

Preuves assistées par ordinateur – TD n° 9

## Définitions inductives de prédicats et analyse par cas

### Exercice 1 – L’énigme de MU (Hofstadter 1986)

Sur l’alphabet à trois lettres  $\{\mathbf{M}; \mathbf{I}; \mathbf{U}\}$ , on considère le langage  $\mathcal{L}$  défini à partir de l’axiome et des règles suivantes :

**Axiome.**  $\mathbf{MI} \in \mathcal{L}$  ;

**Règle 1.** Si  $x\mathbf{I} \in \mathcal{L}$ , alors  $x\mathbf{IU} \in \mathcal{L}$  (avec  $x \in \{\mathbf{M}; \mathbf{U}; \mathbf{I}\}^*$ ) ;

**Règle 2.** Si  $\mathbf{M}x \in \mathcal{L}$ , alors  $\mathbf{M}xx \in \mathcal{L}$  (avec  $x \in \{\mathbf{M}; \mathbf{U}; \mathbf{I}\}^*$ ) ;

**Règle 3.** Si  $x\mathbf{III}y \in \mathcal{L}$ , alors  $x\mathbf{U}y \in \mathcal{L}$  (avec  $x, y \in \{\mathbf{M}; \mathbf{U}; \mathbf{I}\}^*$ ) ;

**Règle 4.** Si  $x\mathbf{UU}y \in \mathcal{L}$ , alors  $xy \in \mathcal{L}$  (avec  $x, y \in \{\mathbf{M}; \mathbf{U}; \mathbf{I}\}^*$ ).

**Question :** Le mot  $\mathbf{MU}$  appartient-il au langage ? Pourquoi ?

\*  
\*\*

On se propose de formaliser ce problème en Coq. Pour cela, on introduit les types de données `alpha` (type des lettres) et `word` (type des listes de lettres) définis inductivement par :

```
Inductive alpha : Set := M : alpha | I : alpha | U : alpha.
```

```
Definition word : Set := list alpha.
```

1. Définir des constantes `word_M`, `word_MI`, `word_MU`, `word_I`, `word_IU`, `word_III`, `word_U`, `word_UU` qui représentent les *mots*  $\mathbf{M}$ ,  $\mathbf{MI}$ ,  $\mathbf{MU}$ ,  $\mathbf{I}$ ,  $\mathbf{IU}$ ,  $\mathbf{III}$ ,  $\mathbf{U}$  et  $\mathbf{UU}$ .

Le langage  $\mathcal{L}$  est modélisé en Coq sous la forme d’un prédicat `lang` : `word`  $\rightarrow$  `Prop` défini inductivement par

```
Inductive lang : word -> Prop :=
| axiom :
  lang word_MI
| rule1 : forall x,
  lang (x ++ word_I) -> lang (x ++ word_IU)
| rule2 : forall x,
  lang (word_M ++ x) -> lang (word_M ++ x ++ x)
| rule3 : forall x y,
  lang (x ++ word_III ++ y) -> lang (x ++ word_U ++ y)
| rule4 : forall x y,
  lang (x ++ word_UU ++ y) -> lang (x ++ y).
```

2. Montrer que tous les mots du langage `lang` commencent par la lettre  $\mathbf{M}$ .

On cherche maintenant à établir que tous les mots du langage ont un nombre d'occurrences de la lettre **I** qui n'est pas un multiple de trois. Pour cela, on formalise d'abord l'arithmétique modulo 3 en Coq à partir de la définition inductive suivante :

`Inductive Z3 : Set := Z0 : Z3 | Z1 : Z3 | Z2 : Z3.`

3. Définir les fonctions `succ : Z3 → Z3` et `plus : Z3 → Z3 → Z3` qui implémentent le successeur et l'addition modulo 3.
4. Montrer que `plus` est commutative, associative, et qu'elle admet `Z0` pour élément neutre.
5. Montrer que pour tout `z : Z3`, `z ≠ Z0 → plus z z ≠ Z0`.
6. Définir une fonction `occurI3 : word → Z3` qui à chaque mot `w : word` associe le nombre d'occurrences de la lettre **I** modulo 3 dans `w`.
7. Montrer que pour tous mots `v, w` on a :

$$\text{occurI3 } (v ++ w) = \text{plus } (\text{occurI3 } v) (\text{occurI3 } w).$$

8. Montrer que pour tout mot `w : word`, `lang w` entraîne que `occurI3 w ≠ Z0`.
9. En déduire que `¬ lang word_MU`.

**Tactiques utiles :** `generalize, elim, induction, case, discriminate, inversion.`