TP 5 - L'opérateur logique « non » et les preuves par l'absurde

Remarque. Pour répondre au questions de ce TD, vous aurez besoin d'importer le module `Mathlib.Logic.Basic`, qui contient la plupart des résultats liés aux démonstrations par l'absurde et au principe du tiers-exclu.

Pour cela, tapez

```
import Mathlib.Logic.Basic
```

au début de votre fichier.

1 La proposition « `False` »

Comme vous le savez, si P est une proposition fausse, alors l'implication $P \Rightarrow Q$ est toujours vraie, quelle que soit la proposition Q.

En LEAN*, si l'une des hypothèses du contexte est trivialement fausse, il est possible de terminer instantanément la preuve de la proposition en tapant « `contradiction` ».*

1. Commencez à taper la preuve d'une proposition `faux_implique_tout`, affirmant la chose suivante :

$$(-1 = 0) \Rightarrow (2 + 2 = 5).$$

Dans la preuve, après avoir introduit l'hypothèse (mot-clé « `intro` »), terminez la démonstration en écrivant « `contradiction` ».

En LEAN*, il existe une proposition « `False` », qui est définie comme étant fausse. Elle verra qu'elle peut servir à faire des démonstrations par l'absurde. Pour le moment, vérifions dans la question suivante que cette proposition implique toute autre proposition.*

2. Tapez le code suivant :

```
theorem faux_implique_tout2 :
    False → 2 + 2 = 5 := by
{
    --Arguments de la preuve
}
```

Après avoir introduit l'hypothèse, terminez la preuve en utilisant le mot-clé « `contradiction` ». Faites des essais en remplaçant la conclusion par n'importe quelle assertion de votre choix.

2 L'opérateur logique « non »

Si P est une proposition, LEAN *interprète la proposition « non(P) » comme synonyme de l'implication « $P \Rightarrow$ Faux », autrement dit, « P engendre une contradiction ». On peut employer cela dans une preuve, comme indiqué à la question suivante.*

3. Tapez le code suivant (pour le symbole « ¬ », tapez « `\not` ») :

```
theorem exemple_negation :
    ¬ ( 2 + 2 = 5 ) := by
{
    intro H
    contradiction
}
```

Voyons un autre exemple un peu plus élaboré.

4. Tapez le code suivant.

```
theorem exemple_negation_2 :
    ¬ ( ∃ x : Nat, ∀  y : Nat, x ≥ y) := by
{
    intro H
    -- Suite de la preuve
}
```

Essayez de terminer la preuve de l'assertion précédente *(Indication : commencez par utiliser* `rcases` *sur l'hypothèse* `H`*)* : l'objectif est d'arriver à une contradiction.

Habituellement, quand on veut démontrer la preuve d'une affirmation écrite sous la forme d'une formule du premier ordre, on ne raisonne pas forcément par contradiction. On peut aussi utiliser les règles de réécriture classiques :

$$\neg(\exists x \in E, P(x)) \Leftrightarrow (\forall x \in E, \neg P(x))$$

$$\neg(\forall x \in E, P(x)) \Leftrightarrow (\exists x \in E, \neg P(x))$$

5. Avant de répondre à la question suivante, réécrivez l'assertion

$$\neg(\exists x \in \mathbb{N}, \forall y \in \mathbb{N}, x \geq y)$$

en utilisant ces règles.

6. On peut utiliser le mot-clé `push_neg` pour invoquer ces règles de réécriture. Essayez ainsi le code suivant :

```
theorem exemple_negation_3 :
    ¬ ( ∃ x : Nat, ∀  y : Nat, x ≥ y) := by
{
    push_neg
    -- Suite de la preuve
}
```

Que se passe-t-il ? Terminez la preuve.

Vous pouvez maintenant regarder l'aide-mémoire, qui contient deux nouveaux encadrés mentionnant les techniques dont on vient de discuter.

7. *(Rappel)* Définissez un prédicat `est_injective` portant sur une fonction $f : \mathbb{Z} \to \mathbb{Z}$ et affirmant que f est injective. De même, définissez un prédicat portant sur une fonction $f : \mathbb{Z} \to \mathbb{Z}$ et affirmant que f est surjective.

8. Montrez maintenant en utilisant LEAN que la fonction $f : x \in \mathbb{Z} \mapsto x^2 \in \mathbb{Z}$ n'est pas injective. Ecrivez en parallèle la preuve au papier.

(après avoir défini la fonction « `f : Int → Int` » adéquate, commencez par démontrer la proposition « `¬(est_injective f)` »)

3 Quelques compléments

Avant de passer aux dernières questions de ce TP, qui portent sur le raisonnement par contraposée, on aura besoin de quelques prérequis pour faire des calculs d'inégalités en LEAN.

3.1 La tactique `linarith`

9. Le mot-clé `linarith` demande à LEAN de démontrer un objectif sous la forme d'une inégalité, en appliquant des règles de calcul élémentaires au contexte. Attention, cette tactique ne fonctionne pas toujours. Pour voir un exemple, tapez le code suivant :

```
theorem exemple_inegalite :
    ∀ x y : Int, x < y   →   3 * x < 3 * y := by
{
    intro x y
    intro H
    -- Suite de la preuve
}
```

Terminez la preuve en remplaçant les commentaires par le mot clé `linarith`.

10. Définissez un prédicat `est_strictement_croissante` portant sur une fonction `f : Int→Int`, et affirmant que la fonction `f` est strictement croissante.

11. En vous servant du prédicat précédent, montrer en LEAN que la fonction $h : x \in \mathbb{Z} \mapsto 3x + 1 \in \mathbb{Z}$ est strictement croissante.

3.2 Un énoncé admis temporairement : utilisation du mot-clé `sorry`

Dans la section suivante, on va avoir besoin de l'énoncé suivant :

$$\forall x, y \in \mathbb{Z},\ x \neq y \Rightarrow (x < y \ \vee \ x > y) \qquad (P).$$

Il nous manque encore quelques éléments pour le démontrer correctement en LEAN, mais on peut demander à la machine de l'admettre temporairement. Pour cela, on peut utiliser le mot-clé `sorry`.

12. Définissez un théorème `diff_implique_ineg` dont l'énoncé est la propriété (P) ci-dessus. Dans la preuve, tapez simplement `sorry`. Que se passe-t-il ?

A partir de maintenant, vous pouvez faire appel dans vos démonstrations au théorème `diff_implique_neg`, *exactement comme s'il faisait partie du contexte.*

4 Raisonnement par contraposée

Lorsqu'on doit démontrer une implication de la forme `P→Q`*, on peut demander à LEAN de raisonner par contraposée, en tapant le mot-clé* `contrapose`*, qui remplace alors l'objectif par* `¬(Q)→(¬P)`*.*

13. En utilisant les prédicats `est_injective` et `est_strictement_croissante`, montrez qu'une application strictement croissante `f : Int→Int` est injective. *(Procédez en commençant par raisonner par contraposée).*

5 Le signe ≠

En LEAN, une expression de la forme « `expr1 ¬ expr2` *» est synonyme de «* `¬(expr1=expr2)` *». Il est donc possible de démontrer une telle inégalité en raisonnant par l'absurde.*

14. Montrer que la fonction g de la question **8** n'est pas surjective. Pour cela, vous pourrez utiliser le théorème `sq_nonneg` de la librairie *Mathlib*, et finalement utiliser la tactique `linarith` pour obtenir une contradiction (nous verrons plus de détails sur ces énoncés à la séance prochaine).