Module Structures Discrètes 2

Logique 2 : Preuves en Logiques

Philippe Balbiani, Frédéric Maris,Jan-Georg Smaus, Martin Strecker
(basé sur travail de Jean-Baptiste Raclet)

Année universitaire 2022-23, Semestre 2

Université de Toulouse III / IRIT

Structures discrètes 2 (partie Logique)

- **Intervenants** :
  - Philippe Balbiani – balbiani@irit.fr
  - Frédéric Maris – maris@irit.fr
  - Jan-Georg Smaus – smaus@irit.fr
  - Martin Strecker – strecker@irit.fr (responsable partie Logique)
- **Volume horaire**
  - Module commun avec Probas/Stats : 6 ECTS
  - Partie Logique : 24h de CTD + un CC en séance
  - Partie Proba/Stats : 24h de CTD + un CC en séance
  - CC commun en semaine 40
  - CC commun de seconde chance en semaine 50
- **Page moodle** :
  `http://moodle.univ-tlse3.fr/course/view.php?id=6129`

Structures discrètes 2 (partie Logique) : Examens

- CC1 commun
  sem. 8 (gr. A : jeudi 23/02 à 18h ; gr. B : mardi 21/02 à 18h)
- CC2 Logique
  sem. 13 (gr. A : mardi 28/03 à 13h30 ; gr. B : mardi 28/03 à 15h45)
- (CC3 Probas/Stats)
- CC4 commun
  sem. 19 (gr. A : mardi 09/05 à 13h30 ; gr. B : mardi 09/05 à 15h45)

La partie Logique compte 50% de la note de CC1 et CC4.

Note finale =
0.25 * max(CC1,CC4) + 0.25 * max(CC2,CC4) + 0.25 * max(CC3,CC4)
+ 0.25 * CC4

Objectif et contenu

Objectif :

S'approprier les bases logiques sur de la théorie de la preuve et de sa méta-théorie.

Contenu : seront présentés les concepts fondamentaux des preuves mathématiques ainsi que diverses techniques de preuve :

- Techniques de preuves
  - $\rightsquigarrow$ *preuve par cas, implication mutuelle, contraposition, absurde*
- Preuve en déduction naturelle
  - $\rightsquigarrow$ *cas de la logique propositionnelle et de la logique des prédicats*
- Méta-théorie
  - $\rightsquigarrow$ *définition inductive, fonction récursive, preuve par induction*
- Preuve d'égalité de termes par unification et application au principe de résolution
- Élimination des quantificateurs

Bibliographie francophone

Ces livres sont conseillés entre autres si vous voulez élargir votre horizon et considérer d'autres points de vue. Quand vous remarquez des divergences par rapport à ce qui est appris en cours, parlez-en avec votre enseignant !


I. Berlanger, V. Degauquier et T. Lucas :
Initiation à la logique formelle.
de boeck, 2014.


P. Lafourcade, M. Levy et S. Devismes :
Logique et démonstration automatique - Introduction à la logique propositionnelle et à la logique du premier ordre.
Ellipses, 2012.


F. Lepage :
Élements de logique contemporaine.
PU Montréal, 2010.

Bibliographie anglophone

Accès libre :

- Open Logic Project : `https://openlogicproject.org/`
  voir `https://builds.openlogicproject.org/` pour les PDF
  en particulier :
  - Sets, Logic, Computation : `https://slc.openlogicproject.org/`
  - Intermediate Logic : `https://builds.openlogicproject.org/courses/intermediate-logic/`
- Logic Matters : `https://www.logicmatters.net/`
  surtout : IFL : `https://www.logicmatters.net/ifl/`

Disponibles à la BU (aussi en ligne) :

- H.-D. Ebbinghaus, J. Flum, W. Thomas: *Mathematical Logic*. Springer, 2013
- D. van Dalen: *Logic and structure*. Springer, 2013

Pré-requis

Éléments vus auparavant :

- **UE Structures Discrètes 1** ayant permis d'acquérir les compétences suivantes :
  - Décrire comment la logique permet de modéliser des situations réelles
  - Convertir des énoncés informels en langage logique
  - Appliquer des méthodes (équivalences, résolution propositionnelle) aux problèmes de référence (SAT, conséquence logique, ...)
  - Décrire les forces et limitations de la logique propositionnelle et de la logique des prédicats
  - Utiliser un solveur pour résoudre des problèmes SAT
  - Preuves par induction

- **UE Ensembles 2** ?

Pré-requis

Si vous n'avez pas ces pré-requis...

- Étudier le support de SD1
- Lire les chapitres 1, 2, 3 (sauf la méthode des matrices) et 6 du livre *"Elements de Logique Contemporaine"* de F. Lepage mentionné dans la biblio et disponible à la BU

Plan

Introduction à la preuve

Définition de preuve (1/2)

Définition (selon wikipedia.fr)

Une **preuve** est un fait ou un raisonnement propre à établir *solidement* la vérité.

La preuve est un enjeu dans plusieurs grands domaines :

- le raisonnement en tant que processus cognitif est un concept-clé en philosophie ;
- en droit, la preuve est utilisée pour établir la vérité lors de procès. On évalue alors le degré de confiance d'une preuve :
  - Preuve parfaite : présomption irréfragable (preuve incontestable)
  - Preuve imparfaite : présomption simple
  - Présomption : faisceau d'éléments ou d'indices

Définition de preuve (2/2)

La preuve est un enjeu dans plusieurs grands domaines *(suite)* :

- en mathématiques, une preuve (ou *démonstration*) est une rédaction argumentée qui établit la véracité d'un énoncé mathématique en s'appuyant sur :
  - des **hypothèses** ;
  - des énoncés supposés évidents, appelés **axiomes** ;
  - des **énoncés précédemment démontrés**, et ;
  - des **règles de déduction**.

- en informatique, la correction d'un algorithme, d'un programme ou d'un système est démontrée afin de garantir sa **fiabilité** et sa **sûreté**.

Exemple de preuve en philosophie (1/2)

- Kurt Gödel est un logicien et mathématicien autrichien naturalisé américain du XXème siècle.
- Il établit une preuve dite *ontologique* de l'existence de Dieu dans le système de logique *modale*.
- Étant donnés 3 **définitions** et 5 **axiomes** :
  - **Définition 1** : *x est divin ssi x contient comme propriétés essentielles toutes les propriétés qui sont positives et seulement celles-ci*
  - ...
  - **Axiome 1** : *Toute propriété entraînée par une propriété positive est positive*
  - **Axiome 2** : *La propriété d'être divin est positive*
  - ...

Exemple de preuve en philosophie (2/2)

- On peut alors démontrer 4 **théorèmes** dont :
  - **Théorème 4** : *La propriété d'être divin est nécessairement exemplifiée*

- La preuve est correcte mais les axiomes sont critiquables.
  En particulier, en changeant un des axiomes, on peut aboutir à une preuve de l'inexistence de Dieu...

Référence : article wikipedia

Exemple de preuve en maths : démonstration directe

Définition (démonstration directe)

La démonstration directe consiste à démontrer la proposition énoncée en partant directement des hypothèses données et en arrivant à la conclusion par une série d'implications.

Exemple :

Théorème

Pour tout entier n, si n est impair alors n^2 est aussi impair.

Preuve : Si n est impair alors il peut s'écrire $n = 2k + 1$ où $k \in \mathbb{Z}$. Alors :

$$\begin{aligned} n^2 &= (2k+1)(2k+1) \\ &= 4k^2 + 4k + 1 \\ &= 2.(2k^2 + 2k) + 1 \\ &= 2k' + 1 \end{aligned}$$

avec $k' = 2k^2 + 2k$ et donc n^2 est impair. □

Ex. de preuve en maths : démonstration par disjonction de cas

Définition (démonstration par disjonction de cas)

Une démonstration par disjonction de cas consiste à montrer que l'énoncé se ramène à un certain nombre (fini) de cas distincts, puis à les démontrer séparément.

Exemple :

Théorème

Pour tout entier n, $\frac{n(n+1)}{2}$ est un entier.

Preuve :

Premier cas : si n est pair alors il existe un k tel que $n = 2k$ et $\frac{n(n+1)}{2} = k(2k+1)$ qui est entier.

Second cas : si n est impair alors il existe un k tel que $n = 2k+1$ et $\frac{n(n+1)}{2} = (2k+1)(k+1)$ qui est entier. □

Exemple de preuve en maths : démonstration par l'absurde

Définition (démonstration par l'absurde)

La démonstration par l'absurde consiste à supposer le contraire de la proposition énoncée et à montrer qu'on arrive alors à une contradiction (absurdité, impossibilité).

Exemple :

Théorème

Soient $a, b \geq 0$, si $\frac{a}{1+b} = \frac{b}{1+a}$ alors $a = b$.

Preuve : par l'absurde, supposons $a, b \geq 0$, $\frac{a}{1+b} = \frac{b}{1+a}$ et $a \neq b$. Alors : $a(1+a) = b(b+1)$ donc $a + a^2 = b + b^2$ d'où $a^2 - b^2 = b - a$.

Cela conduit à $(a-b)(a+b) = -(a-b)$. Comme $a \neq b$, on peut diviser par $(a-b)$ et $a + b = -1$. Or cela est impossible car la somme de deux nombres positifs a et b ne peut être négative. □

Exemple de preuve en maths : démonstration par contraposée

Définition (démonstration par contraposée)

Pour démontrer $P \to Q$, la démonstration par contraposée consiste à démontrer plutôt que $\neg Q \to \neg P$.

Exemple :

Théorème

Soit x un nombre réel tel que pour tout $\epsilon > 0$, on a $x \leq \epsilon$. Alors $x \leq 0$.

Preuve : la contraposée revient à démontrer :
$(x > 0) \to (\exists \epsilon.\ \epsilon > 0 \wedge x > \epsilon)$.

Cette implication est vraie puisque pour chaque $x > 0$ il suffit de prendre $\epsilon = \frac{x}{2}$. □

Remarque

Lorsqu'une démonstration inclut la construction d'un objet dont elle cherche à prouver l'existence, elle est dite constructive.

Exemple de preuve en maths

Remarque

Il existe aussi des démonstrations visant à prouver l'existence d'un objet mais qui sont non-constructives !

Exemple :

Théorème

Il existe des nombres irrationnels a et b tels que a^b est rationnel.

Preuve : on fait une disjonction de cas sur : "$\sqrt{2}^{\sqrt{2}}$ est rationnel".

Cas 1 : $\sqrt{2}^{\sqrt{2}}$ est rationnel. Dans ce cas, il suffit [1] de prendre $a = b = \sqrt{2}$.

Cas 2 : $\sqrt{2}^{\sqrt{2}}$ est irrationnel. Dans ce cas, on prend $a = \sqrt{2}^{\sqrt{2}}$ et $b = \sqrt{2}$. On voit que $a^b = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^{(\sqrt{2}.\sqrt{2})} = \sqrt{2}^2 = 2$. □

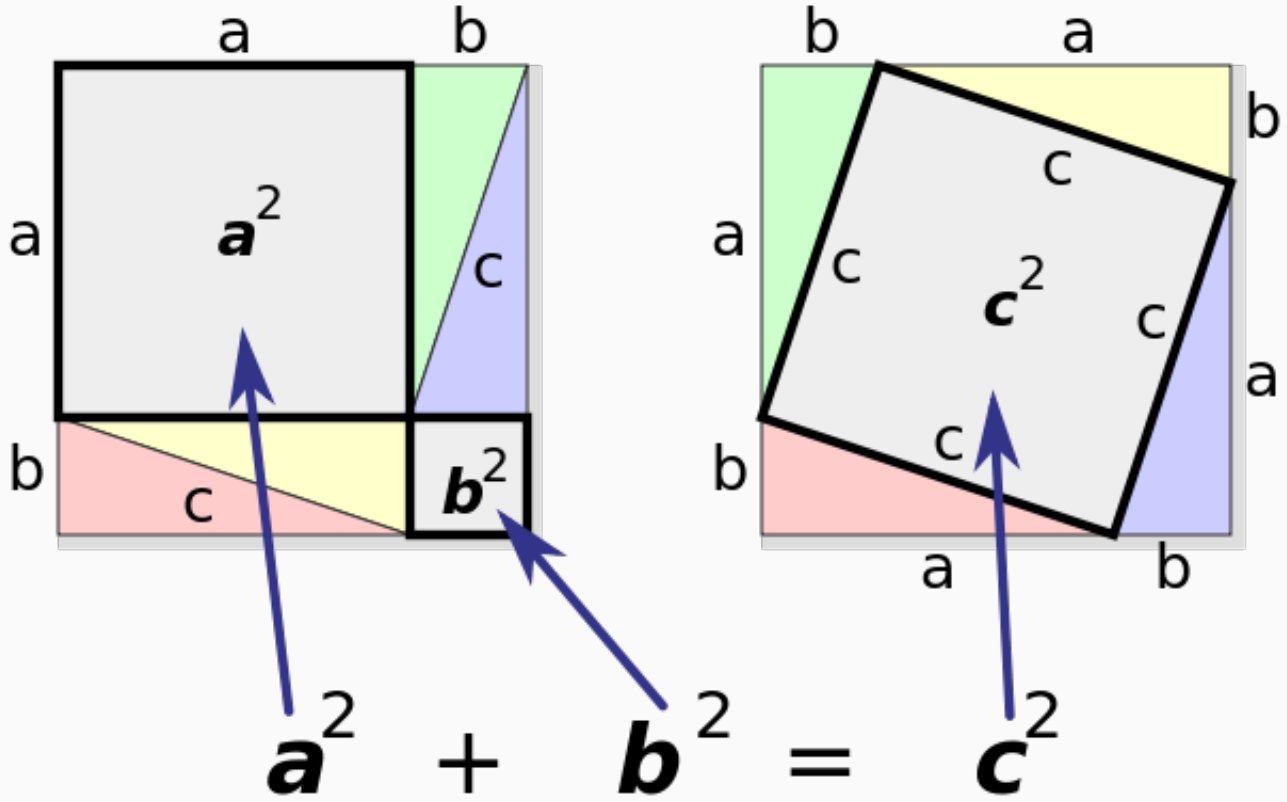
1. On admet ici que $\sqrt{2}$ est irrationnel ; ceci est facilement démontrable par l'absurde.

Exemple de preuve en maths : preuve sans mot

Définition (preuve sans mot)

Une preuve sans mot est une preuve que l'on fait par un diagramme qui la rend évidente.

Exemple : théorème de Pythagore



Exemple de preuve en maths : démonstration par contre-exemple

Définition (démonstration par contre-exemple)

Une démonstration par contre-exemple permet de réfuter un fait en exhibant un exemple qui contredit ce fait.

Exemple :

Conjecture de Fermat

Pour tout entier n, $F_n = 2^{2^n} + 1$ est un nombre premier.

Preuve : Euler a prouvé que cette conjecture est fausse en exhibant le contre-exemple suivant : $F_5 = 4\ 294\ 967\ 297$ est divisible par 641. □

Exemple de preuve en informatique (1/3)

Remarque

La preuve de correction de programme sera vue en détail dans l'UE de L2 **Algorithmique et Programmation**.

On suppose qu'on sait spécifier ce que **doit** faire le programme **indépendamment** de son écriture.

Exemple : un programme qui calcule la factorielle d'un entier positif n calcule le produit suivant :

$$n! = 1 \times 2 \times ... \times (n-1) \times n$$

Un programme réalisant ce calcul met en œuvre un algorithme calculant ce produit.

Exemple de preuve en informatique (2/3)

```
int fact(int n):
   int r = 1, i;
   for (i = 2; i <= n; i++)
      r = r * i
   return r
```

fact(4) : n $\leftarrow$ 4, r $\leftarrow$ 1

i $\leftarrow$ 2, r $\leftarrow$ 1*2 = 2

i $\leftarrow$ 3, r $\leftarrow$ 2*3 = 6

i $\leftarrow$ 4, r $\leftarrow$ 6*4 = 24

i $\leftarrow$ 5

return 24

Comment prouver que le résultat est correct pour tout n ?

Exemple de preuve en informatique (3/3)

Vérification via des assertions et la logique de Hoare (calcul de wp) :

```
 {n ≥ 1} // précondition
int fact(int n):
int r = 1, i = 1;
 {r = i! } // invariant (correction)
 {n − i } // variant (terminaison)
for (i = 2; i <= n; i++) {
  r = r * i
}
return r
 {fact(n) = n!} // postcondition
```

Questions cruciales sur la preuve

- Ces techniques de preuve sont-elles correctes ?
- Peut-on tout prouver avec ?
- Comment les mécaniser ? Peut-on les automatiser ?

A-t'on déjà fait des preuves en "Logique 1" ?

Rappels !

- Vous devez savoir
  - prouver qu'une grille de sudoku admet une solution ;
  - prouver qu'on peut colorer une carte avec 4 couleurs ;
  - prouver qu'il est possible de planifier un ensemble de tâches contraintes ;
  - ...
  - plus généralement, prouver $H \models C$ (**conséquence logique**).
- Le problème précédent est équivalent à l'**insatisfiabilité** d'un ensemble de formules
- Technique de preuve efficace : la **résolution**

Fin de l'introduction : qu'est-ce qui est exigible à l'examen ?

Check-list des attendus à l'examen :

- ☐ Savoir rédiger une preuve correcte par disjonction de cas
- ☐ Savoir rédiger une preuve correcte par l'absurde
- ☐ Savoir rédiger une preuve correcte par contraposée

... pour des énoncés simples (issus de la théorie des ensembles, par exemple)

Preuve en déduction naturelle

Plan

Établir la validité d'un raisonnement (rappel)

Remarque : ceci est un rappel de "Logique 1"

- Un raisonnement du type

  Quand il neige, il fait froid et quand il y a du verglas, il fait froid. Or aujourd'hui, il neige ou il y a du verglas. De plus, en été, il ne fait pas froid. Donc, on n'est pas été.

- ... peut être modélisé sous forme de la **conséquence logique** :

$$\{n \longrightarrow f, v \longrightarrow f, n \vee v, e \longrightarrow \neg f\} \models \neg e$$

  où :

  - n : "il neige"
  - f : "il fait froid"
  - v : "il y a du verglas"
  - e : "on est en été"

Établir la validité d'un raisonnement (rappel)

- Établir la **validité** d'un raisonnement, c'est répondre à la question :

$$H \models C \,?$$

  c'est-à-dire, $H \cup \{\neg C\}$ insatisfiable ?

- On a vu en "Logique 1" que la réponse pouvait être obtenue :
  - en construisant des tables de vérité, ou
  - en appliquant le principe de résolution.

Établir la validité d'un raisonnement (rappel)

- Établir la **validité** d'un raisonnement, c'est répondre à la question :

$$H \models C \,?$$

  c'est-à-dire, $H \cup \{\neg C\}$ insatisfiable ?

- On a vu en "Logique 1" que la réponse pouvait être obtenue :
  - en construisant des tables de vérité, ou
  - en appliquant le principe de résolution.
- Pas satisfaisant pour des tâches exigeant un détail du raisonnement (ou explication) qui mène de H à C
  - Diagnostic médical
  - Diagnostic de panne
  - Tuteur électronique
  - ...

Conséquence et dérivabilité

Conséquence logique : $\{H_1, \ldots, H_n\} \models C$

- une notion **sémantique**
- définie à l'aide d'**interprétations :**
  toute interprétation qui satisfait $\{H_1, \ldots, H_n\}$ satisfait aussi C

Derivabilité : $\{H_1, \ldots, H_n\} \vdash C$

- une notion **syntaxique**
- relative à un **calcul :** avec les règles du calcul, on peut déduire C à partir des hypothèses $\{H_1, \ldots, H_n\}$
- ici, le calcul est la **déduction naturelle**

Critères pour un "bon" calcul :

- Correction : si $\{H_1, \ldots, H_n\} \vdash C$ alors $\{H_1, \ldots, H_n\} \models C$
- Complétude : si $\{H_1, \ldots, H_n\} \models C$ alors $\{H_1, \ldots, H_n\} \vdash C$

Cas spéciaux : Validité : $\models C$, resp. prouvabilité : $\vdash C$

Déduction naturelle : exemple (1)

Exemple de raisonnement “naturel”

1. Quand il neige, il fait froid : $(n \longrightarrow f)$
2. Quand il y a du verglas, il fait froid : $(v \longrightarrow f)$
3. Il neige ou il y a du verglas $(n \vee v)$
4. En été, il ne fait pas froid $(e \longrightarrow \neg f)$
5. Donc, on n’est pas été : $\neg e$

Déduction naturelle : exemple (1)

Exemple de raisonnement "naturel"

1. Quand il neige, il fait froid : $(n \longrightarrow f)$
2. Quand il y a du verglas, il fait froid : $(v \longrightarrow f)$
3. Il neige ou il y a du verglas $(n \vee v)$
4. En été, il ne fait pas froid $(e \longrightarrow \neg f)$
5. Donc, on n'est pas été : $\neg e$

Dérivation de la conclusion (5) *à partir des hypothèses* (1) *, ...,* (4) *:*

6. Pour montrer (5) , supposons e, et dérivons une contradiction $(\perp)$
7. Distinction de cas (avec (3)) :

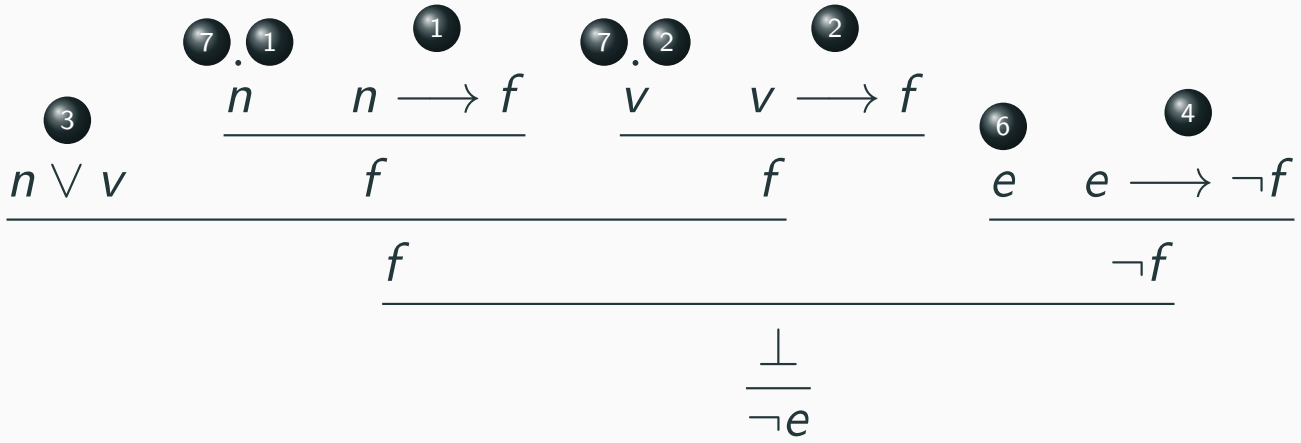
   7.1 Soit n. Alors, avec (1) , on a f.

