UE Logique 2

Preuves en Logiques

Frédéric Maris, Ralph Matthes, Jean-Baptiste Raclet,
Jan-Georg Smaus et Martin Strecker

L2 - Année universitaire 2019-20

Université de Toulouse / IRIT

UE Logique 2 – L2 Informatique S3

- **Responsable administratif** : Jean-Baptiste Raclet
- **Intervenants** :
  - G1 : Martin Strecker – strecker@irit.fr
  - G2 : Frédéric Maris – maris@irit.fr
  - G3 : Martin Strecker – strecker@irit.fr
  - G4 : Ralph Matthes – matthes@irit.fr
  - G5 : Jean-Baptiste Raclet – raclet@irit.fr
  - G6 : Jan-Georg Smaus – smaus@irit.fr
- **Volume horaire** : 30h (1 à 2 séances hebdomadaires)
- 3 ECTS
- **Page moodle** : `http://moodle.univ-tlse3.fr/course/view.php?id=684`

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 :

- Éléments structurants d'une preuve
  $\rightsquigarrow$ *implication, équivalence, contraire, ...*
- 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

Bibliographie

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

F. Lepage.
Eléments de logique contemporaine.
PU Montréal, 2010.

Pré-requis

Éléments vus en L1 :

- **UE Logique 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
- **UE Mathématiques discrètes**

Pré-requis

Si vous n'avez pas ces pré-requis...

- Le support du cours **Logique 1** est disponible sur la page moodle de l'UE **Logique 2**
- 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

Modalités de contrôle des connaissances

- **Première session** :
  - **CCC** : un devoir sur table le mardi 5 novembre 2019
    $\longrightarrow$ 30% de la note de l'UE
  - **CT** : un devoir sur table
    $\longrightarrow$ 70% de la note de l'UE
- **Session de rattrapage** : la note de **CCC** est reportée ; un nouveau devoir sur table est organisé pour rattraper le **CT**

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. $\square$

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. $\square$

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. $\square$

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 $\epsilon = \frac{x}{2}$. $\square$

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 $a^b = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^{(\sqrt{2}.\sqrt{2})} = \sqrt{2}^2 = 2.$ $\square$

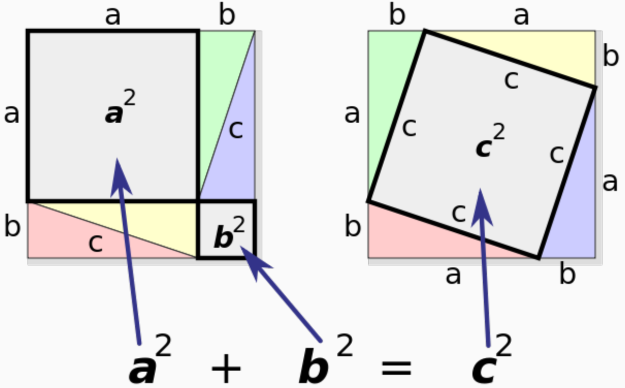
1.On admet ici que $\sqrt{2}$ est irrationnel ; ceci est facilement démontrable par l'absurde.

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. $\square$

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 la **satisfiabilité** 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 par induction

Intuition derrière l'induction

- Voici une définition **inductive** d'une poupée russe :
  - Ceci est une poupée russe...

  - Étant donnée une poupée russe, si on l'enferme dans une figurine creuse de poupée alors on obtient une poupée russe

- La définition précédente en 2 points permet de construire une série comportant un nombre **arbitraire** de poupées gigognes !

Induction : méta-théorie

- L'induction est un concept très général et fondamental dans les mathématiques, l'informatique et la logique.
- Dans ce cours, quand nous étudions l'induction, nous faisons de la méta-théorie.

Domaines d'application de l'induction

- Ici nous regardons quatre exemples :
  1. Les entiers naturels en base 10. Ex. : 0, 1, 7, 42, ...
  2. Les expressions arithmétiques parenthésées avec variables sur les entiers et $+$, $-$, $\times$. Ex. :$((2 + x) \times y)$
  3. Les entiers naturels représentés comme 0, $S0$, $SS0$, ...
  4. Les formules de la logique propositionnelle
- Les trois premiers nous intéressent pour acquérir de l'intuition sur le concept d'induction.
- Le quatrième exemple nous intéresse à titre propre puisque c'est un cours de logique.

Domaines d'application de l'induction

- Ici nous regardons quatre exemples :
  1. Les entiers naturels en base 10. Ex. : 0, 1, 7, 42, ...
  2. Les expressions arithmétiques parenthésées avec variables sur les entiers et $+$, $-$, $\times$. Ex. :$((2+x)\times y)$
  3. Les entiers naturels représentés comme 0, $S0$, $SS0$, ...
  4. Les formules de la logique propositionnelle
- Les trois premiers nous intéressent pour acquérir de l'intuition sur le concept d'induction.
- Le quatrième exemple nous intéresse à titre propre puisque c'est un cours de logique.
- Dans ce cours nous nous intéressons particulièrement à la *preuve*, mais en fait, nous verrons une "triade" :
  - **définition inductive**
  - **fonctions récursives**
  - **preuves par induction**

Exemple 1 : Nombres entiers naturels Nb en base 10

Problème : comment définir l'ensemble des nombres entiers naturels Nb en base 10, ex. : 45, 0, ...($\sim$ trouver des règles pour les constructions permises) ?

Exemple 1 : Nombres entiers naturels *Nb* en base 10

Problème : comment définir l'ensemble des nombres entiers naturels *Nb* en base 10, ex. : 45, 0, ...($\sim$ trouver des règles pour les constructions permises) ?

Définition inductive (ou par induction)

L'ensemble des entiers naturels *Nb* est le plus petit ensemble qui respecte les conditions suivantes :

- $0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \in Nb$ $\leftarrow$ élément de base
- Si $x \in Nb$ et $a \in \{0, .., 9\}$ alors $xa \in Nb$ $\leftarrow$ règle d'assemblage

Attention ! Les lettres x et a ne font pas partie des caractères autorisés dans un nombre ! Ces lettres représentent respectivement un nombre (x) et un chiffre (a). Ce sont des **méta-variables**.

