Licence STS, semestre 4 — 2012–13
Mathématiques pour l'Informatique (Info 229) — 28 janvier 2013
`http://www.lri.fr/~paulin/MathInfo`

TP 1 - Eléments de logique

Exercice 1 (WIMS)

- Se connecter à l'espace numérique de travail.
- Recherche dans l'espace WIMS le cours `Info229`.
- Compléter la feuille d'exercices `TP1`.

Exercice 2 (Logique propositionnelle)

On va démontrer en Coq les énoncés 1–4 de l'Exercice 4 du TD1.

Ces énoncés vous sont donnés en syntaxe Coq. Il vous suffit de les écrire dans la fenêtre de script de CoqIDE et de les charger (bouton ⇓) pour commencer la preuve.

Astuce : on commence (très) souvent une preuve par la tactique `intros` qui permet d'introduire successivement les connecteurs et quantificateurs (universels) principaux de la conclusion du but courant.

A la fin d'une preuve, lorsque tous les buts ont été prouvés, il est nécessaire de demander à Coq de vérifier l'ensemble de la preuve, à l'aide de la commande `Qed`.

1. `Theorem un : forall P Q R : Prop, (P /\ Q) -> R -> (Q /\ R).`
   Quelle est la tactique Coq correspondant à la règle d'introduction de la conjonction ?
2. `Theorem uncurry : forall P Q R : Prop, (P -> Q -> R) -> (P /\ Q -> R).`
3. `Theorem contra  : forall P Q : Prop, (P -> Q) -> ((~ Q) -> (~ P)).`
   Comme vous l'avez vu en TD, pour montrer facilement le principe de contraposée, il faut bien avoir en tête la façon dont on peut manipuler la négation (autrement dit ses règles d'introduction et d'élimination). Attention, la tactique `intros` n'introduit pas la négation, il faut utiliser `intro` à la suite.
4. `Theorem quatre   : forall P Q R : Prop, (P \/ (Q /\ R)) -> P \/ Q.`

Exercice 3 (Quantificateurs)

Nous nous intéressons, dans cet exercice, à la façon de prouver des formules qui contiennent des quantificateurs. Commençons par délimiter le code correspondant à cet exercice en introduisant la commande `Section Exo2.` (il faudra refermer cette section à la fin du code, grâce à la commande `End Exo2.` Faîtes de même pour les autres exercices de ce tp).

1. Nous allons travailler avec un prédicat quelconque sur les entiers naturels. On peut le définir de la manière suivante : `Variable p : nat -> Prop.`
   Prouvez le théorème suivant :

   `Theorem fimpe : (forall x:nat, p x) -> exists y:nat, p y.`

   L'implication inverse est-elle démontrable ?
2. Considérons maintenant un prédicat à deux arguments (autrement dit une *relation*) :
   `Variable rel : nat -> nat -> Prop.`
   Prouvez le théorème suivant :

   `Theorem effe : (exists x, forall y, rel x y) -> (forall y, exists x, rel x y).`

   (Notez que l'on n'a pas précisé les types des variables dans les quantifications de cet exemple, car Coq peut les deviner (les *inférer*) tout seul.)
   L'implication inverse est-elle démontrable ?

3. Nous allons maintenant nous intéresser à la relation "être plus petit que". Cette relation peut se définir en Coq sous la forme d'une relation définie par une formule existentielle :

```
Definition le (x y:nat) : Prop := exists z:nat, y = x + z.
```

   a. Prouvez que l'addition est croissante pour la relation `le` ("less or equal") :

```
Theorem plus_croissante : forall x y:nat, le x (x+y).
```

   Quelle tactique doit-on utiliser pour déplier une définition ?

   b. On dispose en Coq d'une tactique permettant de prouver des buts simples d'arithmétique linéaire (sans multiplication). Cette tactique s'appelle `omega`. Pour pouvoir l'utiliser, vous devez charger le *module* qui la contient :

```
Require Import Omega.
```

   Prouvez que la relation `le` est réflexive :

```
Theorem le_refl : forall x:nat, le x x.
```

   c. Prouvez que la relation `le` est transitive :

```
Theorem le_trans : forall (x y z:nat), le x y /\ le y z -> le x z.
```

   A quel moment est-il judicieux de déplier la définition de `le` ?

Exercice 4 (Vérités et mensonges)

Sur l'île des purs et des pires, il y a des habitants.

```
Variable habitant : Set.
```

Ces habitants sont soit des purs, soit des pires.

```
Variable pur : habitant -> Prop.
Variable pire : habitant -> Prop.
Axiom pur_ou_pire : forall x, pur x \/ pire x.
```

Les purs disent toujours la vérité alors que les pires mentent toujours.

```
Variable say : habitant -> Prop -> Prop.
Axiom ax_pur : forall x P, pur x -> say x P -> P.
Axiom ax_pire : forall x P, pire x -> say x P -> ~P.
```

1. Montrez que si `a` et `b` sont deux habitants de l'île et que `a` dit que `b` est pire, alors au moins l'un des deux est pire. Astuce : introduisez le résultat intermédiaire du fait que `a` est soit pur soit pire, en utilisant la tactique `assert`, puis faîtes un raisonnement par cas.
2. Montrez que si `a` dit que `a` et `b` sont tous les deux pires, alors `a` est pire.
3. Montrez que si `a` dit `Faux` alors c'est un pire.
4. Que peut-on dire si un habitant dit qu'il est lui-même un pire ?