   7.2 Soit v. Alors, avec (2) , on a f.

   Donc, de (1) , (2) , (3) on peut conclure f.
8. Avec (6) et (4) , inférer $\neg f$.
9. (7) et (8) permettent de conclure $\perp$, comme demandé dans (6) .

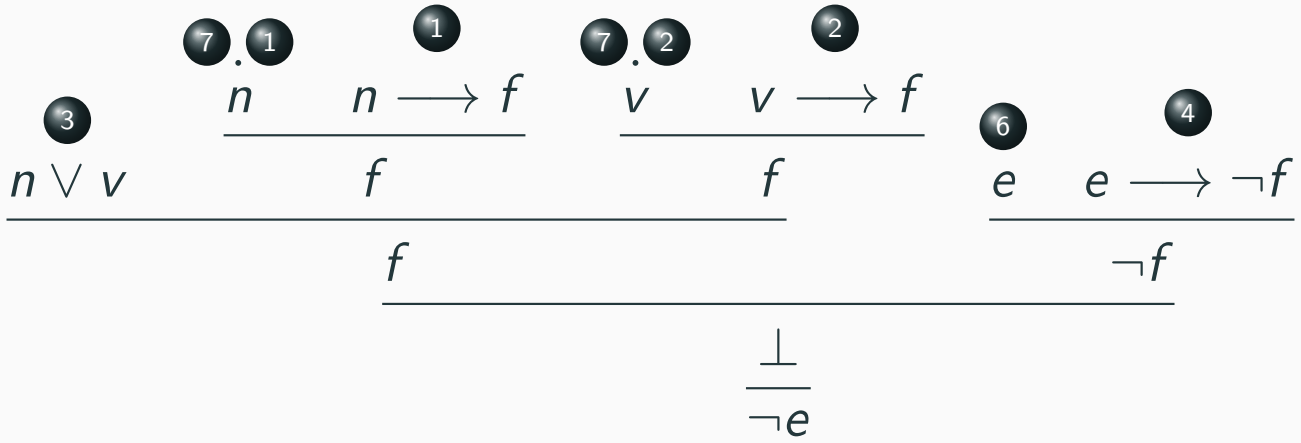
Déduction naturelle : exemple (2)

Arbre de dérivation :



Déduction naturelle : exemple (2)

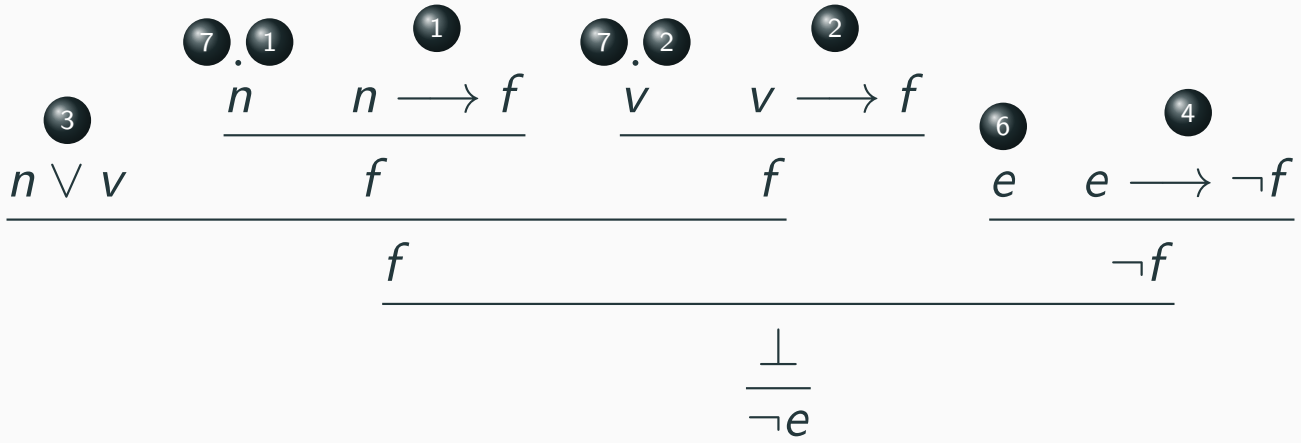
Arbre de dérivation :



Nous dirons : sous les hypothèses 1 ... 4 , on peut *déduire* (ou *dériver*) la *conclusion* 5 .

Déduction naturelle : exemple (2)

Arbre de dérivation :



Nous dirons : sous les hypothèses 1 ... 4 , on peut *déduire* (ou *dériver*) la *conclusion* 5 .

Ce fait est exprimé par le **séquent** (**jugement**) suivant :
$\{n \longrightarrow f, v \longrightarrow f, n \vee v, e \longrightarrow \neg f\} \vdash \neg e$

Déduction naturelle : première règle

Dans **tous les cas** (pour tout langage que nous pourrions considérer), nous aurons la règle suivante :

Axiome :

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ (Ax)$$

Déduction naturelle : première règle

Dans **tous les cas** (pour tout langage que nous pourrions considérer), nous aurons la règle suivante :

Axiome :

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ (Ax)$$

Notation : dans un arbre de dérivation, on n'écrit pas explicitement la précondition $A \in \Gamma$ car c'est inutile, ça se voit :

$$\frac{}{\text{☕}, \text{🚲} \vdash \text{🚲}}\ (Ax)$$ est correct, mais

$$\frac{}{\text{☕}, \text{🚲} \vdash \text{⚽}}\ (Ax)$$ n'est pas correct.

Déduction naturelle : un exemple abstrait

Nous avons déjà donné un exemple *intuitif* (neige, verglas, etc).

Pour mieux apprécier que la déduction est un processus **entièrement mécanique** et que nous ne pouvons pas utiliser notre intuition sémantique comme bon nous semble, il est utile de présenter un exemple **abstrait**, sans aucune intuition.

Déduction naturelle : un exemple abstrait

Nous avons déjà donné un exemple *intuitif* (neige, verglas, etc).

Pour mieux apprécier que la déduction est un processus **entièrement mécanique** et que nous ne pouvons pas utiliser notre intuition sémantique comme bon nous semble, il est utile de présenter un exemple **abstrait**, sans aucune intuition.

- Langage $\mathcal{L} = \{\heartsuit, \clubsuit, \spadesuit, \diamondsuit\}$.

Déduction naturelle : un exemple abstrait

Nous avons déjà donné un exemple *intuitif* (neige, verglas, etc).

Pour mieux apprécier que la déduction est un processus **entièrement mécanique** et que nous ne pouvons pas utiliser notre intuition sémantique comme bon nous semble, il est utile de présenter un exemple **abstrait**, sans aucune intuition.

- Langage $\mathcal{L} = \{\heartsuit, \clubsuit, \spadesuit, \diamondsuit\}$.
- Système déductif donné par les *règles* :

$$\frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \spadesuit}\ (\beta) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \diamondsuit \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

Déduction naturelle : un exemple abstrait

Nous avons déjà donné un exemple *intuitif* (neige, verglas, etc).

Pour mieux apprécier que la déduction est un processus **entièrement mécanique** et que nous ne pouvons pas utiliser notre intuition sémantique comme bon nous semble, il est utile de présenter un exemple **abstrait**, sans aucune intuition.

- Langage $\mathcal{L} = \{\heartsuit, \clubsuit, \spadesuit, \diamondsuit\}$.
- Système déductif donné par les *règles* :

$$\frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \spadesuit}\ (\beta) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \diamondsuit \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

- α : si vous avez un $\diamondsuit$ je vous donne un $\clubsuit$

Déduction naturelle : un exemple abstrait

Nous avons déjà donné un exemple *intuitif* (neige, verglas, etc).

Pour mieux apprécier que la déduction est un processus **entièrement mécanique** et que nous ne pouvons pas utiliser notre intuition sémantique comme bon nous semble, il est utile de présenter un exemple **abstrait**, sans aucune intuition.

- Langage $\mathcal{L} = \{\heartsuit, \clubsuit, \spadesuit, \diamondsuit\}$.
- Système déductif donné par les *règles* :

$$\frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \spadesuit}\ (\beta) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \diamondsuit \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

- α : si vous avez un $\diamondsuit$ je vous donne un $\clubsuit$
- γ : si vous avez un $\clubsuit$ et un $\spadesuit$ je vous donne un $\heartsuit$

Déduction naturelle : un exemple abstrait

Nous avons déjà donné un exemple *intuitif* (neige, verglas, etc).

Pour mieux apprécier que la déduction est un processus **entièrement mécanique** et que nous ne pouvons pas utiliser notre intuition sémantique comme bon nous semble, il est utile de présenter un exemple **abstrait**, sans aucune intuition.

- Langage $\mathcal{L} = \{\heartsuit, \clubsuit, \spadesuit, \diamondsuit\}$.
- Système déductif donné par les *règles* :

$$\frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \spadesuit}\ (\beta) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \diamondsuit \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

- α : si vous avez un $\diamondsuit$ je vous donne un $\clubsuit$
- γ : si vous avez un $\clubsuit$ et un $\spadesuit$ je vous donne un $\heartsuit$
- δ : si vous savez obtenir un $\heartsuit$ à partir d'un $\diamondsuit$, je vous donne un $\heartsuit$

Preuve de ♥

Les règles :

$$\frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \clubsuit}\ (\alpha) \quad \frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \spadesuit}\ (\beta) \quad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \quad \frac{\Gamma, \diamondsuit \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La preuve :

$$\frac{}{\diamondsuit \vdash \diamondsuit}\ (Ax)$$

Ax

Preuve de ♥

Les règles :

$$\frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \spadesuit}\ (\beta) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \blacklozenge \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La preuve :

$$\dfrac{\dfrac{\dfrac{}{\blacklozenge \vdash \blacklozenge}\,(Ax)}{\blacklozenge \vdash \clubsuit}\,(\alpha) \qquad \dfrac{}{}}{\dfrac{}{}}$$

Ax

On applique α.

Preuve de ♥

Les règles :

$$\frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \clubsuit}\ (\alpha)\quad \frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \spadesuit}\ (\beta)\quad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma)\quad \frac{\Gamma, \blacklozenge \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La preuve :

$$\dfrac{\dfrac{\dfrac{}{\blacklozenge \vdash \blacklozenge}\,(Ax)}{\blacklozenge \vdash \clubsuit}\,(\alpha) \qquad \dfrac{\dfrac{}{\blacklozenge \vdash \blacklozenge}\,(Ax)}{\blacklozenge \vdash \spadesuit}\,(\beta)}{\dfrac{}{}}$$

Ax

On applique α.

De la même façon avec β.

Preuve de ♥

Les règles :

$$\frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \clubsuit}\ (\alpha)\quad \frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \spadesuit}\ (\beta)\quad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma)\quad \frac{\Gamma, \blacklozenge \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La preuve :

$$\dfrac{\dfrac{\dfrac{}{\blacklozenge \vdash \blacklozenge}\,(Ax)}{\blacklozenge \vdash \clubsuit}\,(\alpha) \qquad \dfrac{\dfrac{}{\blacklozenge \vdash \blacklozenge}\,(Ax)}{\blacklozenge \vdash \spadesuit}\,(\beta)}{\blacklozenge \vdash \heartsuit}\,(\gamma)$$

Ax

On applique α.

De la même façon avec β.

On applique γ.

Preuve de ♥

Les règles :

$$\frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \clubsuit}\ (\alpha)\quad \frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \spadesuit}\ (\beta)\quad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma)\quad \frac{\Gamma, \blacklozenge \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La preuve :

$$\dfrac{\dfrac{\dfrac{}{\blacklozenge \vdash \blacklozenge}\,(Ax)}{\blacklozenge \vdash \clubsuit}\,(\alpha) \qquad \dfrac{\dfrac{}{\blacklozenge \vdash \blacklozenge}\,(Ax)}{\blacklozenge \vdash \spadesuit}\,(\beta)}{\blacklozenge \vdash \heartsuit}\,(\gamma)$$
$$\dfrac{\blacklozenge \vdash \heartsuit}{\vdash \heartsuit}\,(\delta)$$

Ax

On applique α.

De la même façon avec β.

On applique γ.

On applique δ. Notez que l'hypothèse ♦ a disparu. On appelle une telle dérivation (sans hypothèses) une **preuve**.

Une dérivation avec hypothèses

Autre système de règles (sans β !) :

$$\frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \blacklozenge \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La dérivation :

Sauriez-vous obtenir un ♥ avec seulement des ♠ ?

Une dérivation avec hypothèses

Autre système de règles (sans β !) :

$$\frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \diamondsuit \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La dérivation :

$$\dfrac{\dfrac{}{\spadesuit, \diamondsuit \vdash \diamondsuit}\ (Ax)}{\qquad\qquad\qquad\qquad}$$

$$\qquad\qquad$$

Sauriez-vous obtenir un $\heartsuit$ avec seulement des $\spadesuit$?

Par axiome.

Une dérivation avec hypothèses

Autre système de règles (sans β !) :

$$\frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \blacklozenge \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La dérivation :

$$\dfrac{\dfrac{\dfrac{}{\spadesuit, \blacklozenge \vdash \blacklozenge}\ (Ax)}{\spadesuit, \blacklozenge \vdash \clubsuit}\ (\alpha)}{\qquad\qquad}$$

Sauriez-vous obtenir un $\heartsuit$ avec seulement des $\spadesuit$?

Par axiome.

On applique α.

Une dérivation avec hypothèses

Autre système de règles (sans β !) :

$$\frac{\Gamma \vdash \spadesuit}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \spadesuit \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La dérivation :

$$\dfrac{\dfrac{\dfrac{}{\spadesuit, \spadesuit \vdash \spadesuit}\ (Ax)}{\spadesuit, \spadesuit \vdash \clubsuit}\ (\alpha) \qquad \dfrac{}{\spadesuit, \spadesuit \vdash \spadesuit}\ (Ax)}{\dfrac{}{}}$$

Sauriez-vous obtenir un $\heartsuit$ avec seulement des $\spadesuit$?

Par axiome.

On applique α.

Par axiome.

Une dérivation avec hypothèses

Autre système de règles (sans β !) :

$$\frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \blacklozenge \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La dérivation :

$$\dfrac{\dfrac{\dfrac{}{\spadesuit, \blacklozenge \vdash \blacklozenge}\,(Ax)}{\spadesuit, \blacklozenge \vdash \clubsuit}\,(\alpha) \qquad \dfrac{}{\spadesuit, \blacklozenge \vdash \spadesuit}\,(Ax)}{\spadesuit, \blacklozenge \vdash \heartsuit}\,(\gamma)$$

Sauriez-vous obtenir un $\heartsuit$ avec seulement des $\spadesuit$?

Par axiome.

On applique α.

Par axiome.

On applique γ.

Une dérivation avec hypothèses

Autre système de règles (sans β !) :

$$\frac{\Gamma \vdash \blacklozenge}{\Gamma \vdash \clubsuit}\ (\alpha) \qquad \frac{\Gamma \vdash \clubsuit \quad \Gamma \vdash \spadesuit}{\Gamma \vdash \heartsuit}\ (\gamma) \qquad \frac{\Gamma, \blacklozenge \vdash \heartsuit}{\Gamma \vdash \heartsuit}\ (\delta)$$

La dérivation :

$$\dfrac{\dfrac{\dfrac{}{\spadesuit, \blacklozenge \vdash \blacklozenge}\,(Ax)}{\spadesuit, \blacklozenge \vdash \clubsuit}\,(\alpha) \qquad \dfrac{}{\spadesuit, \blacklozenge \vdash \spadesuit}\,(Ax)}{\dfrac{\spadesuit, \blacklozenge \vdash \heartsuit}{\spadesuit \vdash \heartsuit}\,(\delta)}\,(\gamma)$$

Sauriez-vous obtenir un ♥ avec seulement des ♠ ?

Par axiome.

On applique α.

Par axiome.

On applique γ.

On applique δ. Notez que l'hypothèse ♦ a disparu. Nous avons une dérivation de ♥ à partir de ♠.

Règles d'inférence

Définition (règle)

Une règle

$$\frac{J^1 \quad \dots \quad J^m}{J}$$

est composée de :

- 0, 1 ou plusieurs **antécédents** : J^1, ..., J^m
- un seul **conséquent** : J

Lecture informelle :
"Si tous les antécédents sont prouvables, alors aussi le conséquent"

Chaque J et J^i a la forme d'un séquent $\{H_1, \dots, H_n\} \vdash C$

Souvent, l'ensemble d'hypothèses est abrégé par Γ

- On écrit Γ, A pour $\Gamma \cup \{A\}$
- On écrit $\vdash A$ pour $\{\} \vdash A$

Plan

Logique minimale

Afin de se concentrer sur les concepts de la preuve en déduction naturelle, on va considérer une logique simplissime : la logique minimale.

Elle est constituée d'un seul opérateur : l'**implication** $\longrightarrow$.

La logique propositionnelle et la logique des prédicats peuvent être vues comme des *extensions* de la logique minimale.

Syntaxe de la logique minimale

Définition (formules de la logique minimale) :

Soient $PROP$ un ensemble de variables propositionnelles. L'ensemble L_{min} des formules de la logique minimale est le plus petit ensemble qui satisfait les conditions suivantes :

1. **Variable prop.** : pour tout $p \in PROP$, $p \in L_{min}$
2. **Implication** : si $A, B \in L_{min}$ alors $(A \longrightarrow B) \in L_{min}$

Exemple : soit $PROP = \{p, q\}$, ces formules appartiennent à L_{min}

- $((p \longrightarrow q) \longrightarrow p)$
- $(p \longrightarrow p)$
- p

Convention syntaxique

On pourra enlever des parenthèses inutiles en considérant que l'opérateur $\longrightarrow$ est associatif à droite.

Exemple :

$$(p \longrightarrow (q \longrightarrow r)) \equiv p \longrightarrow q \longrightarrow r$$

par contre,

$$((p \longrightarrow q) \longrightarrow r) \not\equiv p \longrightarrow q \longrightarrow r$$

Sémantique de la logique minimale

(les définitions ci-dessous sont similaires à celles étudiées en "Logique 1")

Définitions (sémantique de la logique minimale)

- Une valuation est une fonction $v : PROP \rightarrow \{0, 1\}$.
- À partir d'une valuation v, son *extension sur* L_{min} (c'est-à-dire, $v : L_{min} \rightarrow \{0, 1\}$), est définie ainsi :
  - (**Variable :** $v(p) = v(p)$ "la valeur de la formule p correspond à la valeur de la variable p")
  - **Implication :** $v(A \longrightarrow B) = max(1 - v(A), v(B))$
- Une valuation v est un modèle d'une formule A si $v(A) = 1$
- $\{H_1, ..., H_n\} \models C$ ssi tout modèle **commun** à H_1, ..., H_n est aussi un modèle de C

Preuve en logique minimale

Problème considéré :
On cherche à établir à partir d'un ensemble Γ d'hypothèses la conclusion C, noté $\Gamma \vdash C$, où :

- $\Gamma = \{H_1, \ldots, H_n\}$ et pour tout i, H_i est une formule de L_{min} ;
- C est une formule de L_{min}.

... à l'aide d'un calcul défini par des règles d'inférence.

Les étapes de calcul permettant d'établir le séquent $\Gamma \vdash C$ sont appelées une dérivation. Si Γ est vide, on parle alors de preuve.

Règles d'inférences pour la dérivation en logique minimale

On s'est déjà donné une première règle :

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ (Ax)$$

On va maintenant définir 2 règles supplémentaires...

Élimination de l'implication : le Modus Ponens

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

Cette règle se lit :

Si on peut dériver les séquents $\Gamma \vdash A \longrightarrow B$ et $\Gamma \vdash A$
alors on peut dériver le séquent $\Gamma \vdash B$.

Elle est appelée Modus Ponens ou règle d'élimination de l'implication.

Utilisation correcte d'une règle

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

- A et B sont des **méta**-variables
- Attention ! l'application d'une règle requiert que chaque occurrence d'une méta-variable corresponde à la même formule ou contexte.

Exemple :

Utilisation correcte d'une règle

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

- A et B sont des **méta**-variables
- Attention ! l'application d'une règle requiert que chaque occurrence d'une méta-variable corresponde à la même formule ou contexte.

Exemple :

$$\frac{p \longrightarrow q, r \longrightarrow p \vdash\ p \longrightarrow q \qquad r, r \longrightarrow p \vdash\ p}{p \longrightarrow q, r \longrightarrow p \vdash\ q}$$

... n'est pas une instance de $(E \longrightarrow)$.