Exemple 2 : Expressions arithmétiques

- Expressions arithmétiques parenthésées avec variables (x, y, z, ...) sur les entiers et les opérateurs binaires $+$, $-$, $\times$ ou unaire $-$.
  Exemple : $((2 + x) \times y)$

Exemple 2 : Expressions arithmétiques

- Expressions arithmétiques parenthésées avec variables (x, y, z, ...) sur les entiers et les opérateurs binaires $+$, $-$, $\times$ ou unaire $-$.
  Exemple : $((2 + x) \times y)$

Définition inductive

L'ensemble *EXP* de ces expressions (*VAR* est un ensemble de variables, *Nb* les entiers naturels définis auparavant) est le *plus petit* ensemble qui respecte les conditions suivantes :

1. **Variable :** Pour tout $x \in VAR$, $x \in EXP$ $\leftarrow$ élément de base
2. **Constante :** Pour tout $c \in Nb$, $c \in EXP$ $\leftarrow$ élément de base
3. **Opposé :** Si $A \in EXP$ alors $(-A) \in EXP$ $\leftarrow$ règle d'assemblage
4. **Addition, Soustraction, Multiplication** : Si $A \in EXP$ et $B \in EXP$ alors $(A + B) \in EXP$, $(A - B) \in EXP$, $(A \times B) \in EXP$
   $\hookleftarrow$ règle d'assemblage

- Cette dernière définition fait intervenir des *variables* représentant un nombre entier et des *méta-variables* (A et B) qui représentent des *expressions*.

Principe d'une définition inductive

- Une définition **inductive** rend précise l'idée que le tout est composé de parties, elles mêmes composées de parties plus petites, etc jusqu'aux parties indivisibles qui sont les éléments de base.
- "*EXP* est le *plus petit* ensemble ..." :
  - $(1 \bowtie x) \notin EXP$, parce qu'aucune règle ne génère $(1 \bowtie x)$

Principe d'une définition inductive

- Une définition **inductive** rend précise l'idée que le tout est composé de parties, elles mêmes composées de parties plus petites, etc jusqu'aux parties indivisibles qui sont les éléments de base.
- "*EXP* est le *plus petit* ensemble ..." :
  - $(1 \bowtie x) \notin EXP$, parce qu'aucune règle ne génère $(1 \bowtie x)$
  - ☕ $\notin EXP$,

Principe d'une définition inductive

- Une définition **inductive** rend précise l'idée que le tout est composé de parties, elles mêmes composées de parties plus petites, etc jusqu'aux parties indivisibles qui sont les éléments de base.
- "*EXP* est le *plus petit* ensemble …" :
  - $(1 \bowtie x) \notin \mathit{EXP}$, parce qu'aucune règle ne génère $(1 \bowtie x)$
  -  $\notin \mathit{EXP}$,  $\notin \mathit{EXP}$,

Principe d'une définition inductive

- Une définition **inductive** rend précise l'idée que le tout est composé de parties, elles mêmes composées de parties plus petites, etc jusqu'aux parties indivisibles qui sont les éléments de base.
- "*EXP* est le *plus petit* ensemble ..." :
  - $(1 \bowtie x) \notin EXP$, parce qu'aucune règle ne génère $(1 \bowtie x)$
  - ☕ $\notin EXP$, 🚲 $\notin EXP$, ⚽ $\notin EXP$, parce que ...

Principe d'une définition inductive

- Une définition **inductive** rend précise l'idée que le tout est composé de parties, elles mêmes composées de parties plus petites, etc jusqu'aux parties indivisibles qui sont les éléments de base.
- "*EXP* est le *plus petit* ensemble ..." :
  - $(1 \bowtie x) \notin EXP$, parce qu'aucune règle ne génère $(1 \bowtie x)$
  - ☕ $\notin EXP$, 🚲 $\notin EXP$, ⚽ $\notin EXP$, parce que ...
  - $1 + (x - 3)) \notin EXP$, parce qu'il manque une parenthèse "(".

Principe d'une définition inductive

- Une définition **inductive** rend précise l'idée que le tout est composé de parties, elles mêmes composées de parties plus petites, etc jusqu'aux parties indivisibles qui sont les éléments de base.
- "*EXP* est le *plus petit* ensemble ..." :
  - $(1 \bowtie x) \notin EXP$, parce qu'aucune règle ne génère $(1 \bowtie x)$
  - ☕ $\notin EXP$, 🚲 $\notin EXP$, ⚽ $\notin EXP$, parce que ...
  - $1 + (x - 3)) \notin EXP$, parce qu'il manque une parenthèse "(".
- L'exemple le plus simple de définition inductive est celui des nombres entiers naturels codés en base 1. Cela nous conduit à *l'exemple 3*, les nombres naturels représentés comme 0, $S0$, $SS0$, ...

Exemple 3 : les nombres naturels

- Nous avons déjà vu les nombres naturels comme 0, 1, 7, 42, ...
- Pour des considérations théoriques, c'est préférable d'utiliser la représentation suivante :
  - 0 est codé 0, et
  - le suivant d'un nombre n par Sn (S comme "successeur").

  Cela simplifie beaucoup certaines démonstrations et définitions (pas de table d'addition/multiplication à connaître)

Nombres naturels : Définition inductive

Les nombres naturels comme ensemble inductif :

Définition inductive

L'ensemble des entiers naturels Nat est le plus petit ensemble qui respecte les conditions suivantes :

1. *Zéro :* $0 \in Nat$ $\leftarrow$ élément de base
2. *Successeur :* Si $n \in Nat$, alors $(Sn) \in Nat$ $\leftarrow$ règle d'assemblage

- Comme ci-dessus, la précision "plus petit ensemble" sert à éliminer les ensembles qui respecteraient les items 1 et 2 :
  - contiendraient donc $0, S0, SS0, \ldots$
  - mais contiendraient aussi des éléments indésirables comme $\bowtie, S \bowtie, SS \bowtie, \ldots$ ou $0, X0, XX0, SX0, SXX0, SSX0, \ldots$
- On omettra souvent les parenthèses autour de (Sn).
- n est une méta-variable.

