leftarrow t]}{\Gamma \vdash \Delta, (\exists x, P)}$$

- Règles inversibles : cas particulier d'affaiblissement
- Possibilité d'utiliser plusieurs fois une hypothèse, et de prouver plusieurs fois une conclusion
- Le séquent prémisse est plus gros que le séquent conclusion

Propriétés du système G

- Règles dérivées
  - affaiblissement

$$\text{si } \Gamma \vdash \Delta \text{ alors } \Gamma, \Gamma' \vdash \Delta, \Delta'$$

  - substitution

$$\text{si } \Gamma \vdash \Delta \text{ alors } \Gamma[x \leftarrow t] \vdash \Delta[x \leftarrow t]$$

- Propriété de la sous-formule :
  toutes les formules qui apparaissent dans un arbre de preuve de $\Gamma \vdash \Delta$ sont des sous-formules de Γ, Δ
  (avec la définition que $P[x \leftarrow t]$ est une sous-formule de $\forall x, P$ et de $\exists x, P$)

Règle de coupure

Definition (Règle de coupure)

On peut ajouter au système la règle dite de *coupure*.

$$(\textsc{Cut})\frac{\Gamma \vdash \Delta, A \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

- Cette règle est correcte, inversible mais elle ne préserve pas la propriété de la sous formule.
- La règle de coupure est redondante :

Proposition (Elimination des coupures)

Si $\Gamma \vdash \Delta$ en utilisant la règle de coupure alors $\Gamma \vdash \Delta$ sans la règle de coupure.

La preuve se fait en transformant les coupures pour faire “diminuer” leur taille

Cas propositionnel

Le calcul des séquents à conclusions multiples nous fournit une procédure de décision de la validité pour les formules propositionnelles.

- dans le cas propositionnel (sans quantificateur) chaque séquent qui apparaît en prémisse d'une règle d'inférence contient moins de connecteurs logiques que celui qui est en conclusion
- le processus qui consiste à appliquer les règles logiques s'arrête forcément
- la profondeur de l'arbre est au maximum égale au nombre de symboles dans le séquent.
- un séquent propositionnel qui n'est pas prouvable n'est pas valide
- donc un séquent propositionnel valide est prouvable (complétude)

Preuves classiques

- Contrairement à la déduction naturelle : pas de règle spéciale pour les preuves par l'absurde.
- On tire partie des conclusions multiples : commencer à décomposer une des conclusions et utiliser les éléments collectés pour établir une autre conclusion.

Exemple (Preuve du tiers exclu)

$$\dfrac{\dfrac{\dfrac{}{P \vdash P}\,\text{hyp}}{\vdash P, \neg P}\,\neg d}{\vdash P \lor \neg P}\,\lor d$$

Exercice : Preuves en calcul des séquents (poly exo 3.11)

Prouver en utilisant les règles du calcul des séquents les propriétés suivantes :

1. $A \Rightarrow (B \Rightarrow A)$
2. $(\neg A \Rightarrow A) \Rightarrow A$
3. $\neg \forall x, \neg P(x) \vdash \exists x, P(x)$
4. $\neg\neg P \vdash P$
5. Paradoxe du buveur : $\exists x, (\neg P(x) \Rightarrow \forall y, \neg P(y))$.

Résultat sur les systèmes de déduction

- Théorème de complétude de Gödel :
  - Il existe un système de preuve pour les formules du calcul des prédicats qui est correct et complet (vrai pour le raisonnement dans une théorie).

$$\mathcal{E} \vdash P \text{ ssi } \mathcal{E} \models P$$

- Si une formule est valide alors on peut en construire une preuve (énumération naive des arbres de dérivation)
- Tout séquent prouvable en déduction naturelle est aussi prouvable en calcul des séquents
- Si $\Gamma \vdash \Delta$ en calcul des séquents alors $\Gamma, \neg\Delta \vdash \perp$ est prouvable en déduction naturelle.

A savoir faire

- Comprendre le sens d'une relation définie par un système d'inférence
- Savoir construire des arbres de dérivation à partir d'un système d'inférence
- Savoir justifier la validité d'un séquent à partir d'une preuve (en déduction naturelle ou) en calcul des séquents.


That's all Folks!