Utilisation correcte d'une règle

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

- A et B sont des **méta**-variables
- Attention ! l'application d'une règle requiert que chaque occurrence d'une méta-variable corresponde à la même formule ou contexte.

Exemple :

$$\frac{p \longrightarrow q, r \longrightarrow p \vdash\ p \longrightarrow q \qquad p \longrightarrow q, r \longrightarrow p \vdash\ r}{p \longrightarrow q, r \longrightarrow p \vdash\ q}$$

... n'est pas une instance de $(E \longrightarrow)$.

Utilisation correcte d'une règle

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

- A et B sont des **méta**-variables
- Attention ! l'application d'une règle requiert que chaque occurrence d'une méta-variable corresponde à la même formule ou contexte.

Exemple :

$$\frac{p \longrightarrow q, r \longrightarrow p \vdash p \longrightarrow q \qquad p \longrightarrow q, r \longrightarrow p \vdash p}{p \longrightarrow q, r \longrightarrow p \vdash q}$$

... est une instance de $(E \longrightarrow)$.

Exemple

On pose $\Gamma = \{p \longrightarrow q, q \longrightarrow r, p\}$. On dérive (= on construit un *arbre de dérivation* pour) $\Gamma \vdash r$:

$$\dfrac{\dfrac{}{\Gamma \vdash q \longrightarrow r}\,(Ax) \quad \dfrac{\dfrac{}{\Gamma \vdash p \longrightarrow q}\,(Ax) \quad \dfrac{}{\Gamma \vdash p}\,(Ax)}{\Gamma \vdash q}\,(E \longrightarrow)}{\Gamma \vdash r}\,(E \longrightarrow)$$

Introduction de l'implication

On ajoute la règle supplémentaire suivante :

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

Cette règle formalise le mode de raisonnement naturel suivant :

*P*our dériver $A \longrightarrow B$, on suppose d'abord A.
Sous cette hypothèse, on dérive B.

Elle est appelée règle d'introduction de l'implication.

Exemple

On pose $\Gamma = \{p \longrightarrow q, q \longrightarrow r\}$. On dérive $\Gamma \vdash p \longrightarrow r$:

$$\dfrac{\dfrac{}{\Gamma, p \vdash q \longrightarrow r}\ (Ax) \quad \dfrac{\dfrac{}{\Gamma, p \vdash p \longrightarrow q}\ (Ax) \quad \dfrac{}{\Gamma, p \vdash p}\ (Ax)}{\Gamma, p \vdash q}\ (E \longrightarrow)}{\dfrac{\Gamma, p \vdash r}{\Gamma \vdash p \longrightarrow r}\ (I \longrightarrow)}\ (E \longrightarrow)$$

Vocabulaire

Définitions

- Une dérivation est une suite d'applications de règles d'inférences.
- Un séquent est dérivable s'il existe une dérivation concluant par ce séquent.
- Une règle est dite d'introduction si elle fait apparaître un connecteur dans le conséquent.
- Une règle est dite d'élimination si elle fait apparaître un connecteur dans un antécédent.

Remarque : ces définitions sont valables dans toute la déduction naturelle, indépendamment de la logique considérée.

Règle dérivée

- Il se peut qu'un enchaînement de règles revienne souvent dans des dérivations de séquents. On parle alors de stratégie de preuve/dérivation.
- Une nouvelle règle dite dérivée peut être définie à partir des règles existantes afin de modéliser cette stratégie.
- Une règle dérivée n'apporte rien d'un point de vue "prouvabilité" mais elle permet de raccourcir les dérivations.

*E*xemple : la règle de coupure suivante...

Règle de coupure

$$\frac{\Gamma \vdash B \qquad \Gamma, B \vdash A}{\Gamma \vdash A}\ (cut)$$

- Cette règle correspond à la pratique dans laquelle un lemme intermédiaire B est introduit pour prouver A.
- Cette règle est dérivée à partir des existantes de la façon suivante :

$$\dfrac{\Gamma \vdash B \qquad \dfrac{\Gamma, B \vdash A}{\Gamma \vdash B \longrightarrow A}\ (I \longrightarrow)}{\Gamma \vdash A}\ (E \longrightarrow)$$

- Difficulté d'utilisation : il faut trouver le bon B !

Validité vs. prouvabilité

On a vu 2 notions différentes de *vérité* pour un séquent :

- une vérité *sémantique* : $\Gamma \models C$
- une vérité *syntaxique* : $\Gamma \vdash C$

$\rightsquigarrow$ **Question :** peut-on les comparer ?

Théorème de correction

Les règles de déduction de la logique minimale ne permettent de dériver que des séquents valides : si $\Gamma \vdash C$ alors $\Gamma \models C$.

Preuve : la preuve du théorème se fait par induction sur la structure de l'arbre de dérivation associé au séquent $\square$

Induction sur la structure d'un arbre

L'ensemble des *arbres de dérivation* est effectivement un ensemble défini par *induction* :

- Pour toute séquence finie de formules Γ et pour tout $A \in \Gamma$,

$$\frac{}{\Gamma \vdash A}\ (Ax)$$

  est un arbre de dérivation.

Induction sur la structure d'un arbre

L'ensemble des *arbres de dérivation* est effectivement un ensemble défini par *induction* :

- Pour toute séquence finie de formules Γ et pour tout $A \in \Gamma$,

$$\frac{}{\Gamma \vdash A}\ (Ax)$$

  est un arbre de dérivation.
- Si

$$\frac{\boxed{\mathsf{T}}}{\Gamma, A \vdash B}$$

  est un arbre de dérivation, alors

$$\frac{\dfrac{\boxed{\mathsf{T}}}{\Gamma, A \vdash B}}{\Gamma \vdash A \longrightarrow B}\ (I \longrightarrow)$$

  est un arbre de dérivation.
- ... (règle $(E \longrightarrow)$: exercice)

Preuve de correction (1)

- Soit une dérivation obtenue par la règle d'axiome (Ax) :

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ (Ax)$$

  Par construction, la proposition A appartient au contexte Γ. Donc, toute valuation v qui satisfait les formules dans Γ satisfait A, et donc $\Gamma \models A$.

Preuve de correction (2)

- Soit une dérivation se terminant par une règle d'élimination $(E \longrightarrow)$:

$$\frac{\begin{array}{cc}\vdots & \vdots \\ \Gamma \vdash A \longrightarrow B & \Gamma \vdash A\end{array}}{\Gamma \vdash B}\ (E \longrightarrow)$$

Hypothèse d'induction : on suppose $\Gamma \models A \longrightarrow B$ et $\Gamma \models A$.

On doit montrer : $\Gamma \models B$.

Soit v une valuation qui rend vraies toutes les formules dans Γ. Par hypothèse d'induction, on a $v(A) = 1$ et $v(A \longrightarrow B) = 1$. Par conséquent, on a $v(B) = 1$ et $\Gamma \models B$.

Preuve de correction (3)

- Soit une dérivation obtenue par la règle d'introduction $(I \longrightarrow)$:

$$\frac{\begin{array}{c}\vdots \\ \Gamma, A \vdash B\end{array}}{\Gamma \vdash A \longrightarrow B}\ (I \longrightarrow)$$

Hypothèse d'induction : on suppose $\Gamma, A \models B$.

On doit montrer : $\Gamma \models A \longrightarrow B$.

Soit v une valuation satisfaisant toutes les formules dans Γ. Deux cas sont possibles :

- Si $v(A) = 1$ alors v satisfait toutes les formules dans Γ ainsi que A et par conséquent par l'hypothèse d'induction, $v(B) = 1$. D'où $v(A \longrightarrow B) = 1$.
- Si $v(A) = 0$ alors $v(A \longrightarrow B) = 1$.

Dans les deux cas, $v(A \longrightarrow B) = 1$ et donc $\Gamma \models A \longrightarrow B$. $\square$

Règle admissible

Des règles *admissibles* sont des règles que l'on peut rajouter au système de la déduction naturelle en préservant la correction, et qui ne sont pourtant pas dérivables.

Règle admissible

Des règles *admissibles* sont des règles que l'on peut rajouter au système de la déduction naturelle en préservant la correction, et qui ne sont pourtant pas dérivables.

Exemple : *affaiblissement* :

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ (\text{Aff})$$

On peut prouver par induction que toute dérivation qui utilise l'affaiblissement peut être traduit en une dérivation qui ne l'utilise pas.

Validité vs. prouvabilité

On a vu 2 notions différentes de *vérité* pour un séquent :

- une vérité *sémantique* : $\Gamma \models C$
- une vérité *syntaxique* : $\Gamma \vdash C$

$\rightsquigarrow$ **Question :** peut-on les comparer ?

Théorème

Il existe au moins un séquent valide non-dérivable dans le calcul défini pour la logique minimale.

Preuve : (partielle) considérez la formule de Peirce :

$$((p \longrightarrow q) \longrightarrow p) \longrightarrow p$$

c'est une tautologie mais : $\vdash ((p \longrightarrow q) \longrightarrow p) \longrightarrow p$ n'a pas de preuve dans le calcul précédent... $\square$

Plan

Cas de la logique propositionnelle

Maintenant, on considère la logique propositionnelle au lieu de la logique minimale.

Les concepts liés à la déduction naturelle introduits dans la section précédente sont conservés :

- séquent $\vdash$, séquent prouvable ;
- règle d'inférence, règle dérivée ;
- arbre de dérivation, dérivation, preuve, ...

On conserve aussi les 3 règles vues :

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ (Ax) \qquad \frac{\Gamma \vdash A \longrightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ (E \longrightarrow) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \longrightarrow B}\ (I \longrightarrow)$$

... et on va ajouter des règles d'élimination et d'introduction pour $\bot$ et les connecteurs $\neg$, $\wedge$ et $\vee$.

Règles pour $\wedge$

- Règles d'élimination :

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ (E\wedge_1) \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ (E\wedge_2)$$

- Règle d'introduction :

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

Règles pour $\neg$

- Règles d'élimination :

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

- Règle d'introduction :

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

Remarque : ce sont des cas spéciaux des règles pour $\longrightarrow$ en considérant que $\neg A$ n'est qu'une abbréviation pour $A \longrightarrow \bot$.

Exercice

Montrer : $\vdash \neg(p \wedge q) \longrightarrow \neg(q \wedge p)$

Règles pour $\bot$

- Règle d'élimination :

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ (E\bot)$$

- Règle d'introduction ? Il n'y en a pas !

Remarque : $(E\bot)$ est aussi appelée *ex falso quod libet* ou encore *ex contradictione sequitur quod libet* ou encore Principe de Pseudo Scotus (logicien médiéval).

Exercice

Montrer : $\vdash \neg(p \longrightarrow q) \longrightarrow \neg p \longrightarrow r$

Règles pour $\vee$

- Règle d'élimination :

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

- Règle d'introduction :

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ (I\vee_1) \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ (I\vee_2)$$

Remarque : $(E\vee)$ correspond à une distinction de cas.

Retour sur l'exemple introductif

Soit $\Gamma = \{n \longrightarrow f, v \longrightarrow f, n \vee v, e \longrightarrow \neg f\}$, $\Gamma' = \Gamma \cup \{e\}$

Arbres de dérivation :

Soit $[A_1] =$

$$\dfrac{\dfrac{}{\Gamma' \vdash n \vee v}\ (Ax) \quad \dfrac{\dfrac{}{\Gamma', n \vdash n \longrightarrow f}\ (Ax) \quad \dfrac{}{\Gamma', n \vdash n}\ (Ax)}{\Gamma', n \vdash f}\ (E \rightarrow) \quad \dfrac{\dfrac{}{\Gamma', v \vdash v \longrightarrow f}\ (Ax) \quad \dfrac{}{\Gamma', v \vdash v}\ (Ax)}{\Gamma', v \vdash f}\ (E \rightarrow)}{\Gamma' \vdash f}\ (E\vee)$$

Alors :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma' \vdash e \longrightarrow \neg f}\ (Ax) \quad \dfrac{}{\Gamma' \vdash e}\ (Ax)}{\Gamma' \vdash \neg f}\ (E \rightarrow) \quad [A_1]}{\Gamma' \vdash \bot}\ (E\neg)}{\Gamma \vdash \neg e}\ (I\neg)$$

Logique intuitionniste vs. classique : validité vs. prouvabilité

Les règles ajoutées pour $\wedge$, $\rightarrow$, $\neg$, $\bot$, et $\vee$ caractérisent la logique (propositionnelle) dite intuitionniste.

Théorème de correction

Les règles de déduction de la logique intuitionniste ne permettent de prouver que des séquents valides : si $\Gamma \vdash C$ alors $\Gamma \models C$.

Logique intuitionniste vs. classique : validité vs. prouvabilité

Théorème

Il existe au moins un séquent valide non-prouvable dans le calcul défini pour la logique intuitionniste.

Preuve : (partielle) considérez la formule :

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

c'est une tautologie mais : $\vdash \neg(p \wedge q) \rightarrow \neg p \vee \neg q$ n'a pas de preuve dans le calcul précédent. □

Règle du "Tertium non datur"

- Pour avoir un calcul complet, il faut étudier la logique classique et ajouter la règle du "Tertium non datur" (*tiers-exclu*).

$$\frac{}{\Gamma \vdash A \vee \neg A}\ (TND) \qquad \text{(tertium non datur)}$$

- Dans la littérature, on trouve aussi deux autres formulations de cette règle qui sont équivalentes mais ce point ne sera pas discuté dans ce cours.

Validité vs. prouvabilité

On obtient alors un calcul correct et complet :

Théorème de correction

Les règles de déduction de la logique classique ne permettent de prouver que des séquents valides de L_{prop} : si $\Gamma \vdash C$ alors $\Gamma \models C$.

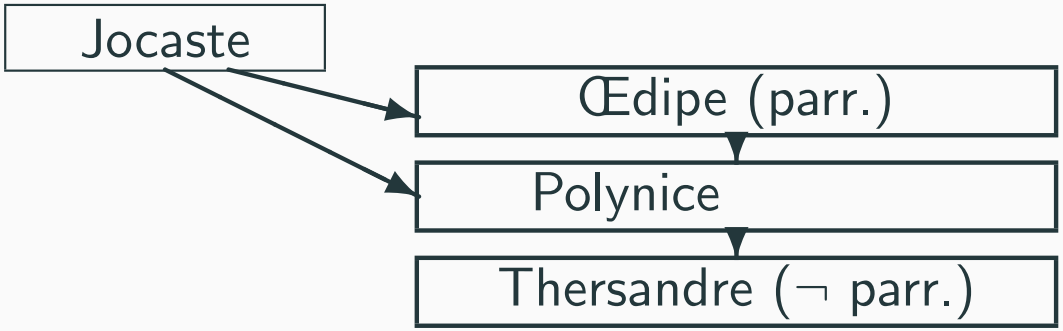
Théorème de complétude

Tout séquent valide est prouvable dans le calcul défini grâce aux règles de la logique classique de L_{prop} : si $\Gamma \models C$ alors $\Gamma \vdash C$.

Exemple du raisonnement classique

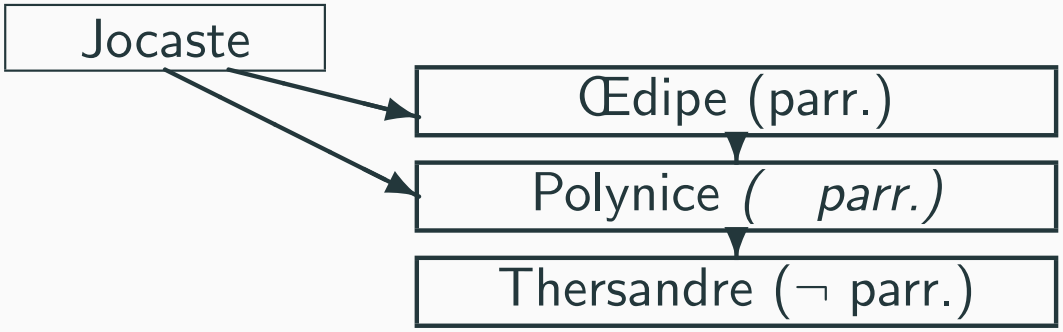
- Jocaste est la mère d'Œdipe.
- Jocaste et Œdipe sont les parents de Polynice.
- Polynice est le père de Thersandre.
- Œdipe est un parricide
- Thersandre n'est pas un parricide.

Exemple du raisonnement classique (2)



Est-ce que Jocaste a un fils qui est un parricide qui lui-même a un fils qui n'est pas un parricide ?

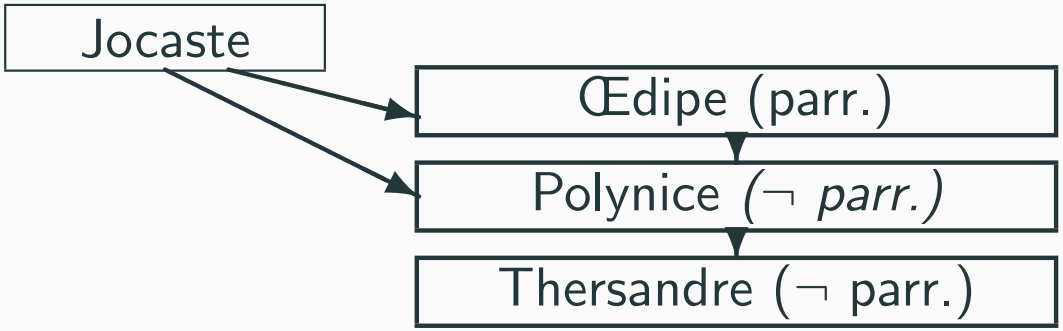
Exemple du raisonnement classique (2)



Est-ce que Jocaste a un fils qui est un parricide qui lui-même a un fils qui n'est pas un parricide ?

Cas 1 : Si Polynice est un parricide, alors Jocaste a un fils (Polynice) qui est un parricide et qui lui-même a un fils (Thersandre) qui n'est pas un parricide.

Exemple du raisonnement classique (2)



Est-ce que Jocaste a un fils qui est un parricide qui lui-même a un fils qui n'est pas un parricide ?

Cas 2 : Si Polynice n' est pas un parricide, alors Jocaste a un fils (Œdipe) qui est un parricide et qui lui-même a un fils (Polynice) qui n'est pas un parricide.

Preuve de $\neg(p \wedge q) \rightarrow \neg p \vee \neg q$

$$\dfrac{\dfrac{}{\Gamma_1 \vdash p \vee \neg p}\ (TND) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_2 \vdash \neg(p \wedge q)}\ (Ax) \quad \dfrac{\dfrac{}{\Gamma_2 \vdash p}\ (Ax) \quad \dfrac{}{\Gamma_2 \vdash q}\ (Ax)}{\Gamma_2 \vdash p \wedge q}\ (I\wedge)}{\Gamma_1, p, q \vdash \bot}\ (E\neg)}{\Gamma_1, p \vdash \neg q}\ (I\neg)}{\Gamma_1, p \vdash \neg p \vee \neg q}\ (I\vee_2) \qquad \dfrac{\dfrac{}{\Gamma_1, \neg p \vdash \neg p}\ (Ax)}{\Gamma_1, \neg p \vdash \neg p \vee \neg q}\ (I\vee_1)}{\Gamma_1 \vdash \neg p \vee \neg q}\ (E\vee)}{\vdash \neg(p \wedge q) \rightarrow \neg p \vee \neg q}\ (I \longrightarrow)$$

avec :

- $\Gamma_1 = \neg(p \wedge q)$
- $\Gamma_2 = \neg(p \wedge q), p, q$

Stratégies de preuve

Remarque

Le calcul finalement obtenu est suffisamment expressif pour modéliser des stratégies de preuves bien connues.

- Une règle correspondant au **raisonnement par l'absurde** :

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

Stratégies de preuve

Remarque

Le calcul finalement obtenu est suffisamment expressif pour modéliser des stratégies de preuves bien connues.

- Une règle correspondant au **raisonnement par l'absurde** :

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

... peut être dérivée des règles présentées précédemment :

$$\dfrac{\dfrac{}{\Gamma \vdash A \vee \neg A}\,(TND) \qquad \dfrac{}{\Gamma, A \vdash A}\,(Ax) \qquad \dfrac{\Gamma, \neg A \vdash \bot}{\Gamma, \neg A \vdash A}\,(E\bot)}{\Gamma \vdash A}\,(E\vee)$$

Stratégies de preuve

- Une règle correspondant au **raisonnement par contraposition** :

$$\frac{\Gamma \vdash \neg Q \longrightarrow \neg P}{\Gamma \vdash P \longrightarrow Q}$$

Stratégies de preuve

- Une règle correspondant au **raisonnement par contraposition** :

$$\frac{\Gamma \vdash \neg Q \longrightarrow \neg P}{\Gamma \vdash P \longrightarrow Q}$$

... peut être dérivée des règles présentées précédemment :

$$\dfrac{\dfrac{}{\Gamma, P \vdash Q \vee \neg Q}\,(TND) \qquad \dfrac{}{\Gamma, P, Q \vdash Q}\,(Ax) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash \neg Q \to \neg P}{\Gamma' \vdash \neg Q \to \neg P}\,(Aff) \qquad \dfrac{}{\Gamma' \vdash \neg Q}\,(Ax)}{\Gamma' \vdash \neg P}\,(E\to) \qquad \dfrac{}{\Gamma' \vdash P}\,(Ax)}{\Gamma' \vdash \bot}\,(E\neg)}{\Gamma, P, \neg Q \vdash Q}\,(E\bot)}{\dfrac{\Gamma, P \vdash Q}{\Gamma \vdash P \to Q}\,(I\to)}\,(E\vee)$$

avec $\Gamma' = \Gamma, P, \neg Q$.

Stratégies de preuve

Exercice

Sur la diapo 14, une stratégie de preuve correspondant à la règle suivante a été utilisée :

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

Montrez que cette règle peut être dérivée à partir du calcul de la déduction naturelle pour la logique des propositions.