Nombres naturels : Définition inductive (2)

$$\bullet \xrightarrow{+1} \bullet \xrightarrow{+1} \bullet \xrightarrow{+1} \bullet$$

représentation décimale	0	1	2	3
représentation unaire	0	$S0$	$SS0$	$SSS0$

Nombres naturels : Définition récursive d'une fonction

Récursion sur les nombres naturels :

Une fois les nombres naturels définis, on peut créer des fonctions s'appliquant à eux :

Exemple : La fonction d'addition $n + m$ peut être définie par :

- *cas Zéro :* $(0 + m) = m$
- *cas Successeur :* $((Sn) + m) = S(n + m)$

Exercice

Définir par récursion sur n les fonctions modulo 2 de n ($m2(n)$), multiplication par n et $\Sigma_{i=0}^{n}$ à l'aide de 0, S et $+$.

Principe d'induction sur les nombres naturels :

Les nombres naturels ainsi définis partagent une caractéristique, une propriété donnée, si :

- (**base**) c'est le cas pour 0 ;
- (**hérédité**) et quand c'est le cas pour un nombre, c'est aussi le cas pour son successeur.

Nombres naturels : Preuves par induction (2)

Schéma d'induction sur les nombres naturels :

Soit $\mathcal{P}$ une propriété :

- **base (Zéro) :** si $\mathcal{P}(0)$
- **hérédité (Successeur) :** et si pour tout n, $\mathcal{P}(n)$ implique $\mathcal{P}(Sn)$

Alors on peut conclure $\forall n.\ \mathcal{P}(n)$ est vrai.

Nombres naturels : Exemple d'une preuve par induction

Théorème

0 est l'élément neutre à droite de l'addition : $\forall n.(n + 0 = n)$

1. *Identifier la propriété $\mathcal{P}$:* Ici : $\mathcal{P}(n) \equiv (n + 0 = n)$
2. *Preuve pour le cas de base :* Montrer : $\mathcal{P}(0)$, donc : $0 + 0 = 0$ (*cas Zéro de* $+$)
3. *Preuve d'hérédité :* Montrer : si $\mathcal{P}(n)$, alors $\mathcal{P}(Sn)$
   - Supposer : $n + 0 = n$ (Hypothèse d'induction)
   - En déduire : $(Sn) + 0 = Sn$.
     Calcul :
     - $(Sn) + 0 = S(n + 0)$ (*Cas Hérédité de* $+$)
     - $= Sn$ (*Hypothèse d'induction*)

Exercices

- Montrer que $\forall m.\forall n.m + (Sn) = S(m + n)$ (induction sur m)
- Utiliser ce résultat pour montrer la commutativité de l'addition : $\forall n.\forall m.(n + m) = (m + n)$

Exemple 4 : Formules propositionnelles

En "Logique 1", les **formules propositionnelles** ont été définies *informellement* de la façon suivante :

Ingrédients

- Des variables propositionnelles dans un ensemble $PROP$: p, q, $\ldots$,
- Des connecteurs : $\wedge$, $\vee$, $\longrightarrow$, $\neg$,
- Des parenthèses,
- Des symboles : $\perp$ (proposition "toujours fausse")

Recette pour obtenir l'ensemble des formules $FORM$

À l'aide de ces ingrédients, en liant les propositions par des connecteurs, on obtient des formules, qui sont associées à des énoncés.

Définition inductive des formules de L_{prop}

Définition

FORM est le plus petit ensemble qui satisfait les conditions :

1. **Variable propositionnelle :** Pour tout $p \in PROP$, $p \in FORM$
2. **Constante "faux" :** $\bot \in FORM$ $\leftarrow$ éléments de base $\rightharpoonup$
3. **Négation :** Si $A \in FORM$,
   alors $(\neg A) \in FORM$ $\leftarrow$ règle d'assemblage
4. **Conjonction** ("et") : Si $A \in FORM$ et $B \in FORM$,
   alors $(A \wedge B) \in FORM$ $\leftarrow$ règle d'assemblage
5. **Disjonction** ("ou") : Si $A \in FORM$ et $B \in FORM$,
   alors $(A \vee B) \in FORM$ $\leftarrow$ règle d'assemblage
6. **Implication** ("si ...alors") : Si $A \in FORM$ et $B \in FORM$,
   alors $(A \longrightarrow B) \in FORM$ $\leftarrow$ règle d'assemblage

Attention ! Nous avons à la fois des variables (propositionnelles) pour les propositions (p, q, ...) et des méta-variables pour les formules (A, B, ...).

Définition inductive des formules (2)

Comprendre une définition inductive :

- $s \in PROP$
  - donc $s \in FORM$

Définition inductive des formules (2)

Comprendre une définition inductive :

- $s \in PROP$
  - donc $s \in FORM$
  - donc $(\neg s) \in FORM$

Définition inductive des formules (2)

Comprendre une définition inductive :

- $s \in PROP$
  - donc $s \in FORM$
  - donc $(\neg s) \in FORM$
- $p \in PROP$, et donc $p \in FORM$

Définition inductive des formules (2)

Comprendre une définition inductive :

- $s \in PROP$
  - donc $s \in FORM$
  - donc $(\neg s) \in FORM$
- $p \in PROP$, et donc $p \in FORM$
- $q \in PROP$, et donc $q \in FORM$

Définition inductive des formules (2)

Comprendre une définition inductive :

- $s \in PROP$
  - donc $s \in FORM$
  - donc $(\neg s) \in FORM$
- $p \in PROP$, et donc $p \in FORM$
- $q \in PROP$, et donc $q \in FORM$
- $r \in PROP$, et donc $r \in FORM$

Définition inductive des formules (2)

Comprendre une définition inductive :

- $s \in PROP$
  - donc $s \in FORM$
  - donc $(\neg s) \in FORM$
- $p \in PROP$, et donc $p \in FORM$
- $q \in PROP$, et donc $q \in FORM$
- $r \in PROP$, et donc $r \in FORM$
- donc $(q \vee r) \in FORM$

