TP 6 - Faire des calculs avec LEAN. Propriétés de base des anneaux

1 La bibliothèque Mathlib

La bibliothèque **Mathlib** *est un effort collaboratif de nombreux mathématiciens et mathématiciennes, dont l'objectif à long terme serait d'écrire en LEAN l'ensemble des résultats mathématiques connus à ce jour. Nous en sommes encore très loin.*

Vous pouvez avoir un aperçu de ce travail ici :

https://leanprover-community.github.io/mathlib-overview.html

Vous pouvez cliquer sur l'un ou l'autre des liens de la page pour voir divers théorèmes prouvés par la communauté. Moyennant quelques détails techniques, vous pourrez constater qu'ils s'écrivent avec une syntaxe proche de celle que nous avons vues dans les séances passées.
En fonction de la partie sur laquelle vous cliquez, il peut s'agir soit de mathématiques de niveau primaire (par exemple : https://leanprover-community.github.io/mathlib4_docs/Mathlib/Data/Nat/Cast/Defs.html#one_add_one_eq_two), soit d'énoncés à la pointe de la recherche !

Aujourd'hui, nous allons utiliser quelques-uns de ces théorèmes pour faire des calculs sur les nombres entiers en LEAN.

2 Quelques énoncés de base

1. Tapez le code suivant :

```
#check (left_distrib)
```

Vous devriez voir s'afficher quelque chose ressemblant à cela :

```
left_distrib :    ∀ (a b c : ?m.2),   a * ( b + c) = a * b + a * c
```

Il s'agit du théorème affirmant que la multiplication est distributive à gauche sur l'addition. Comme cela est vrai pour divers ensembles de nombres (par exemples $\mathbb{N}$*,* $\mathbb{R}$*,* $\mathbb{C}$*, etc), LEAN affiche le symbole* `?m.2` *pour indiquer que le théorème peut s'appliquer si on remplace ce symbole par divers type comme* `Nat`*,* `Int`*,* `Real`*, etc. La documentation de LEAN est censée indiquer pour quels types on peut appliquer ce théorème.*

Voyons un exemple d'utilisation.

2. Tapez le code suivant :

```
theorem exemple_1 :
    4 * ( 2 + 3) = 4 * 2 + 4 * 3 := by
{
    --Arguments de la preuve
}
```

Terminez la preuve en remplaçant les commentaires par :

```
    rw [left_distrib]
```

Il est aussi possible d'indiquer à quelle partie de l'égalité on veut appliquer le théorème, comme on va le voir dans la question suivante :

3. Tapez le code suivant :

```
theorem exemple_2 :
    ∀  a b c d e f : Int,
a * (b + c) + d * (e + f) = a * (b + c) + (d * e + d * f) := by
{
    intro a b c d e f
    --Arguments de la preuve
}
```

Terminez la preuve en remplaçant les commentaires par :

```
    rw [left_distrib d e f]
```

Que se passe-t-il si on écrit simplement « `rw [left_distrib]` » ?

Voyons maintenant les autres théorèmes qui nous seront utiles dans la suite.

3. Tapez les commandes suivantes. Pour chaque ligne, retrouvez le nom de la propriété mathématique dont parle l'énoncé.

```
#check (right_distrib)
#check (mul_comm)
#check (add_comm)
#check (add_assoc)
#check (mul_assoc)
```

Voici une petite indication pour comprendre les deux derniers énoncés ci-dessus : en LEAN, il y a des parenthèses implicites lorsqu'on a deux additions à la suite : plus précisément, une expression de la forme

```
a + b + c
```

est considérée comme synonyme de

```
(a + b) + c
```

On a la même chose avec la multiplication.

4. Utilisez maintenant les propriétés vues ci-dessus et la commande `rw` pour démontrer la propriété suivante :

$$\forall a, b, c, d \in \mathbb{Z}, (a+b)(c+d) = ac + ad + bc + bd.$$

Faites la preuve deux fois : une fois en commençant par utiliser la distributivité à droite, et une autre fois en utisant la distributivité à gauche. Quelle est la version la plus rapide ?
(Indication : commencez par écrire l'énoncé en LEAN. Après avoir introduit les variables, vous pouvez commencer à distribuer l'expression. A la fin, il faudra utiliser l'associativité de l'addition. Sur une feuille, notez les parenthèses implicites à chaque étape pour vous aider).