Plan

Rappel : Syntaxe de la logique des prédicats - Termes et formules

Dans la logique propositionnelle, les expressions syntaxiques sont des **formules** pouvant être vraies ou fausses

Rappel : Syntaxe de la logique des prédicats - Termes et formules

Dans la logique propositionnelle, les expressions syntaxiques sont des **formules** pouvant être vraies ou fausses

Dans la logique des prédicats, nous avons deux *catégories syntaxiques* : les **termes** et les **formules**. Un terme représente un *individu*.

Rappel : Syntaxe de la logique des prédicats - Termes

Définition (termes)

Soient :

- VAR un ensemble de variables d'individus,
- FON_n un ensemble des fonctions n-aires.

L'ensemble $TERM$ des termes est défini par induction comme le plus petit ensemble qui satisfait :

1. **Variable :** si $x \in VAR$, alors $x \in TERM$
2. **Application de fonction :** si $t_1 \in TERM, \ldots, t_n \in TERM$ et $f \in FON_n$, alors $f(t_1, \ldots, t_n) \in TERM$

On appelle des éléments de FON_0 des *constantes*.
Souvent, on écrit c au lieu de $c()$.

Exemples : Soient $f \in FON_2$, $x, y \in VAR$, $g \in FON_1$, $\pi \in FON_0$

- $f(x, \pi) \in TERM$
- $f(x, g(f(y, \pi))) \in TERM$

Rappel : Syntaxe de la logique des prédicats - Formules

Définition (formules)

Soit $PRED_n$ un ensemble des prédicats n-aires, l'ensemble L_{pred} des formules est défini par induction comme le plus petit ensemble qui satisfait :

1. **Application de prédicat :** si $t_1 \in TERM, \ldots, t_n \in TERM$ et $P \in PRED_n$, alors $P(t_1, \ldots, t_n) \in L_{pred}$
2. **Constante "faux" :** $\bot \in L_{pred}$
3. **Négation :** Si $A \in L_{pred}$, alors $(\neg A) \in L_{pred}$
4. **Connecteurs binaires :** Si $A \in L_{pred}$ et $B \in L_{pred}$, alors $(A \wedge B) \in L_{pred}, (A \vee B) \in L_{pred}, (A \longrightarrow B) \in L_{pred}$
5. **Quantificateur universel** ("pour tout") : Si $A \in L_{pred}$ et $x \in VAR$, alors $(\forall x.A) \in L_{pred}$
6. **Quantificateur existentiel** ("il existe") : Si $A \in L_{pred}$ et $x \in VAR$, alors $(\exists x.A) \in L_{pred}$

Remarque : L_{pred} dépend de $TERM$, mais pas inversement.

Logique sortée

Remarque importante, source de confusion avec l'UE Algo.-Prog

- Dans l'UE Algo.-Prog., vous écrivez des spécifications comme :
  $\forall I \in \mathit{Etu}.\ \mathit{YeuxBleus}(I)$

Logique sortée

Remarque importante, source de confusion avec l'UE Algo.-Prog

- Dans l'UE Algo.-Prog., vous écrivez des spécifications comme :
  $$\forall I \in \mathit{Etu}.\ \mathit{YeuxBleus}(I)$$
- Dans les UEs Logique 1 et 2, on écrit plutôt :
  $$\forall I.\ \mathit{Etudiant}(I) \longrightarrow \mathit{YeuxBleus}(I)$$

Logique sortée

Remarque importante, source de confusion avec l'UE Algo.-Prog

- Dans l'UE Algo.-Prog., vous écrivez des spécifications comme :
  $$\forall I \in \mathit{Etu}.\ \mathit{YeuxBleus}(I)$$
- Dans les UEs Logique 1 et 2, on écrit plutôt :
  $$\forall I.\ \mathit{Etudiant}(I) \longrightarrow \mathit{YeuxBleus}(I)$$
- Ce sont deux approches syntaxiquement différentes :
  - La première considère plusieurs univers séparés (en nombre fini) pour les objets (Etu, Prof, nat, bool, etc) $\rightsquigarrow$ logique **sortée**
  - La seconde considère un univers unique dans lequel on utilise des prédicats "ensembles" pour typer les objets

Rappel : Convention pour écrire les formules de L_{pred}

- La formule $\forall x.A \wedge B$ se lit-elle :
  - $(\forall x.A) \wedge B$, ou
  - $\forall x.(A \wedge B)$?

Rappel : Convention pour écrire les formules de L_{pred}

- La formule $\forall x.A \wedge B$ se lit-elle :
  - $(\forall x.A) \wedge B$, ou
  - $\forall x.(A \wedge B)$?

- Convention : le quantificateur porte sur la plus grande formule possible à partir du point après l'occurrence liante de la variable derrière le quantificateur

Rappel : Convention pour écrire les formules de L_{pred}

- La formule $\forall x.A \wedge B$ se lit-elle :
  - $(\forall x.A) \wedge B$, ou
  - $\forall x.(A \wedge B)$?

- Convention : le quantificateur porte sur la plus grande formule possible à partir du point après l'occurrence liante de la variable derrière le quantificateur

- Dans l'exemple, ça donne $\forall x.(A \wedge B)$

- Exemples supplémentaires :
  - $((\forall x.A) \longrightarrow (\forall y.B)) \rightsquigarrow (\forall x.A) \longrightarrow \forall y.B$
  - $(\forall x.(A \longrightarrow (\forall y.B))) \rightsquigarrow \forall x.A \longrightarrow \forall y.B$

Rappel : Variables libres et variables liées

- Toute occurrence d'une variable dans une formule est liée ou libre ou liante. Considérons la formule suivante :

$$(Q(x) \vee \exists x.\, \forall y.\, P(f(x), z) \wedge Q(y)) \vee \forall x.\, R(x, z, g(x))$$

Rappel : Variables libres et variables liées

- Toute occurrence d'une variable dans une formule est liée ou libre ou liante. Considérons la formule suivante :

$$(Q(x) \vee \exists x.\, \forall y.\, P(f(x), z) \wedge Q(y)) \vee \forall x.\, R(x, z, g(x))$$

Voici ses variables liées,

Rappel : Variables libres et variables liées

- Toute occurrence d'une variable dans une formule est liée ou libre ou liante. Considérons la formule suivante :

$$(Q(x) \vee \exists x.\, \forall y.\, P(f(x), z) \wedge Q(y)) \vee \forall x.\, R(x, z, g(x))$$

Voici ses variables liées, libres,

Rappel : Variables libres et variables liées

- Toute occurrence d'une variable dans une formule est liée ou libre ou liante. Considérons la formule suivante :

$$(Q(x) \vee \exists x.\, \forall y.\, P(f(x), z) \wedge Q(y)) \vee \forall x.\, R(x, z, g(x))$$

  Voici ses variables liées, libres, et liantes.

Rappel : Variables libres et variables liées

- Toute occurrence d'une variable dans une formule est liée ou libre ou liante. Considérons la formule suivante :

$$(Q(x) \vee \exists x.\, \forall y.\, P(f(x), z) \wedge Q(y)) \vee \forall x.\, R(x, z, g(x))$$

  Voici ses variables liées, libres, et liantes.

- Une formule sans occurrences libres de variables est close.

Rappel : Variables libres et variables liées

- Toute occurrence d'une variable dans une formule est liée ou libre ou liante. Considérons la formule suivante :
$$(Q(x) \vee \exists x.\, \forall y.\, P(f(x), z) \wedge Q(y)) \vee \forall x.\, R(x, z, g(x))$$
Voici ses variables liées, libres, et liantes.

- Une formule sans occurrences libres de variables est close.

- On note $fv(F)$ pour les variables libres dans la formule F.

Substitutions (1)

On définit la fonction récursive de substitution d'une variable x par un terme s :

Définition (substitution)

- Substitution dans un terme :
  $t[s/x]$, où x est une variable et t et s sont des termes :
  - **Variable :** $x[s/x] = s$ et $y[s/x] = y$ pour $y \neq x$
  - **Application de fonction :**
    $f(t_1, \ldots, t_n)[s/x] = f(t_1[s/x], \ldots, t_n[s/x])$
- Substitution dans une formule :
  $F[s/x]$, où x est une variable, s est un terme et F une formule :
  - **Application de prédicat :**
    $P(t_1, \ldots, t_n)[s/x] = P(t_1[s/x], \ldots, t_n[s/x])$
  - **Constante :** $\bot[s/x] = \bot$
  - **Négation :** $(\neg A)[s/x] = (\neg A[s/x])$
  - **Conjonction :** $(A \wedge B)[s/x] = (A[s/x] \wedge B[s/x])$
  - **Disjonction :** $(A \vee B)[s/x] = (A[s/x] \vee B[s/x])$
  - **Implication :** $(A \longrightarrow B)[s/x] = (A[s/x] \longrightarrow B[s/x])$
  - *... (à suivre)*

Substitutions (2)

On définit la fonction récursive de substitution d'une variable x par un terme s :

Définition (substitution)

- Substitution dans une formule :
  - ... *(suite)*
  - **Quantificateur universel :**
    1. $(\forall x.A)[s/x] = (\forall x.A)$
    2. $(\forall y.A)[s/x] = (\forall y.(A[s/x]))$ si $y \notin fv(s)$
    3. $(\forall y.A)[s/x] = (\forall y'.(A[y'/y][s/x]))$ si $y \in fv(s)$, où y' est une variable "fraîche" (c.-à-d. $y' \notin fv(s), y' \notin fv(A)$)
  - **Quantificateur existentiel :** en analogie avec $\forall$

Veuillez noter que les parenthèses écrites en rouge ci-dessus ne font pas officiellement partie de la formule $(\forall y.A)[s/x]$ mais ne servent que pour délimiter l'appel récursive à la fonction de substitution.

Substitutions (3)

Le renommage dans le troisième cas pour les quantificateurs assure qu'une substitution soit saine : la variable quantifiée y diffère des variables de s.

Autrement on aurait $(\forall y.R(y,x))[f(y)/x] = (\forall y.R(y,f(y)))$, donc l'occurrence libre dans $f(y)$ serait capturée par accident.

Exercices

Calculer les substitutions suivantes :

- $(\forall x.\exists y.R(x,y) \wedge P(z))[f(v)/z]$
- $(\forall x.\exists y.R(x,y) \wedge P(z))[f(x)/z]$
- $(\forall x.\exists y.R(x,y) \wedge P(z))[f(v)/x]$

Déduction Naturelle pour L_{pred}

C'est une **extension** de la déduction naturelle pour L_{prop} :

- Les règles pour L_{prop} restent en vigueur
- Nouvelles règles pour les quantificateurs : $(I\forall)$, $(E\forall)$, $(I\exists)$, $(E\exists)$
- Construction d'un arbre de dérivation, ... comme pour L_{prop}

Déduction Naturelle pour L_{pred}

C'est une **extension** de la déduction naturelle pour L_{prop} :

- Les règles pour L_{prop} restent en vigueur
- Nouvelles règles pour les quantificateurs : $(I\forall)$, $(E\forall)$, $(I\exists)$, $(E\exists)$
- Construction d'un arbre de dérivation, ... comme pour L_{prop}

Règle d'élimination $(E\forall)$

$$\frac{\Gamma \vdash \forall x.A}{\Gamma \vdash A[s/x]}\ (E\forall)$$

Lecture informelle :

A est vrai pour tout x. Donc, il est vrai en particulier pour un s (que je peux choisir).

Règle d'introduction ($I\forall$) (1)

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x.A}\ (I\forall)$$

Condition :

$x \notin fv(\Gamma)$ afin de ne pas découpler les variables qui se réfèrent au même objet.

Lecture informelle :

Si A est vrai pour n'importe quel x, alors aussi $\forall x.A$ est vrai.

Règle d'introduction ($I\forall$) (2)

Exemple d'une preuve incorrecte car ne respectant pas la condition d'application de ($I\forall$) :

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall x.P(x) \vee Q(x)}\,(Ax)}{\Gamma \vdash P(x) \vee Q(x)}\,(E\forall) \qquad \dfrac{\dfrac{\dfrac{}{\Gamma_P \vdash P(x)}\,(Ax)}{\Gamma_P \vdash \forall x.P(x)}\,(I\forall)}{\Gamma_P \vdash (\forall x.P(x)) \vee \forall x.Q(x)}\,(I\vee_1) \qquad \dfrac{\dots}{\Gamma_Q \vdash \dots}\,(I\vee_2)}{\Gamma \vdash (\forall x.P(x)) \vee \forall x.Q(x)}\,(E\vee)$$

où $\Gamma = \forall x.P(x) \vee Q(x)$ et $\Gamma_P = \forall x.P(x) \vee Q(x), P(x)$ et $\Gamma_Q = \forall x.P(x) \vee Q(x), Q(x)$.

Règle d'introduction ($I\exists$)

$$\frac{\Gamma \vdash A[s/x]}{\Gamma \vdash \exists x.A}\ (I\exists)$$

Lecture informelle :

A est vrai pour un s. Donc, il existe un x pour lequel A est vrai.

Règle d'élimination $(E\exists)$ (1)

$$\frac{\Gamma \vdash \exists x.A \quad \Gamma, A \vdash C}{\Gamma \vdash C}\ (E\exists)$$

Condition :
$x \notin fv(\Gamma) \cup fv(C)$.

Lecture informelle :
Pour montrer C, sachant que $\exists x.A$, il suffit de postuler A **pour un** x **dont on ne connaît pas l'identité exacte** et de montrer C.

Règle d'élimination ($E\exists$) (2)

Exemple :

$$\dfrac{\dfrac{}{\Gamma_1 \vdash \exists x.P(x) \wedge Q(x)}\,(Ax) \quad \dfrac{\dfrac{\dfrac{}{\Gamma_2 \vdash P(x) \wedge Q(x)}\,(Ax)}{\Gamma_2 \vdash P(x)}\,(E\wedge_1)}{\Gamma_1, P(x) \wedge Q(x) \vdash \exists x.P(x)}\,(I\exists)}{\dfrac{\exists x.P(x) \wedge Q(x) \vdash \exists x.P(x)}{\vdash (\exists x.P(x) \wedge Q(x)) \longrightarrow \exists x.P(x)}\,(I \longrightarrow)}\,(E\exists)$$

avec

- $\Gamma_1 = \exists x.P(x) \wedge Q(x)$
- $\Gamma_2 = \Gamma_1, P(x) \wedge Q(x)$

Règle d'élimination $(E\exists)$ (3)

Exemple d'une preuve **incorrecte** car ne respectant pas la condition d'application de $(E\exists)$:

$$\dfrac{\dfrac{\dfrac{\exists x.P(x) \vdash \exists x.P(x) \quad \exists x.P(x), P(x) \vdash P(x)}{\exists x.P(x) \vdash P(x)}\,(E\exists)}{\exists x.P(x) \vdash \forall x.P(x)}\,(I\forall)}{\vdash (\exists x.P(x)) \longrightarrow \forall x.P(x)}\,(I \longrightarrow)$$

Remarques :

- L'application de $(I\forall)$ est correcte : x n'apparaît pas *libre* dans l'hypothèse
- L'application de $(E\exists)$ est incorrecte : x apparaît libre dans la conclusion

Validité vs. prouvabilité

Le calcul ainsi défini pour L_{pred} en logique classique est correct et complet :

Théorème de correction

Les règles de déduction de la logique classique ne permettent de prouver que des séquents valides de L_{pred} : si $\Gamma \vdash C$ alors $\Gamma \models C$.

Théorème de complétude

Tout séquent valide est prouvable dans le calcul défini grâce aux règles de la logique classique de L_{pred} : si $\Gamma \models C$ alors $\Gamma \vdash C$.

Fin de cette partie : qu'est-ce qui est exigible à l'examen ?

Check-list des attendus à l'examen :

- ☐ Savoir utiliser le calcul de la logique minimale pour faire une preuve en déduction naturelle
- ☐ Savoir utiliser le calcul de la logique des propositions pour faire une preuve en déduction naturelle
- ☐ Savoir utiliser le calcul de la logique des prédicats pour faire une preuve en déduction naturelle
- ☐ Connaître la différence entre logique intuitionniste et logique classique
- ☐ Savoir lire et interpréter une règle d'inférence quelconque
- ☐ Savoir construire des règles dérivées
- ☐ Le cours doit être relu et compris

Preuve d'égalité de termes par unification

But de l'unification : prouver l'égalité de termes

Problème :

- Étant donnés deux termes t_1, t_2.
- Un problème d'unification a la forme $t_1 \stackrel{?}{=} t_2$.
- Le but de l'unification est de trouver une *substitution* σ telle que t_1, t_2 deviennent égaux, c.-à-d., $\sigma(t_1) = \sigma(t_2)$.

But de l'unification : prouver l'égalité de termes

Problème :

- Étant donnés deux termes t_1, t_2.
- Un problème d'unification a la forme $t_1 \stackrel{?}{=} t_2$.
- Le but de l'unification est de trouver une *substitution* σ telle que t_1, t_2 deviennent égaux, c.-à-d., $\sigma(t_1) = \sigma(t_2)$.

À préciser :

- notion de *terme*
- notion d'*égalité*
- notion de *substitution*

Motivation : application de l'unification (1)

Preuves : unifier hypothèses et conclusion

- *Question typique :* dans le calcul des séquents, est-ce que la conclusion est une conséquence des hypothèses ?
  Exemple :

$$\frac{P(a),\ P(f(a)) \vdash P(f(\mathbf{?}))}{P(a),\ P(f(a)) \vdash \exists x.P(f(x))}\ (I\exists)$$

Motivation : application de l'unification (1)

Preuves : unifier hypothèses et conclusion

- *Question typique :* dans le calcul des séquents, est-ce que la conclusion est une conséquence des hypothèses ?
  Exemple :

$$\frac{P(a),\ P(f(a)) \vdash P(f(\,\mathbf{?}\,))}{P(a),\ P(f(a)) \vdash \exists x.P(f(x))}\ (I\exists)$$

- ... *se réduit* aux problèmes d'unification suivants :
  - $P(a) \stackrel{?}{=} P(f(x))$ $\rightsquigarrow$ échec
  - $P(f(a)) \stackrel{?}{=} P(f(x))$ $\rightsquigarrow$ unificateur $[x \leftarrow a]$

Motivation : applications de l'unification

Langages de programmation : Inférence de types

- *Question typique :*

  Est-ce que la fonction `List.rev : 'a list -> 'a list`

  est applicable à `[2; 3] : int list` ?

Motivation : applications de l'unification

Langages de programmation : Inférence de types

- *Question typique :*

  Est-ce que la fonction `List.rev : 'a list -> 'a list`

  est applicable à `[2; 3] : int list` ?

- ... *se réduit* au problème d'unification `'a list` $\stackrel{?}{=}$ `int list`

Motivation : applications de l'unification

Langages de programmation : Inférence de types

- *Question typique :*
  Est-ce que la fonction `List.rev : 'a list -> 'a list`
  est applicable à `[2; 3] : int list` ?

- ... *se réduit* au problème d'unification `'a list` $\stackrel{?}{=}$ `int list`

- *Réponse :* l'unificateur suivant existe `['a` $\leftarrow$ `int]`
  ... et donc : `List.rev [2; 3] : int list`

Motivation : applications de l'unification

Langages de programmation : Inférence de types

- *Question typique :*
  Est-ce que la fonction `List.rev : 'a list -> 'a list`
  est applicable à `[2; 3] : int list` ?

- ... *se réduit* au problème d'unification `'a list` $\stackrel{?}{=}$ `int list`

- *Réponse :* l'unificateur suivant existe `['a` $\leftarrow$ `int]`
  ... et donc : `List.rev [2; 3] : int list`

- *Remarque :* ici,
  - Termes = termes de types. Exemple : `(int * bool)`, `int list`, ...
  - Égalité = égalité structurelle

Motivation : applications de l'unification

Langages de programmation : Filtrage

- *Question typique :* étant donné le code :

```
match Node(3, Leaf 1, Leaf 2) with
| Leaf x -> x
| Node (y, nd, Leaf lf) -> lf
```

Quelle valeur est renvoyée ?

Motivation : applications de l'unification

Langages de programmation : Filtrage

- *Question typique :* étant donné le code :

```
match Node(3, Leaf 1, Leaf 2) with
| Leaf x -> x
| Node (y, nd, Leaf lf) -> lf
```

  Quelle valeur est renvoyée ?

- ... *se réduit* aux problèmes d'unification
  - `Leaf x` $\stackrel{?}{=}$ `Node(3, Leaf 1, Leaf 2)`
    $\rightsquigarrow$ échec
  - `Node(y,nd,Leaf lf)` $\stackrel{?}{=}$ `Node(3,Leaf 1,Leaf 2)`
    $\rightsquigarrow$ unificateur `[y` $\leftarrow$ `3, nd` $\leftarrow$ `Leaf 1, lf` $\leftarrow$ `2]`

- *Réponse :* la valeur renvoyée est 2.

Unification syntaxique

- Important ! On s'intéresse à savoir quelle substitution satisfait une équation **syntaxiquement**.

- *Exemples :*
  - $x + 2 \stackrel{?}{=} 2 + x$ peut être unifié syntaxiquement (avec $\sigma = [x \leftarrow 2]$) sans s'intéresser à d'éventuelles significations des symboles des fonctions
  - $3 * x + 2 \stackrel{?}{=} 2 + x$ ne peut pas être unifié syntaxiquement mais il y a une solution sous l'interprétation habituelle de $+$ et $*$ (avec $\sigma = [x \leftarrow 0]$)
    $\rightsquigarrow$ cf. unification *modulo théories* **non traitée dans ce cours**

- *Remarque :* dans la suite, on préférera des symboles neutres : $p(x, 2) \stackrel{?}{=} p(2, x)$ et $p(m(3, x), 2) \stackrel{?}{=} p(2, x)$

Rappel : termes

Définition (terme) :

Soit

- VAR un ensemble de variables
- FON_n un ensemble des fonctions n-aires

L'ensemble $TERM$ des termes est défini par induction comme le plus petit ensemble qui satisfait :

1. **Variable :** si $x \in VAR$, alors $x \in TERM$
2. **Application de fonction :** si $t_1 \in TERM, \ldots, t_n \in TERM$ et $f \in FON_n$, alors $f(t_1, \ldots, t_n) \in TERM$

Les éléments de FON_0 sont des *constantes*.
Exemples : 2, *true*, c, d, ...