Définition inductive des formules (2)

Comprendre une définition inductive :

- $s \in PROP$
  - donc $s \in FORM$
  - donc $(\neg s) \in FORM$
- $p \in PROP$, et donc $p \in FORM$
- $q \in PROP$, et donc $q \in FORM$
- $r \in PROP$, et donc $r \in FORM$
- donc $(q \vee r) \in FORM$
- donc $(p \wedge (q \vee r)) \in FORM$

Définition inductive des formules (2)

Comprendre une définition inductive :

- $s \in PROP$
  - donc $s \in FORM$
  - donc $(\neg s) \in FORM$
- $p \in PROP$, et donc $p \in FORM$
- $q \in PROP$, et donc $q \in FORM$
- $r \in PROP$, et donc $r \in FORM$
- donc $(q \vee r) \in FORM$
- donc $(p \wedge (q \vee r)) \in FORM$
- et donc finalement $((p \wedge (q \vee r)) \longrightarrow (\neg s)) \in FORM$

Fonction récursive sur les formules

- **Remarque** : dans la suite, on identifiera par L_{prop} l'ensemble des formules précédent (au lieu de $FORM$)
- La structure inductive d'une définition peut être exploitée dans le cadre de fonction récursive.
- En "Logique 1", nous avons déjà vu l'exemple de la définition d'une *valuation* (*interprétation*) qui peut être appliquée à toute formule :

Définition

On appelle *valuation* la fonction $v : L_{prop} \to \{0, 1\}$ telle que :

- **Variable :** $v(p) = v(p)$
- **Constante :** $v(\perp) = 0$
- **Négation :** $v(\neg A) = 1 - v(A)$
- **Conjonction :** $v(A \wedge B) = min(v(A), v(B))$
- **Disjonction :** $v(A \vee B) = max(v(A), v(B))$
- **Implication :** $v(A \longrightarrow B) = max(1 - v(A), v(B))$

Nous regarderons maintenant un autre exemple...

Fonction récursive : un autre exemple

Exemple : Nous définissons la fonction `nbp` (nombre de parenthèses) :

Définition

On appelle `nbp` la fonction récursive `nbp` : $L_{prop} \to \mathbb{N}$ telle que :

- **Variable** [eq. V] : `nbp(p) = 0`
- **Constante** [eq. C] : $\texttt{nbp}(\bot) = 0$
- **Négation** [eq. N] : $\texttt{nbp}((\neg A)) = \texttt{nbp}(A) + 2$
- **Conjonction** [eq. C] : $\texttt{nbp}((A \wedge B)) = \texttt{nbp}(A) + \texttt{nbp}(B) + 2$
- **Disjonction** [eq. D] : $\texttt{nbp}((A \vee B)) = \texttt{nbp}(A) + \texttt{nbp}(B) + 2$
- **Implication** [eq. I] : $\texttt{nbp}((A \longrightarrow B)) = \texttt{nbp}(A) + \texttt{nbp}(B) + 2$

Exemples d'application :

- $\texttt{nbp}(((p \vee q) \wedge r)) = 4$
- $\texttt{nbp}(((p \wedge q) \vee (\neg(r \vee \bot)))) = 8$

Preuves par induction pour les formules

La structure inductive d'une définition peut être exploitée pour en dériver un schéma de preuve, dite *par induction* :

Définition (schéma d'induction sur les formules)

Soit $\mathcal{P}$ une propriété à prouver sur toute formule de L_{prop} :

- **base (Variable) :** si $\mathcal{P}(p)$ est vrai
- **base (Constante) :** et si $\mathcal{P}(\bot)$ est vrai
- **hérédité (Négation) :** et si pour toute formule A, $\mathcal{P}(A)$ implique $\mathcal{P}((\neg A))$
- **hérédité (Conjonction) :** et si pour tout A, B, $\mathcal{P}(A)$ et $\mathcal{P}(B)$ implique $\mathcal{P}((A \wedge B))$
- **hérédité (Disjonction) :** et si pour tout A, B, $\mathcal{P}(A)$ et $\mathcal{P}(B)$ implique $\mathcal{P}((A \vee B))$
- **hérédité (Implication) :** et si pour tout A, B, $\mathcal{P}(A)$ et $\mathcal{P}(B)$ implique $\mathcal{P}((A \longrightarrow B))$

Alors on peut conclure : $\forall A \in FORM. \mathcal{P}(A)$ est vrai.

Preuve par induction : exemple

Exemple de preuve par induction :

Théorème

Toute formule de L_{prop} a un nombre pair de parenthèses :
$\forall f \in L_{prop}.\ nbp(f) \bmod 2 = 0$

Preuve : on pose $\mathcal{P}(A) \equiv (nbp(A) \bmod 2 = 0)$

↳ on commence par identifier la propriété à prouver

- *Cas de base (Variable) :*
  On montre $\mathcal{P}(p)$ vrai c'est-à-dire, $nbp(p) \bmod 2 = 0$.
  Par définition *[eq. V]*, $nbp(p) = 0$ et donc :
  $$nbp(p) \bmod 2 \ = \ 0 \bmod 2 \ = \ 0$$
- *Cas de base (Constante) :*
  On montre $\mathcal{P}(\perp)$ vrai c'est-à-dire, $nbp(\perp) \bmod 2 = 0$.
  Par définition *[eq. C]*, $nbp(\perp) = 0$ et donc :
  $$nbp(\perp) \bmod 2 \ = \ 0 \bmod 2 \ = \ 0$$

Preuve par induction : exemple (suite)

- *Hérédité (Négation) :*
  On montre si $\mathcal{P}(A)$ alors $\mathcal{P}((\neg A))$, c'est à dire :
  - Hypothèse d'induction *[eq. H]* : $nbp(A) \bmod 2 = 0$
  - Il faut montrer : $nbp((\neg A)) \bmod 2 = 0$

  Or :

$$\begin{aligned} nbp((\neg A)) \bmod 2 &= (nbp(A) + 2) \bmod 2 \text{ (par } \textit{[eq. N]}\text{)} \\ &= nbp(A) \bmod 2 \text{ (arithmétique)} \\ &= 0 \text{ (par } \textit{[eq. H]}\text{)} \end{aligned}$$

Preuve par induction : exemple (suite)

- *Hérédité (Conjonction) :*
  On montrer si $\mathcal{P}(A)$ et $\mathcal{P}(B)$, alors $\mathcal{P}((A \wedge B))$, c'est-à-dire :
  - Hypothèses d'induction
    - *[eq. H_1]* : $nbp(A) \bmod 2 = 0$
    - *[eq. H_2]* : $nbp(B) \bmod 2 = 0$
  - Il faut montrer : $nbp((A \wedge B)) \bmod 2 = 0$.