Pour la question suivante, on va avoir besoin des théorèmes `pow_two` *et* `two_mul`*. Commencez par utiliser la commande* `#check` *pour voir ce que disent ces énoncés.*

5. Démontrez l'identité remarquable

$$\forall a, b \in \mathbb{Z}, \quad (a+b)^2 = a^2 + 2ab + b^2.$$

6. Optionnel*. Cherchez les énoncés pertinents dans la bibliothèque Mathlib pour démontrer les identités remarquables

$$\forall a, b \in \mathbb{Z}, \quad a^2 - b^2 = (a+b)(a-b),$$

et

$$\forall a, b \in \mathbb{Z}, \quad (a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$
$$\forall a, b \in \mathbb{Z}, \quad a^3 - b^3 = (a+b)(a^2 + ab + b^2).$$

En fait, LEAN dispose de tactiques de simplification qui permettent de trivialiser la plupart du code vu ci-dessus. Pour simplifier des expressions n'utilisant que les symboles `+` *et* `*`*, il suffit dans beaucoup de situations d'appliquer la tactique «* `ring` *» qui essaie de simplifier l'expression en utilisant les propriétés des lois de composition dans un anneau. (« ring » signifie « anneau » en anglais.*

7. Reprenons la question **4**. Pour cela, tapez le code suivant :

```
theorem distributivite_double
   ∀ a b c d : Int, (a + b)*(c + d) = a*c + a*d + b*c + b*d := by
{
   intro a b c d
   --Fin de la preuve
}
```

Remplacez maintenant les commentaires par la commande `ring`. Que se passe-t-il ? Vérifiez que vous pouvez aussi démontrer instantanément un ou deux des énoncés de la question **6**.

3 La tactique `calc`

Il est parfois utile d'organiser son code de façon à ce que l'on voie bien quelles sont les différentes étapes d'un calcul. Si l'objectif est de montrer une égalité

$$A = B,$$

on peut utiliser le mot-clé `calc` *qui permet d'organiser son code de la façon suivante :*

```
calc
A  = (Etape_1) :=     by   {Justification_1}
 _ = (Etape_2) :=     by   {Justification_2}
 _ = (Etape_3) :=     by   {Justification_3}
   ...
 _ = B         :=     by   {Justification_finale}
```

Bien sûr, il faut remplacer `A`, `B`, *et chacune des étapes et justifications par ce qui est pertinent suivant le contexte.*

Reprenons la question **4** *pour voir cela sur un exemple. On peut démontrer la formule de la façon suivante.*

```
theorem distributivite_double_calc
   ∀ a b c d : Int, (a + b)*(c + d) = a*c + a*d + b*c + b*d := by
{
   intro a b c d
   calc
   (a+b)*(c+d) = a * (c + d) + b * (c + d)          :=  by { rw [right_distrib]}
             _ = (a * c + a * d) + (b * c + b * d) :=  by { rw [left_distrib, left_distrib]}
             _ = a * c + a * d + b * c + b * d     :=  by { rw [←add_assoc]}
}
```

Pour écrire la preuve au papier en parallèle, on peut suivre les mêmes calculs, en mettant entre paranthèse la justification. Cela donnerait la chose suivante :

Soient $a, b, c, d \in \mathbb{Z}$. Montrons que $(a+b)(c+d) = ac + ad + bc + bd$.

On calcule :

$$\begin{aligned}(a+b)(c+d) &= a(b+d) + b(c+d) && \text{(distributivité à droite)}\\ &= (ab+ad) + (bc+bd) && \text{(distributivité à gauche, deux fois)}\\ &= ab + ad + bc + bd && \text{(associativité de l'addition)}\end{aligned}$$

Ceci montre le résultat.

8. Tapez le code ci-dessus. N'oubliez pas les symboles « _ ». La console va afficher des messages d'erreur quand vous taperez le mot `calc` : c'est normal. Normalement, dans les accolades venant après chaque mot-clé « `by` », la console devrait afficher un objectif correspondant à l'égalité de la ligne en cours.

9. Utilisez la syntaxe `calc` pour démontrer l'identité remarquable de la question **5.**

4 Un peu d'applications

Dans les questions suivantes, n'utilisez pas la tactique `ring`*, ni la tactique* `simp`.

10. Définissez un prédicat `est_pair` portant sur un entier $n \in \mathbb{Z}$ et affirmant que n est pair.

11. Utilisez LEAN et le prédicat `est_pair` pour montrer que la somme de deux entiers pairs est paire. Ecrivez la preuve au papier en parallèle.

12. De même, montrez que le produit de n'importe quel entier avec un entier pair est pair.

13. Montrer maintenant que pour tous entiers $n, m \in \mathbb{Z}$, si nm n'est pas pair, alors n n'est pas pair et m n'est pas pair.

(Utiliser le théorème démontré à la question précédente, et rappelez-vous le TP précédent en ce qui concerne les négations).

14.(BONUS) Utilisez maintenant LEAN pour montrer que pour tous $a, b \in \mathbb{Z}$, l'entier $(a + b)^2 - a^2 - b^2$ est pair. Ecrivez la preuve au papier en parallèle.

(Cette question est plus dure : écrivez au moins la rédaction en utilisant la tactique `ring`*. Arriverez-vous à écrire une preuve sans utiliser cette tactique ? Vous aurez sans doute besoin de fouiller un peu dans la documentation en ligne.)*