Substitution (1)

Définitions (substitution)

- Une substitution σ est une **fonction** qui associe un terme à chaque variable, $\sigma : VAR \rightarrow TERM$.

  Notation : $\sigma = [x_1 \leftarrow t_1;\ \ldots\ ; x_n \leftarrow t_n]$ correspond à la fonction telle que $\sigma(x_i) = t_i$ pour tout entier i avec $1 \leq i \leq n$

- Une substitution σ est étendue à l'application de fonction de manière **récursive** :
  $\sigma(f(t_1, ..., t_n)) = f(\sigma(t_1), ..., \sigma(t_n))$

- La composition de substitutions correspond à la composition usuelle de fonctions : $\sigma' = \sigma_1 \circ \sigma_2$ avec $\sigma'(x) = \sigma_1(\sigma_2(x))$

Substitution (2)

Exemple : $\sigma\big(f(x, g(y))\big)$, où $\sigma = [x \leftarrow h(y); y \leftarrow 42]$

$$\begin{aligned}
&\sigma\big(f(x, g(y))\big) \\
&\quad = f\big(\sigma(x), \sigma(g(y))\big) \\
&\qquad = f\big(h(y), g(\sigma(y))\big) \\
&\qquad\quad = f\big(h(y), g(42)\big)
\end{aligned}$$

À noter ! La substitution est parallèle, pas séquentielle :
$\sigma\big(f(x, g(y))\big) \neq f\big(h(42), g(42)\big)$

Substitution (2)

Exemple : $\sigma\big(f(x, g(y))\big)$, où $\sigma = [x \leftarrow h(y); y \leftarrow 42]$

$$\begin{aligned}
&\sigma\big(f(x, g(y))\big) \\
&\quad = f\big(\sigma(x), \sigma(g(y))\big) \\
&\qquad = f\big(h(y), g(\sigma(y))\big) \\
&\qquad\quad = f\big(h(y), g(42)\big)
\end{aligned}$$

À noter ! La substitution est parallèle, pas séquentielle :
$\sigma\big(f(x, g(y))\big) \neq f\big(h(42), g(42)\big)$

Dans la suite on ne considère que les substitutions idempotentes, pour lesquelles cette distinction est sans objet.

Substitution (3)

Définition (substitution plus générale)

La substitution σ est plus générale que σ' s'il existe une substitution σ_i avec $\sigma' = \sigma_i \circ \sigma$.

Exemple : $\sigma = [x \leftarrow g(y)]$ est plus générale que $\sigma' = [x \leftarrow g(5), z \leftarrow 3, y \leftarrow 5]$. Ici, $\sigma_i = [y \leftarrow 5, z \leftarrow 3]$.

Par exemple : $\sigma'\big(f(x,z)\big) = f(g(5),3)) = \sigma_i\big(f(g(y),z) = \sigma_i(\sigma(f(x,z)))$

Unificateur

Définitions (unificateur)

- Un unificateur des termes t_1, t_2 est une substitution σ telle que $\sigma(t_1) = \sigma(t_2)$.
- On appelle l'unificateur le plus général (*most general unifier*, mgu) de t_1 et t_2 l'unificateur qui est plus général que tout autre unificateur de t_1 et t_2.

Exemple : Pour $g(x, f(y)) \stackrel{?}{=} g(x, f(f(z)))$

- le *mgu* est $[y \leftarrow f(z)]$
- Des unificateurs moins généraux sont :
  - $[y \leftarrow f(4); z \leftarrow 4]$, et
  - $[y \leftarrow f(z), x \leftarrow 7]$

Variables libres d'un terme

Définition (variable libre d'un terme)

L'ensemble des variables libres d'un terme t, noté $fv(t)$, est l'ensemble des variables qui apparaissent dans t (c.-à-d., contrairement au cas des formules, toutes les variables sont libres).

Exemples : soit $VAR = \{x, y, z\}$,

- $fv\big(f(x, g(y))\big) = \{x, y\}$
- $fv\big(f(h(x, x), c)\big) = \{x\}$

Algorithme d'unification (1)

- **L'algorithme** simplifie des paires (E, S), où
  - E est un multi-ensemble d'équations à résoudre
  - S est un ensemble de solutions

Algorithme d'unification (1)

- **L'algorithme** simplifie des paires (E, S), où
  - E est un multi-ensemble d'équations à résoudre
  - S est un ensemble de solutions
- On applique des règles de simplification qui ont la forme $(E, S) \Longrightarrow (E', S')$

Algorithme d'unification (1)

- **L'algorithme** simplifie des paires (E, S), où
  - E est un multi-ensemble d'équations à résoudre
  - S est un ensemble de solutions
- On applique des règles de simplification qui ont la forme $(E, S) \Longrightarrow (E', S')$
- Un ensemble de solutions S a la forme $\{x_1 = s_1, \ldots, x_n = s_n\}$ où
  - tous les x_i sont différents,
  - aucun des x_i n'apparaît dans l'un des s_j.

Algorithme d'unification (1)

- **L'algorithme** simplifie des paires (E, S), où
  - E est un multi-ensemble d'équations à résoudre
  - S est un ensemble de solutions
- On applique des règles de simplification qui ont la forme $(E, S) \Longrightarrow (E', S')$
- Un ensemble de solutions S a la forme $\{x_1 = s_1, \ldots, x_n = s_n\}$ où
  - tous les x_i sont différents,
  - aucun des x_i n'apparaît dans l'un des s_j.
- Idée de l'algorithme : étant donnés t_1, t_2 à unifier :
  - L'algorithme simplifie $(\{t_1 \stackrel{?}{=} t_2\}, \{\})$ à l'aide des règles de simplification jusqu'à arriver à $(\{\}, \{x_1 = s_1, ..., x_n = s_n\})$ **ou alors** on termine avec un échec
  - En cas de non-échec, on va démontrer que le *mgu* est $[x_1 \leftarrow s_1, ..., x_n \leftarrow s_n]$ obtenu directement à partir du dernier S calculé.

Algorithme d'unification (2)

Règles de simplification :

- **Delete :** $(\{t \stackrel{?}{=} t\} \cup E, S) \Longrightarrow (E, S)$

Algorithme d'unification (2)

Règles de simplification :

- **Delete :** $(\{t \stackrel{?}{=} t\} \cup E, S) \Longrightarrow (E, S)$
- **Decompose :** $(\{f(s_1, ..., s_n) \stackrel{?}{=} f(t_1, ..., t_n)\} \cup E, S) \Longrightarrow$
  $(\{s_1 \stackrel{?}{=} t_1, ..., s_n \stackrel{?}{=} t_n\} \cup E, S)$

Algorithme d'unification (2)

Règles de simplification :

- **Delete :** $(\{t \stackrel{?}{=} t\} \cup E, S) \Longrightarrow (E, S)$
- **Decompose :** $(\{f(s_1, ..., s_n) \stackrel{?}{=} f(t_1, ..., t_n)\} \cup E, S) \Longrightarrow$
  $(\{s_1 \stackrel{?}{=} t_1, ..., s_n \stackrel{?}{=} t_n\} \cup E, S)$
- **Clash :** $(\{f(s_1, ..., s_n) \stackrel{?}{=} g(t_1, ..., t_m)\} \cup E, S) \Longrightarrow$ *fail*
  si f et g sont des symboles de fonction différents

Remarque : *fail* = échec

Algorithme d'unification (2)

Règles de simplification :

- **Delete :** $(\{t \stackrel{?}{=} t\} \cup E, S) \Longrightarrow (E, S)$
- **Decompose :** $(\{f(s_1, ..., s_n) \stackrel{?}{=} f(t_1, ..., t_n)\} \cup E, S) \Longrightarrow$
  $(\{s_1 \stackrel{?}{=} t_1, ..., s_n \stackrel{?}{=} t_n\} \cup E, S)$
- **Clash :** $(\{f(s_1, ..., s_n) \stackrel{?}{=} g(t_1, ..., t_m)\} \cup E, S) \Longrightarrow$ *fail*
  si f et g sont des symboles de fonction différents
- **Eliminate :** $(\{x \stackrel{?}{=} t\} \cup E, S) \Longrightarrow (E[x \leftarrow t]; S[x \leftarrow t] \cup \{x = t\})$
  si $t \neq x$ et $x \notin fv(t)$

Remarque : *fail* = échec

Remarque : $E[x \leftarrow t]$ correspond à substituer la variable x par le terme t dans tout terme apparaissant dans les équations de E ($S[x \leftarrow t]$: *idem*).

Algorithme d'unification (2)

Règles de simplification :

- **Delete :** $(\{t \stackrel{?}{=} t\} \cup E, S) \Longrightarrow (E, S)$
- **Decompose :** $(\{f(s_1, ..., s_n) \stackrel{?}{=} f(t_1, ..., t_n)\} \cup E, S) \Longrightarrow$
  $(\{s_1 \stackrel{?}{=} t_1, ..., s_n \stackrel{?}{=} t_n\} \cup E, S)$
- **Clash :** $(\{f(s_1, ..., s_n) \stackrel{?}{=} g(t_1, ..., t_m)\} \cup E, S) \Longrightarrow$ *fail*
  si f et g sont des symboles de fonction différents
- **Eliminate :** $(\{x \stackrel{?}{=} t\} \cup E, S) \Longrightarrow (E[x \leftarrow t]; S[x \leftarrow t] \cup \{x = t\})$
  si $t \neq x$ et $x \notin fv(t)$
- **Check :** $(\{x \stackrel{?}{=} t\} \cup E, S) \Longrightarrow$ *fail*
  si $t \neq x$ et $x \in fv(t)$

Remarque : *fail* = échec ; "check" vient de vérifier si $x \in fv(t)$

Remarque : $E[x \leftarrow t]$ correspond à substituer la variable x par le terme t dans tout terme apparaissant dans les équations de E ($S[x \leftarrow t]$: *idem*).

Algorithme d'unification (2)

Règles de simplification :

- **Delete :** $(\{t \stackrel{?}{=} t\} \cup E, S) \Longrightarrow (E, S)$
- **Decompose :** $(\{f(s_1, ..., s_n) \stackrel{?}{=} f(t_1, ..., t_n)\} \cup E, S) \Longrightarrow$
  $(\{s_1 \stackrel{?}{=} t_1, ..., s_n \stackrel{?}{=} t_n\} \cup E, S)$
- **Clash :** $(\{f(s_1, ..., s_n) \stackrel{?}{=} g(t_1, ..., t_m)\} \cup E, S) \Longrightarrow$ *fail*
  si f et g sont des symboles de fonction différents
- **Eliminate :** $(\{x \stackrel{?}{=} t\} \cup E, S) \Longrightarrow (E[x \leftarrow t]; S[x \leftarrow t] \cup \{x = t\})$
  si $t \neq x$ et $x \notin fv(t)$
- **Check :** $(\{x \stackrel{?}{=} t\} \cup E, S) \Longrightarrow$ *fail*
  si $t \neq x$ et $x \in fv(t)$
- **Orient** $(\{t \stackrel{?}{=} x\} \cup E, S) \Longrightarrow (\{x \stackrel{?}{=} t\} \cup E, S)$
  si t n'est pas une variable

Remarque : *fail* = échec ; "check" vient de vérifier si $x \in fv(t)$

Remarque : $E[x \leftarrow t]$ correspond à substituer la variable x par le terme t dans tout terme apparaissant dans les équations de E ($S[x \leftarrow t]$: *idem*).

Algorithme d'unification (3)

Observations :

- la règle **Orient** permet de dériver des variantes symétriques aux règles **Eliminate** et **Check**.
- Les règles sont (presque) mutuellement exclusives.
- **Question :** quelle règle faut-il modifier pour avoir une simplification déterministe ? Comment ?
- **Question :** pourquoi est-ce que ceci ne modifie pas le résultat de l'algorithme ?
- **Conclusion :** on peut appliquer les règles dans n'importe quel ordre.

Algorithme d'unification (4)

Exemple 1 :

$$
\begin{array}{ll}
 & (\{p(x,2) \stackrel{?}{=} p(2,x)\};\{\}) \\
\stackrel{\text{Decompose}}{\Longrightarrow} & (\{x \stackrel{?}{=} 2,\ \ 2 \stackrel{?}{=} x\};\{\}) \\
\stackrel{\text{Eliminate}}{\Longrightarrow} & (\{2 \stackrel{?}{=} 2\};\{x = 2\}) \\
\stackrel{\text{Delete}}{\Longrightarrow} & (\{\};\{x = 2\})
\end{array}
$$

Donc : $\sigma = [x \leftarrow 2]$

Algorithme d'unification (5)

Exemple 2 :

$$
\begin{array}{ll}
 & (\{f(x,y) \stackrel{?}{=} f(y,g(x))\};\{\}) \\
\stackrel{\text{Decompose}}{\Longrightarrow} & (\{x \stackrel{?}{=} y,\ \ y \stackrel{?}{=} g(x)\};\{\}) \\
\stackrel{\text{Eliminate}}{\Longrightarrow} & (\{y \stackrel{?}{=} g(y)\};\{x = y\}) \\
\stackrel{\text{Check}}{\Longrightarrow} & \mathit{fail}
\end{array}
$$

Exercice

Quel est le *mgu* pour l'équation suivante ? $p(m(3,x),2) \stackrel{?}{=} p(2,x)$

Propriétés de l'algorithme d'unification

Questions à se poser : étant donnés deux termes t_1, t_2 :

- Correction : est-ce que l'algorithme est correct, c.-à-d. est-ce que toute σ qu'il fournit satisfait $\sigma(t_1) = \sigma(t_2)$?
- Complétude : est-ce que l'algorithme est complet, c.-à-d. est-ce qu'il fournit un unificateur si t_1, t_2 sont unifiables ?
- Terminaison : est-ce que l'algorithme s'arrête quelle que soit l'équation à résoudre ?
- Non-blocage : est-ce que l'algorithme termine ou bien avec *fail*, ou bien avec un résultat de la forme $(\{\}, S)$?

Terminaison de l'algorithme

Théorème de terminaison

Pour toute équation à résoudre, l'algorithme d'unification s'arrête.

Preuve :

Argument semi-formel : Après chaque application de règle $(E, S) \Longrightarrow (E', S')$

- le nombre de variables dans E' est inférieur au nombre de variables dans E, *ou bien*
- le nombre des variables dans E et E' est égal, mais la taille des termes décroît, *ou bien*
- le nombre des variables et la taille des termes restent égaux, mais le nombre d'équations $t \stackrel{?}{=} x$ avec $t \notin VAR$ décroît.

Argument formel : Combinaison d'un ordre lexicographique et multi-ensemble $\rightsquigarrow$ non traitée ici. □

Non-blocage

Théorème de non-blocage

Pour toute équation à résoudre, l'algorithme d'unification termine ou bien avec *fail*, ou bien avec un résultat de la forme $(\{\}, S)$

Preuve : (partielle) pour prouver la propriété de non-blocage, il faut enlever l'une des règles de simplification et montrer alors qu'on peut obtenir une situation de blocage.

Par exemple, sans la règle **Delete**, il est impossible de simplifier $(\{x \stackrel{?}{=} x\}, S)$ □

Correction et Complétude (1)

Théorème de correction et complétude

Pour deux termes t_1 et t_2, l'algorithme d'unification est

- **correct**
- **complet** : il calcule le *mgu* de t_1, t_2

Preuve : par induction sur la longueur de la dérivation

$$(E_0, S_0) \Longrightarrow (E_1, S_1) \Longrightarrow \ldots \Longrightarrow (E_n, S_n)$$

ou

$$(E_0, S_0) \Longrightarrow (E_1, S_1) \Longrightarrow \ldots \Longrightarrow \mathit{fail}$$

Correction et Complétude (2)

Preuve (suite) :

... en utilisant le Lemme :

- Si $(E, S) \Longrightarrow (E', S')$, alors l'ensemble des unificateurs de (E, S) est égal à l'ensemble des unificateurs de (E', S')
- Si $(E, S) \Longrightarrow \mathit{fail}$, alors (E, S) n'a pas d'unificateur

... et en complétant la preuve en :

- précisant la notion d'"ensemble des unificateurs de (E, S)"
- démontrant la propriété pour chaque règle $\square$

Fin de cette partie : qu'est-ce qui est exigible à l'examen ?

Check-list des attendus à l'examen :

- ☐ Savoir utiliser l'algorithme d'unification pour décider de l'égalité de termes donnés
- ☐ Le cours doit être relu et compris

Preuve d'insatisfiabilité par résolution

Rappels : principe de résolution pour L_{prop}

- En "Logique 1", le principe de résolution a été étudié pour répondre à la question suivante :

$$\textit{La formule } F \in L_{prop} \textit{ est-elle insatisfiable ?}$$

- Le principe de résolution est une technique **efficace** pour y répondre, contrairement à la construction de table de vérité

- Il se décompose en **plusieurs** étapes :
  1. mise en forme normale conjonctive de F ;
  2. passage à la forme clausale ;
  3. calculs de résolvantes de façon itérative.

- Pour rappel :
  - F valide $\Leftrightarrow$ $\neg F$ insatisfiable
  - $\{H_1, \cdots, H_n\} \vDash C \Leftrightarrow (H_1 \wedge \cdots H_n \wedge \neg C)$ insatisfiable

Principe de résolution pour L_{pred}

- En "Logique 2", on va étudier le principe de résolution afin de répondre à la question suivante :

$$\textit{La formule } F \in L_{pred} \textit{ est-elle insatisfiable ?}$$

Intuition sur la procédure : exemple 1

- **Problème** : on veut montrer
  $\forall x.P(x) \longrightarrow Q(x) \vDash (\forall x.P(x)) \longrightarrow \forall x.Q(x)$ c'est-à-dire, montrer :
  $(\forall x.P(x) \longrightarrow Q(x)) \wedge (\forall x.P(x)) \wedge \exists x.\neg Q(x)$ insatisfiable

- **Solution** :
  - on introduit une constante fraîche a ;
  - *clairement*, $(\forall x.P(x) \longrightarrow Q(x)) \wedge (\forall x.P(x)) \wedge \exists x.\neg Q(x)$ est insatisfiable si et seulement si la formule $(\forall x.P(x) \longrightarrow Q(x)) \wedge (\forall x.P(x)) \wedge \neg Q(a)$ l'est aussi ;
  - essayons de dériver une contradiction à partir de la formule précédente :

    1. $\forall x.P(x) \longrightarrow Q(x)$ ..... axiome
    2. $\forall x.P(x)$ ..... axiome
    3. $\neg Q(a)$ ..... axiome
    4. $P(a)$ ..... comme conséquence de 2
    5. $Q(a)$ ..... comme conséquence de 1 et 4
    6. $\bot$ ..... contradiction entre 3 et 5

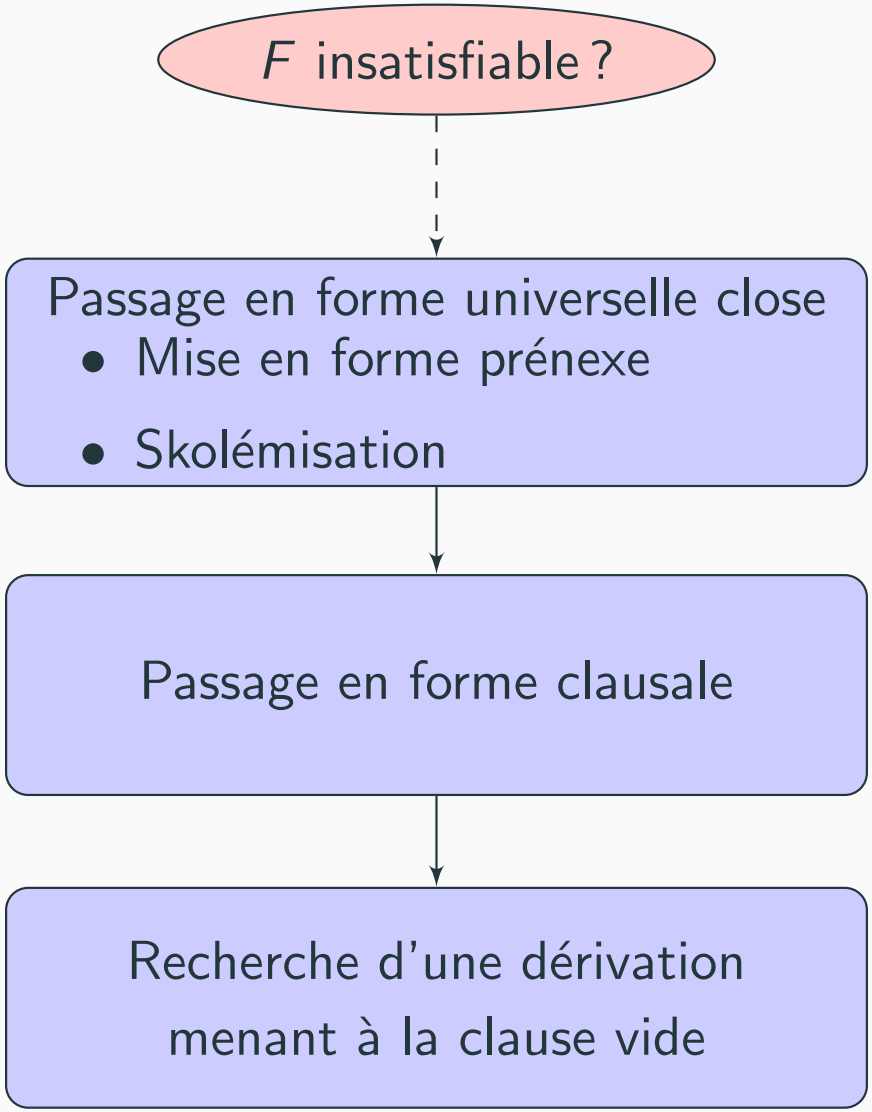
Intuition sur la procédure : exemple 2

- **Problème** : montrer
  $(\forall x.P(x, f(x)) \longrightarrow Q(x)) \wedge (\forall x.\forall y.P(f(x), f(y))) \wedge \exists x.\neg Q(f(x))$
  insatisfiable

- **Solution** :
  - on introduit une constante fraîche a ;
  - essayons de dériver une contradiction à partir de la formule précédente :

    1. $\forall x.P(x, f(x)) \longrightarrow Q(x)$ ..... axiome
    2. $\forall x.\forall y.P(f(x), f(y))$ ..... axiome
    3. $\neg Q(f(a))$ ..... axiome
    4. $P(f(a), f(f(a)))$ ..... comme conséquence de 2
    5. $Q(f(a))$ ..... comme conséquence de 1 et 4
    6. $\bot$ ..... contradiction entre 3 et 5

Intuition sur la procédure : les grandes étapes



Intuition sur la procédure : les grandes étapes

Problème

La formule F avec $F \in L_{pred}$ est-elle insatisfiable ?