  Or :

$$
\begin{aligned}
nbp((A \wedge B)) \bmod 2 &= (nbp(A) + nbp(B) + 2) \bmod 2 \text{ (par \textit{[eq. C]})} \\
&= (nbp(A) \bmod 2 + nbp(B) \bmod 2) \bmod 2 \\
&= (0 + nbp(B) \bmod 2) \bmod 2 \text{ (par \textit{[eq. } H_1\text{]})} \\
&= (0 + 0) \bmod 2 \text{ (par \textit{[eq. } H_2\text{]})} \\
&= 0 \text{ (arithmétique)}
\end{aligned}
$$

- *Hérédité (Disjonction et Implication) :* similaire à la conjonction $\square$

Induction : résumé

La triade :

- **Ensembles inductifs :**
  - générés à partir d'éléments de base,
  - en appliquant des règles d'assemblage

- **Fonctions récursives** :
  - décomposent une structure inductive composée
  - s'arrêtent sur les éléments de base

  ... et synthétisent un résultat en remontant

- **Preuves par induction :** permettent de prouver qu'une propriété est satisfaite pour <u>tout</u> élément d'un ensemble inductif
  - si elle est satisfaite pour les éléments de base
  - si elle est héréditaire pour les éléments composés

Induction : ouverture

- L'induction est un concept très général qui se base sur la notion mathématique d'ordre bien fondé.
  $\rightsquigarrow$ cf. UE **Maths discrètes** en L1.

- La plupart des types de données peuvent être définis inductivement.
  Exemple : l'ensemble des listes d'entiers naturels L :
  - La liste vide $[\,] \in L$ $\qquad\leftarrow$ élément de base
  - Si $l \in L$ et $a \in Nat$ alors $[l]@[a] \in L$ $\qquad\leftarrow$ règle d'assemblage

  C'est à la base du paradigme de programmation fonctionnel.
  $\rightsquigarrow$ cf. UE **Prog. fonctionnelle, Intro. aux types abstraits** en L3.

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

Check-list des attendus à l'examen :

- ☐ Savoir appliquer une fonction récursive à une formule en détaillant les étapes de calcul
- ☐ Savoir définir de nouvelles fonctions récursives s'appliquant aux formules
- ☐ Savoir rédiger une preuve correcte suivant le schéma d'induction sur les formules
- ☐ Le cours doit être relu et compris

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.

Prouver $H \models C$

- 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
  - Agent artificiel intelligent
  - ...
- Parfois pas de prouveur disponible (problème trop gros, logique indécidable, ...)

Validité et Prouvabilité

Validité : $\{H_1, \ldots, H_n\} \models C$

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

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

- 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 $\{H_1, \ldots, H_n\} \vdash C$

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 .

Arbre de dérivation :

$$\dfrac{\dfrac{\dfrac{\overset{3}{n \vee v} \qquad \dfrac{\overset{7\ 1}{n} \qquad \overset{1}{n \longrightarrow f}}{f} \qquad \dfrac{\overset{7\ 2}{v} \qquad \overset{2}{v \longrightarrow f}}{f}}{f} \qquad \dfrac{\overset{6}{e} \qquad \overset{4}{e \longrightarrow \neg f}}{\neg f}}{\bot}}{\neg e}$$

Déduction naturelle : exemple (2)

Arbre de dérivation :

$$
\dfrac{
  \dfrac{
    \overset{3}{n \vee v}
    \qquad
    \dfrac{\overset{7.1}{n} \qquad \overset{1}{n \longrightarrow f}}{f}
    \qquad
    \dfrac{\overset{7.2}{v} \qquad \overset{2}{v \longrightarrow f}}{f}
  }{f}
  \qquad
  \dfrac{\overset{6}{e} \qquad \overset{4}{e \longrightarrow \neg f}}{\neg f}
}{
  \dfrac{\bot}{\neg e}
}
$$

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

Déduction naturelle : exemple (2)

Arbre de dérivation :

$$\cfrac{\cfrac{\overset{3}{n \vee v} \qquad \cfrac{\overset{7.\,1}{n} \qquad \overset{1}{n \longrightarrow f}}{f} \qquad \cfrac{\overset{7.\,2}{v} \qquad \overset{2}{v \longrightarrow f}}{f}}{f} \qquad \cfrac{\overset{6}{e} \qquad \overset{4}{e \longrightarrow \neg f}}{\neg f}}{\cfrac{\perp}{\neg e}}$$

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

Ce fait est exprimé par le **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, nous aurons toujours 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 $A \in \Gamma$. Parfois, on supprime même l'entière application de la règle (Ax) dans la présentation.

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 $\heartsuit$ (vous pouvez emprunter une carte, mais il faudra la rendre !)

Preuve de $\heartsuit$

Les règles :

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

Ax

La preuve :

$$\dfrac{\dfrac{\dfrac{}{\diamondsuit \vdash \diamondsuit} \qquad \dfrac{}{\quad}}{\quad}}{\quad}$$

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 :

$$\frac{\blacklozenge \vdash \blacklozenge}{\blacklozenge \vdash \clubsuit}\ (\alpha)$$

Ax

On applique α.

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 :

$$\cfrac{\cfrac{\diamondsuit \vdash \diamondsuit}{\diamondsuit \vdash \clubsuit}\ (\alpha) \quad \cfrac{\diamondsuit \vdash \diamondsuit}{\diamondsuit \vdash \spadesuit}\ (\beta)}{\cfrac{}{}}$$

Ax

On applique α.

De la même façon avec β.

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 :

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

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 :

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

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 !) :

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

La dérivation :

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

Une dérivation avec hypothèses

Autre système de règles (sans β !) :

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

Par axiome.

La dérivation :

$$\frac{}{\spadesuit, \blacklozenge \vdash \blacklozenge}$$

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

Une dérivation avec hypothèses

Autre système de règles (sans β !) :

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

Par axiome.

On applique α.

La dérivation :

$$\cfrac{\cfrac{\spadesuit, \blacklozenge \vdash \blacklozenge}{\spadesuit, \blacklozenge \vdash \clubsuit}\ (\alpha)}{\cfrac{}{}}$$

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

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 :

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

Par axiome.

On applique α.

Par axiome.

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

Une dérivation avec hypothèses

Autre système de règles (sans β !) :

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

La dérivation :

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

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 :

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

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

Par axiome.

On applique α.

Par axiome.

On applique γ.

On applique δ. Notez que l'hypothèse $\blacklozenge$ a disparu. Nous avons une dérivation de $\heartsuit$ à partir de $\spadesuit$.

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 jugement $\{H_1, \dots, H_n\} \vdash C$

Souvent, l'ensemble d'hypothèses est abbrévié 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 des formules *FORM* de la logique minimale L_{min} est le plus petit ensemble qui satisfait les conditions suivantes :

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

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 v à toute formule de L_{min} est définie récursivement de la façon suivante :
  - pour tout $p \in PROP$, $v(p) = v(p)$
  - pour tout $A, B \in FORM$, $v(A \longrightarrow B) = max(1 - v(A), v(B))$
- 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) $\Gamma \vdash r$:

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

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 $\Gamma \vdash p \longrightarrow r$:

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

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 l'antécédent principal.

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

Induction

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

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

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

  est un arbre de dérivation, alors

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

  est un arbre de dérivation.

- ... (règle ($E \longrightarrow$) : exercice)

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 $\square$

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 $(E \longrightarrow)$:

$$\dfrac{\begin{array}{c}\vdots \\ \Gamma \vdash A \longrightarrow B\end{array} \qquad \begin{array}{c}\vdots \\ \Gamma \vdash A\end{array}}{\Gamma \vdash B}\ (E \longrightarrow)$$

  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 $(I \longrightarrow)$:

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

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 $\Gamma \models A \longrightarrow B$. $\square$

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 d'incomplétude

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 $\perp$ 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 : $\vdash \neg(A \wedge B) \longrightarrow \neg(B \wedge A)$

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 : $\vdash \neg(A \longrightarrow B) \longrightarrow \neg A \longrightarrow C$

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 d'incomplétude

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

Preuve : (partielle) considérez la formule :

$$\neg(A \wedge B) \rightarrow \neg A \vee \neg B$$

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

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 $\Gamma \models C$.

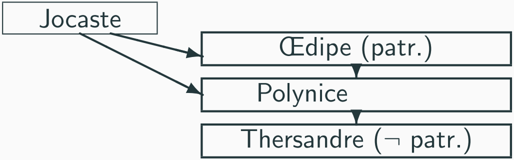
Théorème de complétude

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

Exemple du raisonnement classique

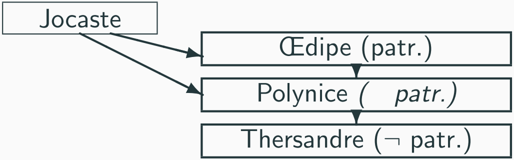
- Jocaste est la mère d'Œdipe.
- Jocaste et Œdipe sont les parents de Polynice.
- Polynice est le père de Thersandre.
- Œdipe est un patricide
- Thersandre n'est pas un patricide.

Exemple du raisonnement classique (2)



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

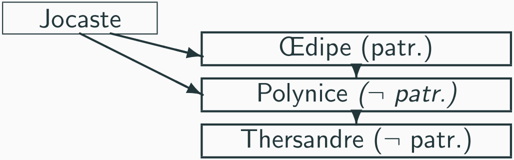
Exemple du raisonnement classique (2)



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

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

Exemple du raisonnement classique (2)



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

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

Preuve de $\neg(A \wedge B) \to \neg A \vee \neg B$

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(TND)}
\UnaryInfC{$\Gamma_1 \vdash A \vee \neg A$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma_2 \vdash \neg(A \wedge B)$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma_2 \vdash A$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma_2 \vdash B$}
\RightLabel{$(I\wedge)$}
\BinaryInfC{$\Gamma_2 \vdash A \wedge B$}
\RightLabel{$(E\neg)$}
\BinaryInfC{$\Gamma_1, A, B \vdash \bot$}
\RightLabel{$(I\neg)$}
\UnaryInfC{$\Gamma_1, A \vdash \neg B$}
\RightLabel{$(I\vee_2)$}
\UnaryInfC{$\Gamma_1, A \vdash \neg A \vee \neg B$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma_1, \neg A \vdash \neg A$}
\RightLabel{$(I\vee_1)$}
\UnaryInfC{$\Gamma_1, \neg A \vdash \neg A \vee \neg B$}
\RightLabel{$(E\vee)$}
\TrinaryInfC{$\Gamma_1 \vdash \neg A \vee \neg B$}
\RightLabel{$(I \longrightarrow)$}
\UnaryInfC{$\vdash \neg(A \wedge B) \to \neg A \vee \neg B$}
\end{prooftree}
$$

avec :

- $\Gamma_1 = \neg(A \wedge B)$
- $\Gamma_2 = \neg(A \wedge B), A, B$

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 :

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(TND)}
\UnaryInfC{$\Gamma, P \vdash Q \vee \neg Q$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma, P, Q \vdash Q$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma' \vdash P$}
\AxiomC{$\color{red}{\Gamma' \vdash \neg Q \rightarrow \neg P}$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma' \vdash \neg Q$}
\RightLabel{$(E \rightarrow)$}
\BinaryInfC{$\Gamma' \vdash \neg P$}
\RightLabel{$(E\neg)$}
\BinaryInfC{$\Gamma' \vdash \bot$}
\RightLabel{$(E\bot)$}
\UnaryInfC{$\Gamma, P, \neg Q \vdash Q$}
\RightLabel{$(E\vee)$}
\TrinaryInfC{$\Gamma, P \vdash Q$}
\RightLabel{$(I \rightarrow)$}
\UnaryInfC{$\color{red}{\Gamma \vdash P \rightarrow Q}$}
\end{prooftree}
$$