1ère étape : on met F sous forme universelle close, c.-à-d. sous la forme :

$$\forall x_1. \cdots \forall x_n. \phi$$

où ϕ est une formule sans quantificateur.

Cette transformation s'effectue en 2 temps :

1. Mise sous forme **prénexe** :

   $$\mathcal{Q}_1 x_1. \mathcal{Q}_2 x_2. \cdots \mathcal{Q}_n x_n. \phi$$

   où $\mathcal{Q}_1, \cdots, \mathcal{Q}_n \in \{\exists, \forall\}$ et ϕ ne contient pas de quantificateur.
2. **Skolémisation** : élimination des quantificateurs existentiels à l'aide de nouveaux symboles de fonctions et de constantes (tous différents).
   Exemples : $\exists x. \neg Q(x) \rightsquigarrow \neg Q(a)$
   $\forall x. \exists y. P(x, y) \rightsquigarrow \forall x. P(x, f(x))$

Intuition sur la procédure : les grandes étapes

2ème étape : on passe la formule obtenue à l'étape précédente sous forme clausale.

La mise en forme clausale s'effectue en plusieurs temps :

1. Étant donnée une formule sous forme universelle close :

$$\forall x_1. \cdots \forall x_n. \phi$$

   ϕ est mise sous forme **normale conjonctive** ;

2. On utilise l'équivalence $\forall x. \psi \wedge \psi' \equiv (\forall x. \psi) \wedge \forall x. \psi'$ pour obtenir une **conjonction de disjonctions universellement quantifiées** ;

3. chaque disjonction obtenue représente une clause dont on peut **renommer** les variables (puisqu'elles sont universellement quantifiées) de telle sorte que 2 clauses ne partagent aucune variable

Intuition sur la procédure : les grandes étapes

Exemple :

Étant donnée la formule :
$\forall x.\forall y.\forall z.(\neg P(x, f(x)) \vee Q(x)) \wedge P(f(y), f(z)) \wedge \neg Q(f(a))$

1. Elle est en forme universelle close
2. Avec l'équivalence $\forall x.\psi \wedge \psi' \equiv (\forall x.\psi) \wedge \forall x.\psi'$ on obtient $(\forall x.\forall y.\forall z.\neg P(x, f(x)) \vee Q(x)) \wedge (\forall x.\forall y.\forall z.P(f(y), f(z))) \wedge (\forall x.\forall y.\forall z.\neg Q(f(a)))$
3. Aucun renommage n'est nécessaire et on obtient l'ensemble de clauses :
   $\{\{\neg P(x, f(x)), Q(x)\}, \{P(f(y), f(z))\}, \{\neg Q(f(a))\}\}$

Intuition sur la procédure : les grandes étapes

3ème étape : calcul de résolvantes.

Contrairement au cas de L_{prop}, il est nécessaire d'**unifier** avant de calculer les résolvantes.

Exemple : à partir des clauses précédentes :

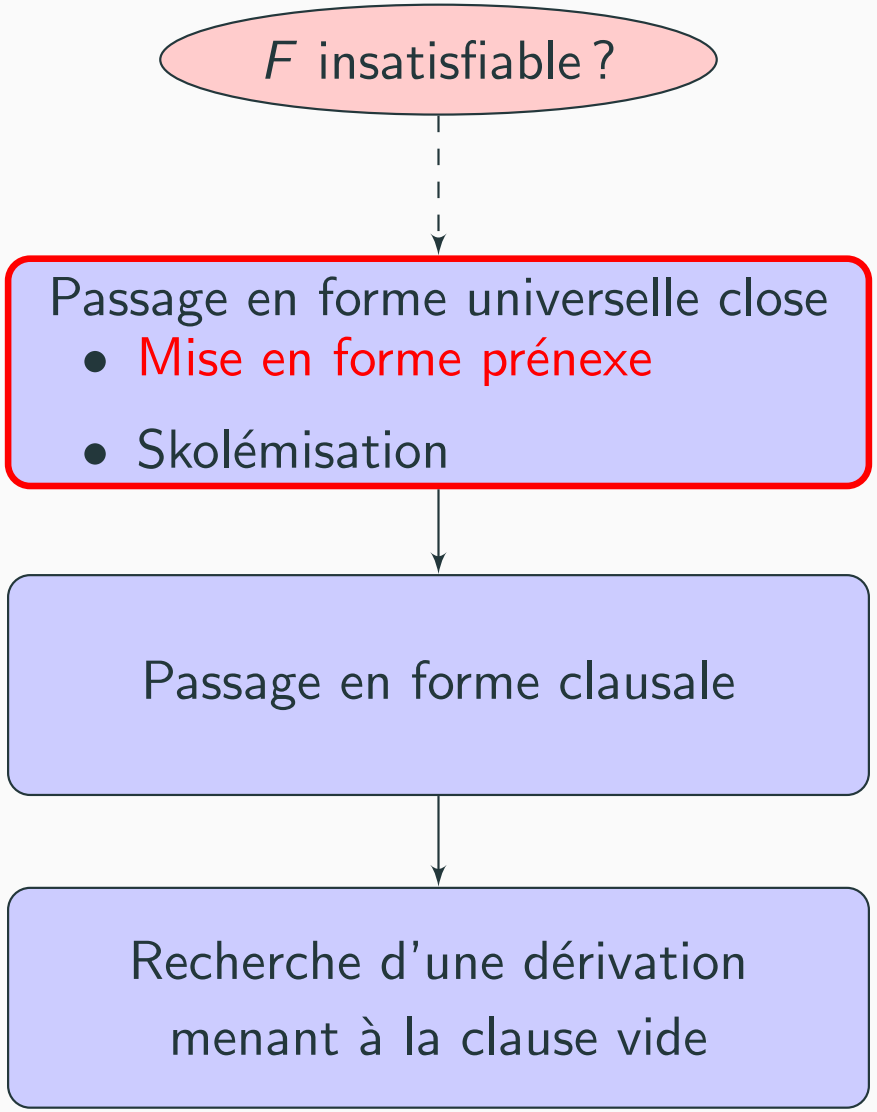
1. $\{\neg P(x, f(x)), Q(x)\}$
2. $\{P(f(y), f(z))\}$
3. $\{\neg Q(f(a))\}$

Il faut d'abord unifier 1. et 3. grâce à $[x \leftarrow f(a)]$ pour obtenir :

4. $\{\neg P(f(a), f(f(a)))\}$

Finalement, on obtient la clause vide avec 2. et 4. en unifiant $P(f(y), f(z))$ avec $\neg P(f(a), f(f(a)))$ en prenant comme unificateur $[y \leftarrow a; z \leftarrow f(a)]$.

Les grandes étapes



Mise en forme prénexe : définition

Définition (forme prénexe)

Une formule est sous forme prénexe si et seulement elle est de la forme

$$\mathcal{O}_1 x_1 . \mathcal{O}_2 x_2 . \cdots \mathcal{O}_n x_n . \phi$$

où $\mathcal{O}_1, \cdots, \mathcal{O}_n \in \{\exists, \forall\}$ et ϕ ne contient pas de quantificateur.

Exemples

Les formules en vert ci-dessous sont en forme prénexe, contrairement à celle en rouge :

- $\exists x . \forall y . P(x) \longrightarrow Q(f(x), y)$
- $\forall x . \forall y . \neg P(x)$
- $P(x) \vee Q(x)$
- $\forall x . P(x) \vee \exists y . Q(y)$

Mise en forme prénexe : théorème

Théorème

Pour toute formule de la logique des prédicats, il existe une formule **équivalente** qui soit en forme prénexe.

Mise en forme prénexe : algorithme de conversion

1. **Éliminer les connecteurs** $\longrightarrow$:
   - Réécrire $A \longrightarrow B$ en $\neg A \vee B$
2. **Tirer les négations à l'intérieur**, éliminer les doubles négations :
   - $\neg(A \vee B)$ devient $\neg A \wedge \neg B$ et $\neg(A \wedge B)$ devient $\neg A \vee \neg B$
   - $\neg(\exists x.P(x))$ devient $\forall x.\neg P(x)$ et $\neg(\forall x.P(x))$ devient $\exists x.\neg P(x)$
   - $\neg\neg A$ devient A
3. **Distinguer les variables** : tant qu'il existe une sous-formule de la forme $\mathcal{Q}x.\phi$ telle que x apparaît en dehors de ϕ, on remplace $\mathcal{Q}x.\phi$ par $\mathcal{Q}y.\phi[x \leftarrow y]$ où y est une variable fraîche
4. **Faire passer les quantificateurs à gauche** en appliquant autant que possible les règles suivantes :
   4.1 $(\mathcal{Q}x.A) \wedge B$ devient $\mathcal{Q}x.(A \wedge B)$
   4.2 $(\mathcal{Q}x.A) \vee B$ devient $\mathcal{Q}x.(A \vee B)$
   4.3 $A \wedge (\mathcal{Q}x.B)$ devient $\mathcal{Q}x.(A \wedge B)$
   4.4 $A \vee (\mathcal{Q}x.B)$ devient $\mathcal{Q}x.(A \vee B)$

   ... où $\mathcal{Q} \in \{\exists, \forall\}$

Mise en forme prénexe : exemple

Soit la formule :

$$(\forall x.P(x)) \longrightarrow \exists x.R(x)$$

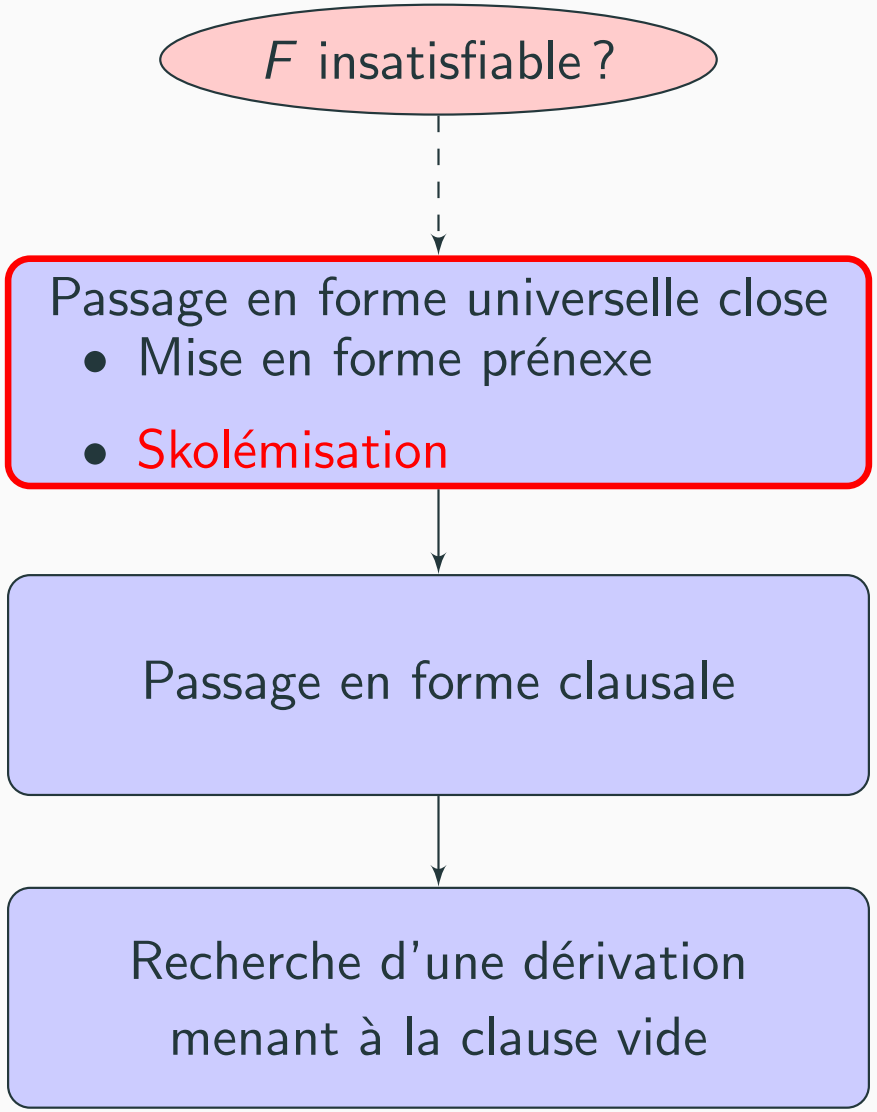
Par application des étapes précédentes, sa forme prénexe est :

1. $\neg(\forall x.P(x)) \vee (\exists x.R(x))$ (élimination de $\longrightarrow$)
2. $(\exists x.\neg P(x)) \vee (\exists x.R(x))$ (négation propagée vers l'intérieur)
3. $(\exists x.\neg P(x)) \vee (\exists y.R(y))$ (distinction des variables)
4. $\exists x.(\neg P(x) \vee (\exists y.R(y)))$ (application de la règle 4.2)

   $\exists x.\exists y.(\neg P(x) \vee R(y))$ (application de la règle 4.4)

On peut supprimer les parenthèses autour $\neg P(x) \vee R(y)$ dans la formule résultante.

Les grandes étapes



Skolémisation : objectif et principe

- **But :** supprimer les quantificateurs $\exists$
- **Observation justifiant l'approche :** considérons la formule

$$\forall x.\exists y.P(x,y)$$

  - elle dit que pour tout x, on peut trouver un y tel que $P(x,y)$
  - soit f la fonction qui pour chaque x donne ce y, c.-à-d. : $y = f(x)$
  - la formule devient : $\forall x.P(x,f(x))$

- La skolémisation généralise cette observation : s'il y a plusieurs quantificateurs universels, la fonction fraîche introduite dépend de toutes les variables venant avant le $\exists y$

Skolémisation partielle : définition

Définition (skolémisation partielle)

Soit F une formule en forme prénexe de la forme :

$$\forall x_1 . \cdots \forall x_n . \exists x_{n+1} . \mathcal{Q}_{n+2} x_{n+2} . \cdots \mathcal{Q}_{n+i} x_{n+i} . \phi$$

Soit f un nouveau symbole fonctionnel d'arité n. La formule :

$$\forall x_1 . \cdots \forall x_n . \mathcal{Q}_{n+2} x_{n+2} . \cdots \mathcal{Q}_{n+i} x_{n+i} . \phi[x_{n+1} \leftarrow f(x_1, \cdots, x_n)]$$

est la skolémisation partielle de F.

Skolémisation partielle : exemples

Exemples

- $\forall x.\forall y.\exists z.R(x,z) \quad \rightsquigarrow \quad \forall x.\forall y.R(x,f(x,y))$
- $\forall x.\exists z.\forall y.\exists w.\exists w'.S(w',x,f(z)) \quad \rightsquigarrow$
  $\forall x.\forall y.\exists w.\exists w'.S(w',x,f(g(x)))$
- $\exists x.\exists z.\forall y.S(x,x,z) \quad \rightsquigarrow \quad \exists z.\forall y.S(a,a,z)$

Skolémisation : définition et théorème

Définition (skolémisation)

Soit F une formule en forme prénexe ayant n quantificateurs existentiels. La skolémisation de F est obtenue par n applications successives de la skolémisation partielle.

Théorème

La formule F est satisfiable si et seulement si sa skolémisation est satisfiable.

*R*emarque : la skolémisation préserve la satisfiabilité mais ne produit pas une formule équivalente à la formule initiale !

Skolémisation : exemples

Exemples

- $\forall x.\forall y.\exists z.R(x,z) \quad \rightsquigarrow \quad \forall x.\forall y.R(x,f(x,y))$
- $\forall x.\exists z.\forall y.\exists w.\exists w'.S(w',x,f(z)) \quad \rightsquigarrow$
  $\forall x.\forall y.S(h(x,y),x,f(g(x)))$
- $\exists x.\exists z.\forall y.S(x,x,z) \quad \rightsquigarrow \quad \forall y.S(a,a,b)$

Forme universelle close

Définition (forme universelle close)

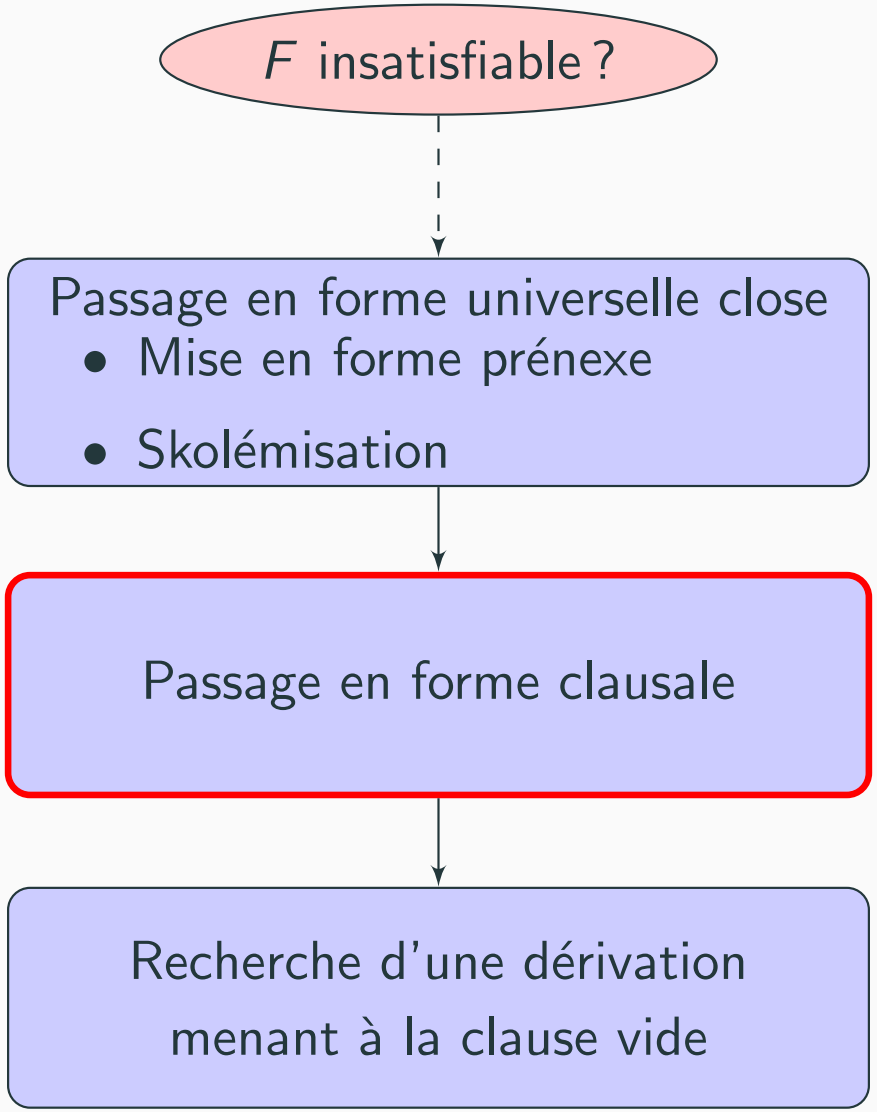
La forme universelle close d'une formule de la logique des prédicats est obtenue en lui appliquant successivement une mise en forme **prénexe** puis une **skolémisation** et est ainsi de la forme :

$$\forall x_1. \cdots \forall x_n. \phi$$

où ϕ est une formule sans quantificateur.

Remarque : le terme *clos* indique uniquement que la seule façon de quantifier est universelle, à l'extérieur ; il ne fait pas référence à l'absence de variable libre dans la formule obtenue.

Les grandes étapes



Forme normale conjonctive : définitions

La notion de littéral change :

Définition (littéral)

Un **littéral** est une formule de la forme $R(t_1, \cdots, t_n)$ ou $\neg R(t_1, \cdots, t_n)$ où $R \in PRED_n$ et $t_1, \cdots, t_n \in TERM$.

Le reste est similaire au cas de L_{prop} (**cf. cours de "Logique 1"**) :

Définitions

- Une clause est une disjonction de littéraux : $L_1 \vee \cdots \vee L_q$ où chaque L_i est un littéral. Elle peut être représentée par l'ensemble de ses littéraux : $\{L_1, \cdots, L_q\}$.
- La clause vide $\{\ \}$ représente $\bot$.
- Une formule est en forme normale conjonctive (FNC) ssi elle est une conjonction de clauses : $C_1 \wedge \cdots \wedge C_n$ où chaque C_i est une clause.