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

Exercice

Sur la diapo 13, 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 L_{pred} - Termes et formules

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

Rappel : Syntaxe de L_{pred} - 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 L_{pred} - 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 L_{pred} - Formules

Définition (formules)

Soit $PRED_n$ un ensemble des prédicats n-aires, l'ensemble $FORM$ 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 $P(t_1, \ldots, t_n) \in FORM$
2. **Constante "faux" :** $\bot \in FORM$
3. **Négation :** Si $A \in FORM$, alors $(\neg A) \in FORM$
4. **Connecteurs binaires :** Si $A \in FORM$ et $B \in FORM$, alors $(A \wedge B) \in FORM, (A \vee B) \in FORM, (A \longrightarrow B) \in FORM$
5. **Quantificateur universel** ("pour tout") : Si $A \in FORM$ et $x \in VAR$, alors $(\forall x.A) \in FORM$
6. **Quantificateur existentiel** ("il existe") : Si $A \in FORM$ et $x \in VAR$, alors $(\exists x.A) \in FORM$

Remarque : $FORM$ dépend de $TERM$, mais pas inversement.

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

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

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 :
  $$\forall I \in Etu.\, YeuxBleus(I)$$
- Dans les UEs Logique 1 et 2, on écrit plutôt :
  $$\forall I.\, Etudiant(I) \longrightarrow YeuxBleus(I)$$
- 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 :
  $$(Q(x) \lor \exists x.\, \forall y.\, P(f(x), z) \land Q(y)) \lor \forall x.\, R(x, z, g(x))$$
  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 :** $\perp[s/x] = \perp$
  - **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}

Exemple d'une preuve intuitive de :

On montre : $\forall x.\exists y.x < y$

1. Soit x une valeur arbitraire (mais fixe), pour laquelle il faut prouver $\exists y.x < y$.
2. Pour montrer $\exists y.x < y$, il faut trouver un y qui satisfait $x < y$ (et qui peut dépendre du x). On choisit $x + 1$ pour y.
3. On vérifie que $x < x + 1$ est satisfait.

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 $(I\forall)$:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall x.P(x) \lor Q(x)}\ (Ax)}{\Gamma \vdash P(x) \lor Q(x)}\ (E\forall) \qquad \dfrac{\dfrac{\dfrac{}{\Gamma_P \vdash P(x)}\ (Ax)}{\Gamma_P \vdash \forall x.P(x)}\ \color{red}{(I\forall)}}{\Gamma_P \vdash (\forall x.P(x)) \lor \forall x.Q(x)}\ (I\lor_1) \qquad \dfrac{\dots}{\Gamma_Q \vdash \dots}\ (I\lor_2)}{\Gamma \vdash (\forall x.P(x)) \lor \forall x.Q(x)}\ (E\lor)$$

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

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 :

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

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 logiques 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 (2)

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 (2)

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 (2)

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 (2)

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 (3)

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 (3)

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 : $\sigma = [x_1 \leftarrow t_1;\ ...\ ; x_n \leftarrow t_n]$ 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 $\sigma = [x \leftarrow h(y); y \leftarrow 42]$

$$\sigma\big(f(x, g(y))\big)$$
$$= f\big(\sigma(x), \sigma(g(y))\big)$$
$$= f\big(h(y), g(\sigma(y))\big)$$
$$= f\big(h(y), g(42)\big)$$

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 :** $(\{f(s_1, \dots, s_n) \stackrel{?}{=} f(t_1, \dots, t_n)\} \cup E, S) \Longrightarrow$
  $(\{s_1 \stackrel{?}{=} t_1, \dots, s_n \stackrel{?}{=} t_n\} \cup E, S)$
- **Clash :** $(\{f(s_1, \dots, s_n) \stackrel{?}{=} g(t_1, \dots, t_m)\} \cup E, S) \Longrightarrow$ *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 :** $(\{f(s_1, \ldots, s_n) \stackrel{?}{=} f(t_1, \ldots, t_n)\} \cup E, S) \Longrightarrow$
  $(\{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\} \cup E, S)$
- **Clash :** $(\{f(s_1, \ldots, s_n) \stackrel{?}{=} g(t_1, \ldots, t_m)\} \cup E, S) \Longrightarrow$ *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 :

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

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. $\square$

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 $(\{x \stackrel{?}{=} x\}, S)$ $\square$

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 $(E, S) \Longrightarrow \textit{fail}$, 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 :

  La formule $F \in L_{prop}$ 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
  - $\{H_1, \cdots, H_n\} \models C \;\Leftrightarrow\; (H_1 \wedge \cdots H_n \wedge \neg C)$ insatisfiable

Principe de résolution pour L_{pred}

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

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

Intuition sur la procédure : exemple 1

- **Problème** : on veut montrer $\forall x.P(x) \longrightarrow Q(x) \models (\forall x.P(x)) \longrightarrow \forall x.Q(x)$ c'est-à-dire, montrer : $(\forall x.P(x) \longrightarrow Q(x)) \land (\forall x.P(x)) \land \exists x.\neg Q(x)$ insatisfiable
- **Solution** :
  - on introduit une constante fraîche a ;
  - *clairement*, $(\forall x.P(x) \longrightarrow Q(x)) \land (\forall x.P(x)) \land \exists x.\neg Q(x)$ est insatisfiable si et seulement si la formule $(\forall x.P(x) \longrightarrow Q(x)) \land (\forall x.P(x)) \land \neg Q(a)$ 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. $\perp$ .... 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. $\perp$ .... contradiction entre 3 et 5

Intuition sur la procédure : les grandes étapes

F insatisfiable ?