Forme normale conjonctive : conversion

1. Éliminer les connecteurs autres que $\neg, \wedge, \vee$
   1.1 Réécrire $A \leftrightarrow B$ en $(A \longrightarrow B) \wedge (B \longrightarrow A)$
   1.2 Réécrire $A \longrightarrow B$ en $\neg A \vee B$
2. Tirer les négations à l'intérieur, éliminer les doubles négations :
   - $\neg(A \wedge B)$ devient $\neg A \vee \neg B$
   - $\neg(A \vee B)$ devient $\neg A \wedge \neg B$
   - $\neg\neg A$ devient A
3. Distribuer $\vee$ sur $\wedge$:
   - $A \vee (B \wedge C)$ devient $(A \vee B) \wedge (A \vee C)$
   - et pareil pour $(B \wedge C) \vee A$
4. Éliminer $\bot$ et $\top$: $B \wedge (A \vee \bot)$, $A \vee \neg\bot$ ...

Rappel : la FNC d'une formule n'est pas unique.

Remarque

Le passage en forme prénexe comprend déjà les étapes 1 et 2 de la conversion ci-dessus.

Forme normale conjonctive : théorème

Théorème

Pour toute formule sans quantificateur F, il existe une formule F' sans quantificateur et en FNC telle que $F \equiv F'$.

Passage en forme clausale

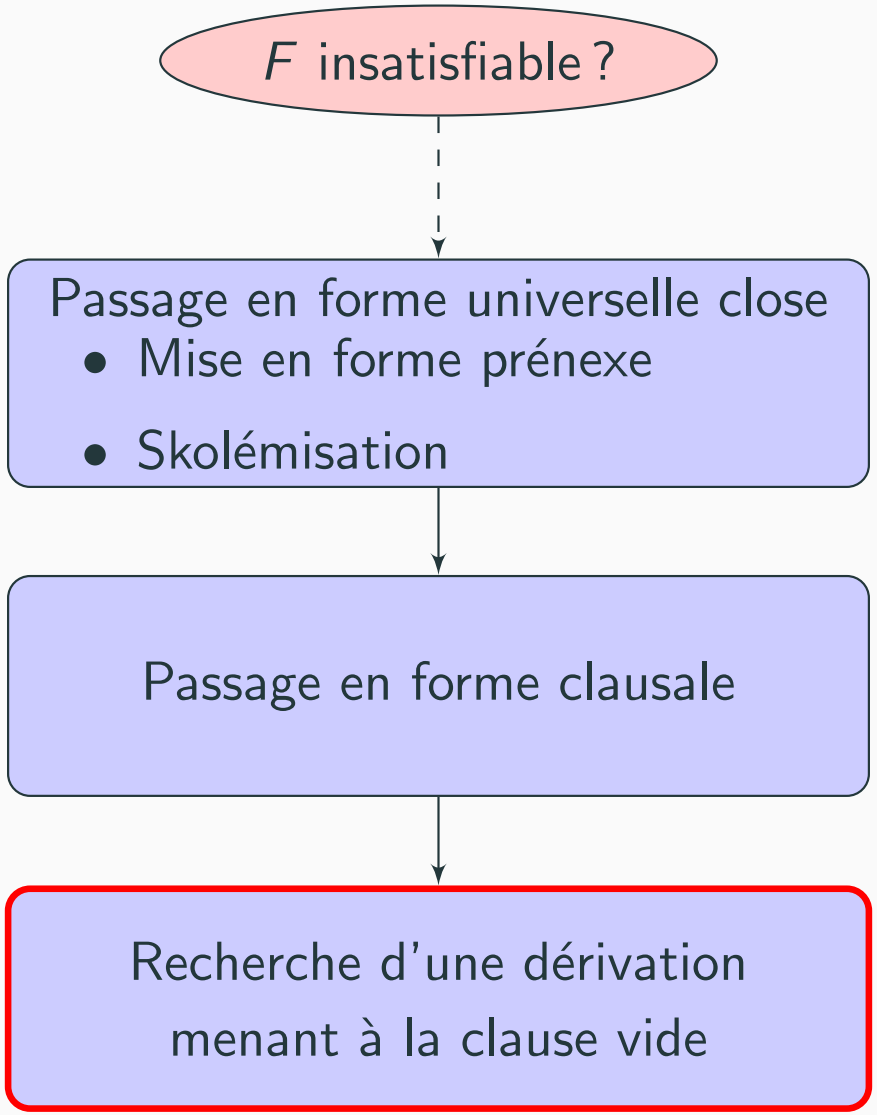
- Soit la formule suivante en forme universelle close :

$$\forall x_1. \cdots \forall x_n. \phi$$

où ϕ est en forme normale conjonctive

- On peut utiliser l'équivalence $\forall x.(\psi \wedge \psi') \equiv (\forall x.\psi) \wedge (\forall x.\psi')$ pour transformer la formule précédente en une **conjonction de disjonctions universellement quantifiées** ;
- Chaque disjonction obtenue représente une clause dont on peut **renommer** les variables (puisqu'elles sont universellement quantifiées) de telle sorte que 2 clauses ne partagent aucune variable $\rightsquigarrow$ c'est la forme clausale
- La forme clausale d'une formule est notée sous la forme de l'ensemble des clauses, elles-mêmes représentées comme un ensemble de littéraux
- Dans la suite, à chaque fois qu'on "utilise" une clause, ce sera une copie "fraîche" qu'on utilise.

Les grandes étapes



Résolution pour L_{pred}

- Cas L_{prop} (**rappel**) :

$$\text{Insatisfiabilité de } F \in L_{prop}$$
$$\Leftrightarrow$$
$$\{\ \} \text{ obtenu par calcul itératif de résolvantes à partir des clauses de } F$$

- Cas L_{pred} (*à venir*) :

$$\text{Insatisfiabilité de } F \in L_{pred}$$
$$\Leftrightarrow$$
$$\text{Dérivation de } \{\ \} \text{ à partir des clauses de } F\text{, dans un calcul de résolvantes modulo unification}$$

Résolution pour L_{pred} : règles

Axiome :

$$\frac{}{C} \quad (C \in \Delta \text{ où } \Delta \text{ est un ensemble de clauses})$$

Règles d'inférence :

$$\frac{D \cup \{R(s_1, \cdots, s_n)\} \qquad C \cup \{\neg R(t_1, \cdots, t_n)\}}{\sigma(D \cup C)} \ (R - \text{résolvante})$$

$$\frac{D \cup \{R(s_1, \cdots, s_n)\} \cup \{R(t_1, \cdots, t_n)\}}{\sigma(D \cup \{R(s_1, \cdots, s_n\})} \ (F^+ - \text{factorisation})$$

$$\frac{D \cup \{\neg R(s_1, \cdots, s_n)\} \cup \{\neg R(t_1, \cdots, t_n)\}}{\sigma(D \cup \{\neg R(s_1, \cdots, s_n)\})} \ (F^- - \text{factorisation})$$

où :

- σ est l'unificateur le plus général du problème $\{s_1 \stackrel{?}{=} t_1, \cdots, s_n \stackrel{?}{=} t_n\}$ et les clauses C et D ne partagent pas de variables (grâce au rénommage !)

Résolution pour L_{pred} : dérivation

Définition (dérivation)

Soit Δ un ensemble de clauses et A une clause, on écrit $\Delta \rhd A$ lorsqu'il est possible de **dériver** A à partir de l'ensemble Δ par résolution, c'est-à-dire en appliquant les règles de résolvante et factorisation précédentes.

En particulier, on cherchera à voir si : $\Delta \rhd \{\ \}$ afin de décider si la formule dont la forme clausale correspond à Δ est **insatisfiable**.

Résolution pour L_{pred} : exemple

On souhaite montrer $\{H_1, H_2, H_3\} \models C$ où :

- $H_1 = \exists x_0.T(x_0)$
- $H_2 = \forall x_2.(D(x_2) \longrightarrow \forall x_1.R(x_1, x_2))$
- $H_3 = \forall x_3 \forall x_4.\neg(T(x_3) \longrightarrow \neg Q(x_4)) \longrightarrow \neg R(x_3, x_4)$
- $C = \forall x_5.(\neg D(x_5) \vee \neg Q(x_5))$

Ceci revient (**exercice** !) à considérer l'ensemble Δ de clauses suivant :

$$\Delta = \{ \quad \{T(a)\}, \{\neg D(x_2), R(x_1, x_2)\}, \{\neg T(x_3), \neg Q(x_4), \neg R(x_3, x_4)\}, \{D(b)\}, \{Q(b)\}\}$$

La dérivation suivante montre $\Delta \rhd \{\ \}$:

$$\dfrac{\dfrac{\dfrac{}{\{D(b)\}}\ (Ax) \qquad \dfrac{\dfrac{}{\{\neg D(x_2), R(x_1, x_2)\}}\ (Ax) \qquad \dfrac{\dfrac{\dfrac{}{\{\neg T(x_3), \neg Q(x_4), \neg R(x_3, x_4)\}}\ (Ax) \qquad \dfrac{}{\{T(a)\}}\ (Ax)}{\{\neg Q(x_4), \neg R(a, x_4)\}}\ (R) \qquad \dfrac{}{\{Q(b)\}}\ (Ax)}{\{\neg R(a, b)\}}\ (R)}{\{\neg D(b)\}}\ (R)}{\{\}}$$

Résolution pour L_{pred} : théorèmes

Théorème de correction

La résolution est correcte : si $\Delta \rhd \{\ \}$ alors la formule dont la forme clausale est Δ est insatisfiable.

Théorème de complétude

La résolution est complète : si une formule dont la forme clausale est Δ est insatisfiable alors il existe une dérivation telle que $\Delta \rhd \{\ \}$.

Résolution pour L_{pred} : terminaison

- Pour L_{prop}, on a vu en "Logique 1" un algorithme qui termine : on calcule des résolvantes tant que la clause vide n'a pas été générée ou qu'on génère de nouvelles clauses. En cas de saturation et d'absence de la clause vide, l'ensemble de clauses est **satisfiable**

- Pour L_{pred}, il se peut que cet algorithme ne termine pas. C'est le cas en particulier lorsque l'ensemble des clauses est satisfiable

- Alonzo Church a montré en 1936, à partir des travaux d'Alan Turing, qu'il n'existe pas de procédure qui permet de *décider* la satisfiabilité d'une formule de L_{pred}

Résolution pour L_{pred} : terminaison

Exemple d'un calcul itératif de résolvante ne terminant pas :

Soit $\Delta = \{\{P(x)\}, \{\neg P(y), P(f(y))\}\}$

- L'application de (R) sur $\{P(x)\}$ et $\{\neg P(y), P(f(y))\}$ avec $\sigma_1 = [x \leftarrow y]$ donne une nouvelle clause : $\{P(f(y))\}$
- On doit renommer $\{\neg P(y), P(f(y))\}$ avec une variable fraîche y' afin de pouvoir la réutiliser dans le processus
- L'application de (R) sur $\{P(f(y))\}$ et $\{\neg P(y'), P(f(y'))\}$ avec $\sigma_2 = [y' \leftarrow f(y)]$ donne une nouvelle clause : $\{P(f(f(y)))\}$
- L'application de (R) sur $\{P(f(f(y)))\}$ et $\{\neg P(y'), P(f(y'))\}$ avec $\sigma_3 = [y' \leftarrow f(f(y))]$ donne une nouvelle clause : $\{P(f(f(f(y))))\}$
- L'application de (R) sur $\{P(f(f(f(y))))\}$ et $\{\neg P(y'), P(f(y'))\}$ avec $\sigma_4 = [y' \leftarrow f(f(f(y)))]$ donne une nouvelle clause : $\{P(f(f(f(f(y)))))\}$
- ...

Fin de cette partie : qu'est-ce qui est exigible à l'examen ?

Check-list des attendus à l'examen :

- ☐ Savoir mettre en forme prénexe une formule
- ☐ Savoir skolémiser une formule
- ☐ Savoir mettre en forme universelle close une formule
- ☐ Savoir passer une formule en forme clausale
- ☐ Savoir calculer des dérivations dans le cadre de la résolution
- ☐ Le cours doit être relu et compris

Élimination des quantificateurs

Plan

Modèles : Logique propositionnelle

Rappel :

- Un modèle d'une formule A est une valuation v telle que $v(A) = 1$
- Un modèle d'un ensemble de formules H est une valuation v telle que pour tout $A \in H$, $v(A) = 1$

Exemples

- $v_1(p) = 1$ et $v_1(q) = 0$ est un modèle de $p \vee \neg q$
- $v_2(p) = 1$ et $v_2(q) = 1$ est un autre modèle de $p \vee \neg q$
- $p \wedge \neg p$ n'a pas de modèle

Modèles : Logique des prédicats

Définitions (Structures)

- Une signature $S = (R_1, \ldots R_m; f_1, \ldots f_n)$ décrit les symboles (de relation ; de fonction) d'un langage.
- L'arité $ar(R)$ resp. $ar(f)$ d'un symbole est son nombre d'arguments.
- Pour une S-signature, une S-structure $\mathcal{A}$ est composée
  - d'un domaine A non vide. Notation : $A = dom(\mathcal{A})$
  - une relation $R^{\mathcal{A}} \subseteq A^{ar(R)}$ pour tout symbole de relation $R \in S$
  - une fonction $f^{\mathcal{A}} : A^{ar(f)} \to A$ pour tout symbole de fonction $f \in S$

Notation : souvent, on écrit

- une signature de la forme :
  $S = (R_1[ar(R_1)], \ldots R_m[ar(R_m)]; f_1[ar(f_1)], \ldots f_n[ar(f_n)])$
- une structure de la forme :
  $\mathcal{A} = (domaine; R_1, \ldots R_m; f_1, \ldots f_n)$

Modèles : Logique des prédicats

Définitions (Interprétations et modèles)

- Une interprétation $\mathcal{I}$ sur une signature S est un couple $(\mathcal{A}, v)$ où $\mathcal{A}$ est une S-structure et $v : \mathit{Var} \to \mathit{dom}(\mathcal{A})$ une interprétation des variables.
- $\mathcal{I}$ est un modèle d'une formule F si $\mathcal{I} \models F$
- $\mathcal{I}$ est un modèle d'un ensemble de formules H, écrit $\mathcal{I} \models H$, si $\mathcal{I} \models F$ pour tout $F \in H$

Notes :

- Les constantes sont assimilées à des fonctions 0-aires.
- La définition de $\mathcal{I} \models F$ se fait par récursion sur F avec entre autres :
  - $\mathcal{I} \models \neg F_1 := \mathcal{I} \not\models F_1$
  - $\mathcal{I} \models F_1 \wedge F_2 := (\mathcal{I} \models F_1 \text{ et } \mathcal{I} \models F_2)$

  voir détails cours Logique 1.

(Digression syntaxique : Surcharge)

Un opérateur est (syntaxiquement) surchargé s'il est appliqué à des types essentiellement différents.

En logique, l'opérateur $\models$ est surchargé :

- $H \models F$, où H est un ensemble de formules, F une formule
  Signification : tout modèle de H est aussi un modèle de F
- $\mathcal{I} \models F$, où $\mathcal{I}$ est une interprétation, F une formule
  Signification : voir transparent précédent
- $\mathcal{A} \models F$, où $\mathcal{A}$ est une structure, F une formule close (sans variable libre)
  Signification : équivalent à $(\mathcal{A}, v) \models F$ pour n'importe quel v.

Modèles : Exemple banal

Exemple : Soit $S = (<; +, \overline{0}, \overline{1}, \overline{2} \dots)$ avec $ar(<) = 2$, $ar(+) = 2$, $ar(\overline{0}) = ar(\overline{1}) = \dots = 0$

- Soit $\mathcal{N}_\infty = (\mathbb{N}, <^{\mathcal{N}_\infty}, +^{\mathcal{N}_\infty}, \overline{0}^{\mathcal{N}_\infty}, \overline{1}^{\mathcal{N}_\infty}, \dots)$ avec
  - $<^{\mathcal{N}_\infty}$ "inferieur" usuel sur les naturels : $\{(0,1),(0,2),(1,2)\dots\}$
  - $+^{\mathcal{N}_\infty}$ l'addition usuelle : $(0,0) \mapsto 0, \dots (2,3) \mapsto 5, \dots$
  - $\overline{0}^{\mathcal{N}_\infty}, \overline{1}^{\mathcal{N}_\infty}$ les nombres 0, 1, ...

  On a : $\mathcal{N}_\infty \models \overline{2} + \overline{3} < \overline{6}$
- Soit $\mathcal{N}_3 = (\{0,1,2\}, <^{\mathcal{N}_3}, +^{\mathcal{N}_3}, \overline{0}^{\mathcal{N}_3}, \overline{1}^{\mathcal{N}_3}, \dots)$ avec
  - $<^{\mathcal{N}_3} = \{(0,1),(0,2),(1,2)\}$
  - $+^{\mathcal{N}_3}$ l'addition modulo 3 : $(0,0) \mapsto 0, \dots (1,2) \mapsto 0, (2,2) \mapsto 1$
  - $\overline{n}^{\mathcal{N}_3} = n \mod 3$, par ex. $\overline{5}^{\mathcal{N}_3} = 2$

  On a : $\mathcal{N}_3 \not\models \overline{2} + \overline{3} < \overline{6}$ parce que $(\overline{2} + \overline{3})^{\mathcal{N}_3} = 2$ et $\overline{6}^{\mathcal{N}_3} = 0$
- Soit $\mathcal{B} = (\{0,1\}^*, <^{\mathcal{B}}, +^{\mathcal{B}}, \overline{0}^{\mathcal{B}}, \overline{1}^{\mathcal{B}}, \dots)$ avec
  - $<^{\mathcal{B}}$ "est préfixe de", par ex. $11 <^{\mathcal{B}} 110$ et $10 \not<^{\mathcal{B}} 11$
  - $+^{\mathcal{B}}$ concaténation
  - $\overline{n}^{\mathcal{B}}$ représentation binaire de n, par ex. $\overline{5}^{\mathcal{B}} = 101$

  On a : $\mathcal{B} \not\models \overline{2} + \overline{3} < \overline{6}$ parce que $(\overline{2} + \overline{3})^{\mathcal{B}} = 1011$ et $\overline{6}^{\mathcal{B}} = 110$

Modèles et théories

Fixons une signature S.

Soit H un ensemble de formules sur S et $\mathcal{A}$ une S-structure et $\mathcal{S}$ un ensemble de S-structures.

Définitions

- L'ensemble des modèles de H : $Mod(H) = \{\mathcal{A} \,.\, \mathcal{A} \models H\}$
- La théorie d'une structure : $Th(\mathcal{A}) = \{F \,.\, \mathcal{A} \models F\}$
- La théorie d'un ensemble de structures : $Th(\mathcal{S}) = \bigcap_{\mathcal{A} \in \mathcal{S}} Th(\mathcal{A})$

Démontrez :

- Anti-monotonie de Mod : $H_1 \subseteq H_2 \Longrightarrow Mod(H_2) \subseteq Mod(H_1)$
- Anti-monotonie de Th : $\mathcal{S}_1 \subseteq \mathcal{S}_2 \Longrightarrow Th(\mathcal{S}_2) \subseteq Th(\mathcal{S}_1)$
- $H \subseteq Th(Mod(H))$
- $\mathcal{S} \subseteq Mod(Th(\mathcal{S}))$

Fermeture déductive

Définitions

La fermeture déductive d'un ensemble de formules est définie par $Ded(H) = Th(Mod(H))$

Montrez : $Ded(H) = \{F . H \models F\}$

Ded est un opérateur "fermeture", avec la propriété :
$Ded(Ded(H)) = Ded(H)$

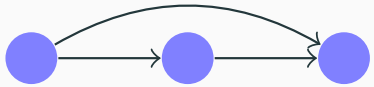
Illustration : exemple banal

- Modèles d'une formule (notation : $Mod(F)$ au lieu de $Mod(\{F\})$) :
  - Nous avons vu : $\mathcal{N}_\infty \in Mod(\overline{2} + \overline{3} < \overline{6})$ et $\mathcal{N}_3, \mathcal{B} \notin Mod(\overline{2} + \overline{3} < \overline{6})$
  - L'addition est commutative (aussi modulo 3), la concaténation ne l'est pas :
    $\mathcal{N}_\infty, \mathcal{N}_3 \in Mod(\forall x, y.\ x + y = y + x)$ et
    $\mathcal{B} \notin Mod(\forall x, y.\ x + y = y + x)$
- Théories (d'un ensemble) de structures :
  - Nous avons vu : $(\overline{2} + \overline{3} < \overline{6}) \in Th(\mathcal{N}_\infty), \notin Th(\mathcal{N}_3)$
  - $(\forall x, y.\ x + y = y + x) \in Th(\{\mathcal{N}_\infty, \mathcal{N}_3\})$

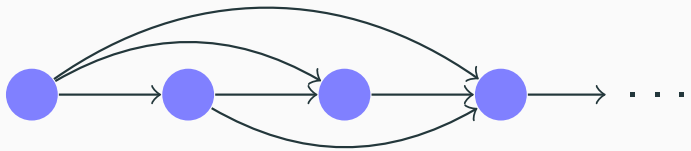
Axiomatisations : Introduction informelle (1)

Intention : caractériser un ensemble de structures $\mathcal{S}$ par un ensemble de formules H. Donc : $H = Th(\mathcal{S})$

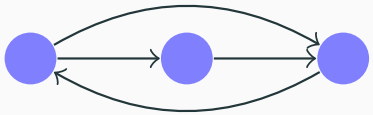
Exemple : Quelques structures appartenant à $\mathcal{S}$:



et



mais pas :

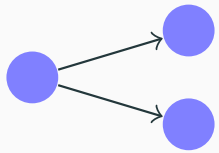


Axiomatisation possible : Structure $(\prec;)$ avec un symbole de relation d'ordre $\prec$, pas de symbole de fonction et les axiomes

- *transitive* (Trans) : $\forall x, y, z.\ x \prec y \wedge y \prec z \longrightarrow x \prec z$
- et *asymétrique* (Asym) : $\forall x, y.\ x \prec y \longrightarrow \neg(y \prec x)$

Axiomatisations : Introduction informelle (2)

Exemple continué : Est-ce qu'on admet des ordres partiels avec par ex. structure :



ou est-ce qu'on se limite à un ordre *total* avec axiome, en plus de Trans et Asym :

- *connectivité* (Conn) : $\forall x, y.\ x = y \vee x \prec y \vee y \prec x$

Et, si en plus, on souhaite admettre des structures de la forme :  on est tenté de rajouter l'axiome

- (Refl) : $\forall x.\ x \prec x$

Axiomatisations

Définition

Un ensemble de structures $\mathcal{S}$ est axiomatisable s'il existe un ensemble de formules H tel que $\mathcal{S} = Mod(H)$. Les formules de H sont appelés les axiomes de $\mathcal{S}$.

Remarque : Pour que $\mathcal{S} \neq \{\}$ soit axiomatisable, son ensemble d'axiomes doit être satisfiable.

Exercice : Montrez qu'un ensemble d'axiomes contenant ($\textsc{Asym}$) et ($\textsc{Refl}$) n'est pas satisfiable.

ℹ Indécidabilité Il est en général impossible de vérifier la satisfiabilité d'un ensemble d'axiomes.

Existence d'axiomatisations

Nous savons que $\mathcal{S} \subseteq Mod(Th(\mathcal{S}))$. Pourquoi pas prendre $Th(\mathcal{S})$ comme axiomatisation de $\mathcal{S}$?

- *Objection 1 :* typiquement, on ne peut pas générer $Th(\mathcal{S})$ par des moyens algorithmiques.
- *Objection 2 :* $Th(\mathcal{S})$ n'est pas fini, contredisant un besoin de minimalité.
- *Objection 3 :* Il peut y avoir des modèles non-standards $\mathcal{A} \in Mod(Th(\mathcal{S}))$ et $\mathcal{A} \notin \mathcal{S}$.

Pour une axiomatisation H d'un ensemble de structures $\mathcal{S}$, un modèle $\mathcal{A}$ est dit non-standard si $\mathcal{A} \in Mod(H)$ et $\mathcal{A} \notin \mathcal{S}$.

Arithmétique de Peano

Les axiomes *PA* de l'Arithmétique de Peano [2] sur la signature $(; +[2], *[2], s[1], 0[0])$:

- $\forall x.\ \neg(s(x) = 0)$ [s et 0 sont des constructeurs différents]
- $\forall x, y.\ s(x) = s(y) \longrightarrow x = y$ [injectivité de s]
- $\forall x.\ x + 0 = x$ [cas de base addition]
- $\forall x, y.\ x + s(y) = s(x + y)$ [cas héréditaire addition]
- $\forall x.\ x * 0 = 0$ [cas de base multiplication]
- $\forall x, y.\ x * s(y) = x * y + x$ [cas héréditaire multiplication]

et un *schéma d'induction* pour toute formule ϕ de premier ordre :

$$(\phi[0] \wedge (\forall y.\ \phi[y] \longrightarrow \phi[s(y)])) \longrightarrow \forall y.\ \phi[y]$$

2. Giuseppe Peano (1858-1932), mathématicien italien

Modèles non-standards de l'arithmétique

Notes :

- s est la fonction "successeur" correspondant à $+1$.
- Il y a une infinité de formules ϕ, l'ensemble PA est donc infini.
- Le modèle standard des nombres naturels $\mathcal{N}$ est dans $Mod(PA)$
- ... mais $Mod(PA)$ contient aussi des modèles non dénombrables.

ℹ Il n'y a pas axiomatisation de 1er ordre qui a $\mathcal{N}$ comme seul modèle.
(Il faudrait une axiomatisation avec un axiome d'induction de 2nd ordre)

Indépendance et minimalité d'une axiomatisation

Indépendance

- Une formule F est indépendante d'un ensemble d'axiomes Ax si $F \notin Ded(Ax)$ et $\neg F \notin Ded(Ax)$
- Une formule F dépend d'un ensemble d'axiomes Ax si $F \in Ded(Ax)$
- (Est-il possible d'avoir $F \in Ded(Ax)$ et $\neg F \in Ded(Ax)$?)

Minimalité

- Un ensemble d'axiomes Ax est minimal s'il n'existe pas de $F \in Ax$ avec $Ded(Ax - \{F\}) = Ded(Ax)$
  "Tout axiome est indispensable"

Axiomatisation de la théorie des groupes

La signature des groupes est : $(;\cdot, e, ^{-1})$ où

- (sans symbole de relation)
- $\cdot$ est l'opération (binaire) de groupe
- e est l'élément neutre
- $^{-1}$ est l'inverse (unaire)

Une axiomatisation (Ax_1) de la théorie des groupes est :

- *associativité :* $\forall x, y, z.\ (x \cdot y) \cdot z = x \cdot (y \cdot z)$
- *effet élément neutre :* $\forall x.\ x \cdot e = x$ et $\forall x.\ e \cdot x = x$
- *effet de l'inverse : :* $\forall x.\ x \cdot x^{-1} = e$ et $\forall x.\ x^{-1} \cdot x = e$

L'axiomatisation Ax_1 n'est pas minimale.
L'axiomatisation $Ax_2 = Ax_1 - \{\forall x.\ e \cdot x = x\}$ est équivalente :
$Ded(Ax_2) = Ded(Ax_1)$

Preuve : on montre, pout tout x :
$e \cdot x = (x \cdot x^{-1}) \cdot x = x \cdot (x^{-1} \cdot x) = x \cdot e = x$

Plan

Décidabilité

Définition (approximation grossière)

Une classe de problèmes P est décidable s'il existe un algorithme α tel que pour tout $p \in P$

- $\alpha(p) = 1$ après un temps de calcul fini si p admet une solution
- $\alpha(p) = 0$ après un temps de calcul fini si p n'admet pas de solution

Ici :

- classe de problèmes : ensemble des formules de la logique du 1er ordre
- admet une solution : est valide

ℹ Résultat : La logique du 1er ordre est indécidable.

Plus précisément : Le problème de *validité* des formules de 1er ordre est indécidable.

Montrez que le problème de *satisfiabilité* est aussi indécidable.

Preuves : procédures généralistes et dédiées

Procédures généralistes :

- *Exemples :* Résolution et Déduction Naturelle
- Applicables à toute formule F
  (peut-être après pré-traitement : conversion en forme clausale . . .)
- Devraient être correctes et complètes : $\vdash F \iff \models F$
- Indécidabilité : tentative de preuve d'une formule
  - *valide* réussit après un temps fini
  - *non valide* risque de continuer éternellement

Preuves : procédures généralistes et dédiées

Procédures spécialisées :

- Limitées à une classe de formules appartenant à une *théorie*
- de préférence décidable
- souvent particulièrement importante pour la vérification de programmes.

Exemples : Théories des

- nombres entiers ; réels
- tableaux ; chaînes de caractères
- ordres
- fonctions non interprétées ...

Remarque (surcharge du mot *theórie*) :

- Nous avons vu : *Th* ("théorie de structures"), fonction appliquée à (un ensemble de) structures
- Ici : Ensemble de formules possédant des symboles (= signature) avec une interprétation spécifique

Procédure de décision

Une procédure de décision pour une théorie T prend une formule F et essaie de décider F dans T.

Résultats possibles :

- F est vraie dans T : $T \models F$ (certificat *par ex.* dérivation de $T \vdash F$)
- F n'est pas vraie dans T : $T \not\models F$
  (certificat *par ex.* contre-modèle $\mathcal{M}$ avec $\mathcal{M} \models T$ mais $\mathcal{M} \not\models F$

Exemple :

- $\forall x.\, \exists y.\, y < x$ est vraie dans la théorie des nombres entiers $\mathbb{Z}$
- $\forall x.\, \exists y.\, y < x$ n'est pas vraie dans la théorie des nombres naturels $\mathbb{N}$
  Contre-exemple : $x = 0$

Différentes *méthodes d'implantation*, par exemple :

- Résolution (logique propositionnelle)
- Automates (logiques temporelles ; arithmétique)
- Élimination de quantificateurs

Élimination de quantificateurs

Une procédure d'élimination de quantificateurs

- convertit une formule F en une formule F' équivalente sans quantificateurs.
- souvent relatif à une théorie :
  - $(\forall x.\, \exists y.\, y < x) \leftrightarrow \top$ dans $\mathbb{Z}$
  - $(\forall x.\, \exists y.\, y < x) \leftrightarrow \bot$ dans $\mathbb{N}$

Remarques :

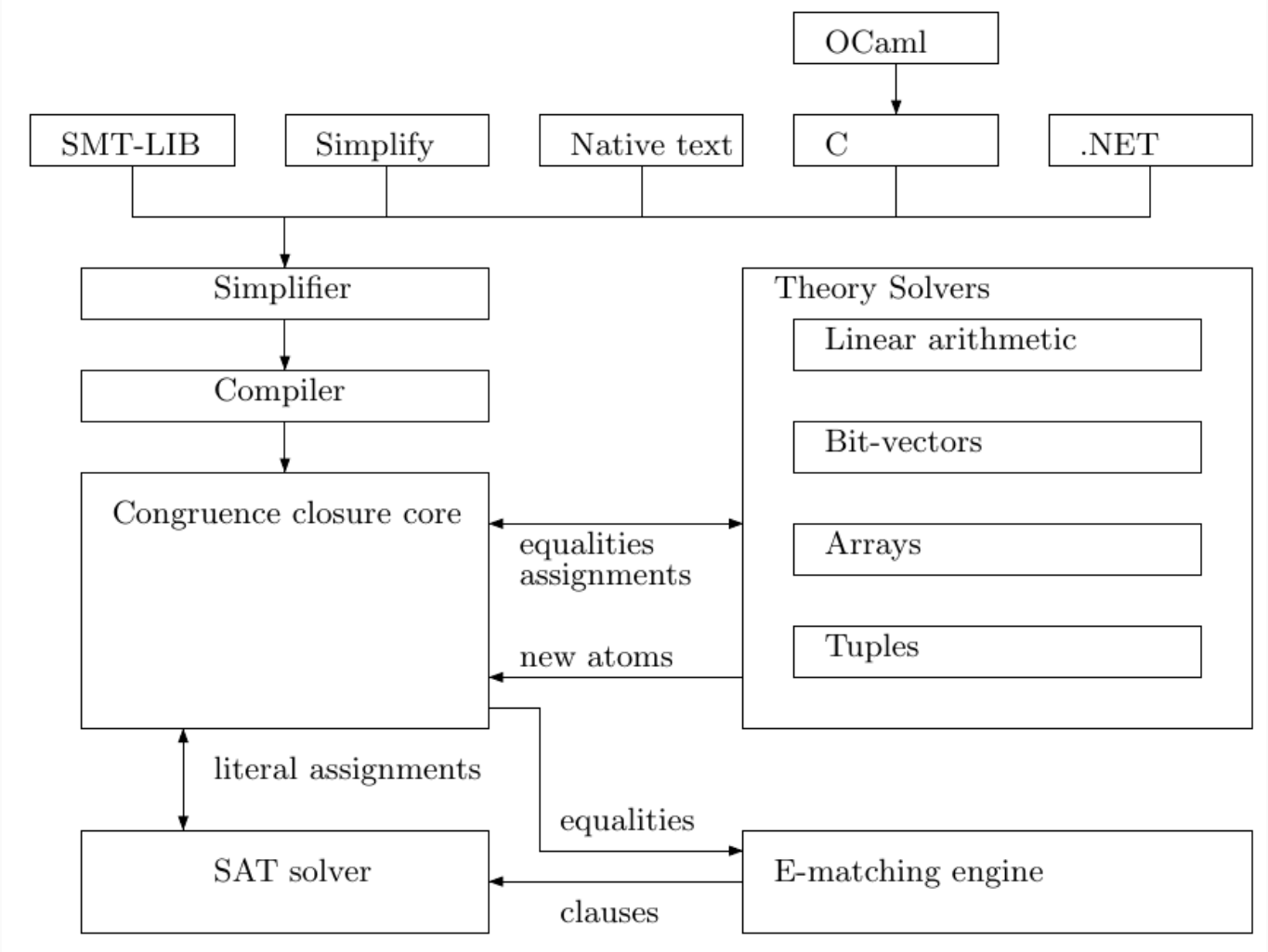
- Il y a des théories qui n'admettent pas d'élimination de quantificateurs (voir exos de TD)
- Élim. de quantif. *peut* être une procédure de décision
- ... mais pas forcément (*exemple :* logique propositionnelle)

SMT solveurs

Satisfiability Modulo Theories :

- Extension des SAT (satisfiability) solveurs
- Organise l'interaction entre SAT et des solveurs spécifiques
- Utilisé par des outils de vérification de programmes (Why3 ...)
- Standardisation de l'interface via le langage SMT-LIB (`http://smtlib.cs.uiowa.edu/`)
- Quelques SMT solveurs connus :
  - Alt-Ergo (`https://alt-ergo.ocamlpro.com/`)
  - CVC5 (`https://cvc5.github.io/`)
  - veriT (`https://www.verit-solver.org/`)
  - Yices (`https://yices.csl.sri.com/`)
  - Z3 (`https://github.com/Z3Prover/z3/wiki`)

SMT solveurs : architecture



(de l'article : *Z3 : An efficient SMT Solver*, par L. de Moura et N. Bjørner)

SMT solveurs : interaction entre composants

DO : théorie des ordres denses sans bornes

A titre d'exemple : une procédure de décision pour la théorie des ordres denses sans bornes, *DO*, pour une relation $\prec$ avec l'axiomatisation :

- ($\textsc{Trans}$) : $\forall x, y, z.\ x \prec y \wedge y \prec z \longrightarrow x \prec z$ [transitive]
- ($\textsc{Asym}$) : $\forall x, y.\ x \prec y \longrightarrow \neg(y \prec x)$ [asymétrique]
- ($\textsc{Conn}$) : $\forall x, y.\ x = y \vee x \prec y \vee y \prec x$ [connectivité]
- ($\textsc{Dense}$) : $\forall x, y.\ x \prec y \longrightarrow \exists z.\ x \prec z \wedge z \prec y$ [ordre dense]
- ($\textsc{SExtr}$) $\forall x.\ \exists y, z.\ y \prec x \wedge x \prec z$ [sans extrema]

Notes :

- L'ordre $\prec$ n'est pas réflexif : $\neg(x \prec x)$ à cause de ($\textsc{Asym}$)
- ($\textsc{Conn}$) est une forme faible de ($\textsc{Total}$) pour ordres asymétriques
- Voir exos de TD pour variante de la procédure pour ordres partiels

Éléments notables de $\mathit{Mod}(\mathit{DO})$:

- $\mathbb{Q}$ ou $\mathbb{R}$ avec ordre $<$
- Intervalles ouverts $(\mathit{inf}, \mathit{sup}) \subseteq \mathbb{Q}$ ou $\mathbb{R}$ avec ordre $<$

Procédure de décision pour DO : Aperçu

Principe : La procédure prend une formule F et la convertit en $\top$ ou $\bot$ par élimination de quantificateurs

Exigences :

- F doit être close (sans variables libres)
- Sinon : fermer avec $\forall x_0 \ldots x_n.\ F$ pour $\{x_0 \ldots x_n\} = fv(F)$
- Les seules symboles relationels permis sont : ordre $\prec$, égalité $=$; pas de fonctions ou constantes.

Démarche :

1. Convertir F en forme prénexe
2. Convertir quantif. universels en existentiels : $(\forall x.\phi) \leftrightarrow \neg(\exists x.\ \neg\phi)$
3. Éliminer les quantificateurs à partir de l'intérieur

Élimination de quantificateurs : Prétraitement

Simplification d'une formule de la forme $\exists x.\ \phi$

1. Tirer les négations à l'intérieur : forme normale négative
2. Éliminer les négations devant les relations :
   2.1 $\neg(z \prec z') \leftrightarrow (z = z' \vee z' \prec z)$
   2.2 $\neg(z = z') \leftrightarrow (z \prec z' \vee z' \prec z)$
3. Transformer en forme normale disjonctive : $\phi \leftrightarrow \bigvee_j \psi_j$
4. Tirer la quantification existentielle à l'intérieur de la disjonction : $\exists x.\ \phi \leftrightarrow \bigvee_j (\exists x.\ \psi_j)$

Les quantifications existentielles intérieures ont maintenant le format $\exists x.\ \psi$, où ψ est une conjonction de relations $u \prec v$ ou $u = v$

Élimination de quantificateurs : Suppression de variable

Suppression de la quantification existentielle de $\exists x.\ \psi$:

- Si $x \notin fv(\psi)$: $(\exists x.\ \psi) \leftrightarrow \psi$
- Si ψ contient le terme $x < x$: $(\exists x.\ \psi) \leftrightarrow \bot$
- Sinon, regrouper les termes de ψ en :
  $(\bigwedge_i x \prec u_i) \wedge (\bigwedge_j v_j \prec x) \wedge (\bigwedge_k w_k = x) \wedge \chi$ avec $x \notin fv(\chi)$
- Si $(\bigwedge_k w_k = x)$ présent : choisir une variable w_0 parmi les w_k, et
  $(\exists x.\ \psi) \leftrightarrow (\bigwedge_i w_0 \prec u_i) \wedge (\bigwedge_j v_j \prec w_0) \wedge (\bigwedge_k w_k = w_0) \wedge \chi$
- Sinon, si $(\bigwedge_i x \prec u_i)$ et $(\bigwedge_j v_j \prec x)$ présents dans ψ :
  $(\exists x.\ \psi) \leftrightarrow \bigwedge_{i,j} v_j \prec u_i \wedge \chi$
- Sinon, si uniquement $(\bigwedge_i x \prec u_i)$ ou $(\bigwedge_j v_j \prec x)$ présent dans ψ :
  $(\exists x.\ \psi) \leftrightarrow \chi$

Post-traitement : Simplifier conjonctions/disjonctions

Élimination de quantificateurs : Exemple

Formule originale : $F = \forall x.\, \exists y.\, \neg(x < y) \wedge (\forall z.\, y < z \longrightarrow x < z)$

Démarches préliminaires :

1. Conversion en forme prénexe :
   $\forall x.\, \exists y.\, \forall z.\, \neg(x < y) \wedge (y < z \longrightarrow x < z)$
2. Conversion quantif. universels :
   $\neg(\exists x.\, \neg(\exists y.\, \neg(\exists z.\, \neg(\neg(x < y) \wedge (y < z \longrightarrow x < z)))))$

Éliminiation de la quantification $\exists z.\, \neg(\neg(x < y) \wedge (y < z \longrightarrow x < z))$

- Prétraitement :
  1. NNF : $\exists z.\, (x < y) \vee (y < z \wedge \neg(x < z))$
  2. Élim nég. avant $<$: $\exists z.\, (x < y) \vee (y < z \wedge (x = z \vee z < x))$
  3. DNF : $\exists z.\, (x < y) \vee (y < z \wedge x = z) \vee (y < z \wedge z < x)$
  4. Quantif. à l'intérieur :
     $(\exists z.\, (x < y)) \vee (\exists z.\, (y < z \wedge x = z)) \vee (\exists z.\, (y < z \wedge z < x))$

Élimination de quantificateurs : Exemple (ctd)

Simplification de chaque disjonction de
$\psi_z = (\exists z.\ (x < y)) \vee (\exists z.\ (y < z \wedge x = z)) \vee (\exists z.\ (y < z \wedge z < x))$

- $(\exists z.\ (x < y)) \leftrightarrow (x < y)$ [parce que $z \notin fv(x < y)$]
- $(\exists z.\ (y < z \wedge x = z)) \leftrightarrow (y < x \wedge x = x) \leftrightarrow (y < x)$
- $(\exists z.\ (y < z \wedge z < x)) \leftrightarrow (y < x)$

Donc : $\psi_z \leftrightarrow (x < y \vee y < x)$

Éliminiation de la quantification $(\exists y.\ \neg(x < y \vee y < x))$

Prétraitement : $(\exists y.\ \neg(x < y) \wedge \neg(y < x)) \leftrightarrow$
$(\exists y.\ (x = y \vee y < x) \wedge (y = x \vee x < y)) \leftrightarrow$
$(\exists y.\ (x = y) \vee (x = y \wedge x < y) \vee (y < x \wedge y = x) \vee (y < x \wedge x < y))$

Élimination de quantificateurs : Exemple (ctd)

Simplification de chaque disjonction de $\psi_y = (\exists y.\ (x = y)) \vee \ldots$

- $(\exists y.\ (x = y)) \leftrightarrow (x = x) \leftrightarrow \top$ [On pourrait s'arrêter ici ...]
- $(\exists y.\ x = y \wedge x < y) \leftrightarrow (x < x) \leftrightarrow \bot$
- $(\exists y.\ y < x \wedge y = x) \leftrightarrow (x < x) \leftrightarrow \bot$
- $((\exists y.\ y < x \wedge x < y)) \leftrightarrow (x < x) \leftrightarrow \bot$

Donc : $\psi_y = \top$

Le reste de la formule : $\neg(\exists x.\ \neg\psi_y) \leftrightarrow \neg(\exists x.\ \bot) \leftrightarrow \top$

La formule originale est donc prouvée : $F \leftrightarrow \top$

Fin de cette partie : qu'est-ce qui est exigible à l'examen ?

Check-list des attendus à l'examen :

-