Passage en forme universelle close

- Mise en forme prénexe
- Skolémisation

Passage en forme clausale

Recherche d'une dérivation menant à la clause vide

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

F insatisfiable ?

Passage en forme universelle close

- Mise en forme prénexe
- Skolémisation

Passage en forme clausale

Recherche d'une dérivation menant à la clause vide

Mise en forme prénexe : définition

Définition (forme prénexe)

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

$$\eth_1 x_1.\eth_2 x_2.\cdots \eth_n x_n.\phi$$

où $\eth_1, \cdots, \eth_n \in \{\exists, \forall\}$ 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 $\circlearrowleft x.\phi$ telle que x apparaît en dehors de ϕ, on remplace $\circlearrowleft x.\phi$ par $\circlearrowleft y.\phi[x \leftarrow y]$ où y est une variable fraîche
4. **Faire passer les quantificateurs à gauche** en appliquant autant que possible les règles suivantes :
   4.1 $(\circlearrowleft x.A) \wedge B$ devient $\circlearrowleft x.(A \wedge B)$
   4.2 $(\circlearrowleft x.A) \vee B$ devient $\circlearrowleft x.(A \vee B)$
   4.3 $A \wedge (\circlearrowleft x.B)$ devient $\circlearrowleft x.(A \wedge B)$
   4.4 $A \vee (\circlearrowleft x.B)$ devient $\circlearrowleft x.(A \vee B)$

   ... où $\circlearrowleft \in \{\exists, \forall\}$

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. $\neg(\forall x.P(x)) \lor (\exists x.R(x))$ (élimination de $\longrightarrow$)
2. $(\exists x.\neg P(x)) \lor (\exists x.R(x))$ (négation propagée vers l'intérieur)
3. $(\exists x.\neg P(x)) \lor (\exists y.R(y))$ (distinction des variables)
4. $\exists x.(\neg P(x) \lor (\exists y.R(y)))$ (application de la règle 4.2)
   $\exists x.\exists y.(\neg P(x) \lor R(y))$ (application de la règle 4.4)

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

Les grandes étapes

F insatisfiable ?

Passage en forme universelle close

- Mise en forme prénexe
- Skolémisation

Passage en forme clausale

Recherche d'une dérivation menant à la clause vide

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 $\exists y$

Définition (skolémisation partielle)

Soit F une formule en forme prénexe de la forme :

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

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

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

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)

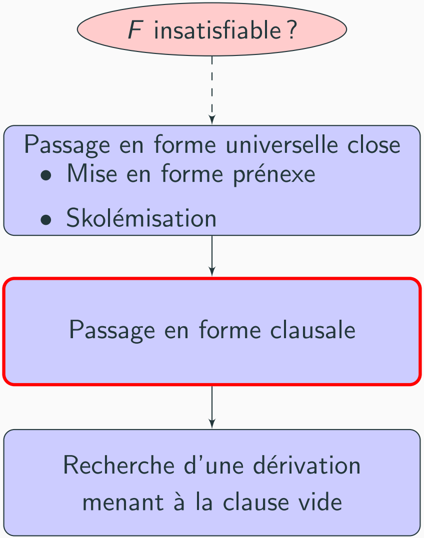
La forme universelle close d'une formule de la logique des prédicats est obtenue en lui appliquant successivement une mise en forme **prénexe** puis une **skolémisation** et est ainsi de la forme :

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

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 $\perp$.
- 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 $p \leftrightarrow q$ en $(p \longrightarrow q) \wedge (q \longrightarrow p)$
   1.2 Réécrire $p \longrightarrow q$ en $\neg p \vee q$
2. Tirer les négations à l'intérieur, éliminer les doubles négations :
   - $\neg(p \wedge q)$ devient $\neg p \vee \neg q$
   - $\neg(p \vee q)$ devient $\neg p \wedge \neg q$
   - $\neg\neg p$ devient p
3. Distribuer $\vee$ sur $\wedge$:
   - $p \vee (q \wedge r)$ devient $(p \vee q) \wedge (p \vee r)$
   - et pareil pour $(q \wedge r) \vee p$
4. Éliminer $\perp$ et $\top$: $q \wedge (p \vee \perp)$, $p \vee \neg\perp \ldots$

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 $F \equiv F'$.

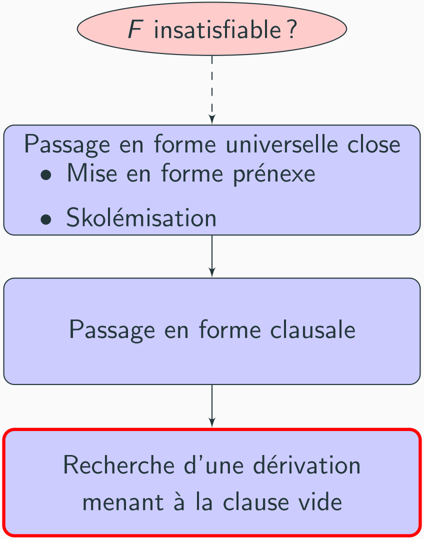
Passage en forme clausale

- 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

Les grandes étapes



Résolution pour L_{pred}

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

Insatisfiabilité de $F \in L_{prop}$

$\Leftrightarrow$

$\{\ \}$ obtenu par calcul itératif de résolvantes à partir des clauses de F

- Cas L_{pred} (*à venir*) :

Insatisfiabilité de $F \in L_{pred}$

$\Leftrightarrow$

Dérivation de $\{\ \}$ à partir des clauses de F, dans un calcul de résolvantes modulo unification

Résolution pour L_{pred} : règles

Axiome :

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

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
- $\sigma(C)$ revient à appliquer σ à chaque littéral de la clause C

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 :

$\Delta = \{\{t(a)\}, \{\neg d(x_2), R(x_1, x_2)\}, \{\neg t(x_3), \neg Q(x_4), \neg R(x_3, x_4)\}, \{d(b)\}, \{Q(b)\}$

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

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

Résolution pour L_{pred} : théorèmes

Théorème de correction

La résolution est correcte : si $\Delta \triangleright \{ \}$ 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 $\Delta \triangleright \{ \}$.

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