

## 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  $\mathcal{M}; \mathcal{I}; \mathcal{U}$ , on considère le langage  $\mathcal{L}$  défini à partir de l'axiome et des règles suivantes :

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

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

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

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

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

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

\*  
\*\*

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

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

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

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

Le langage  $\mathcal{L}$  est modélisé en Coq sous la forme d'un prédicat `lang : word -> 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  $\mathcal{M}$ .

On cherchera 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 la succession et l'addition modulo 3.
4. Montrer que `plus` est commutatif, associatif, et qu'il admet Z0 pour élément neutre.
5. Montrer que pour tout  $z : Z3$ ,  $z \neq Z0 \rightarrow plus\ z\ z \neq Z0$ .
6. Définir une fonction `occureI3 : 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 :

$$occureI3\ (v ++ w) = plus\ (occureI3\ v)\ (occureI3\ w).$$

8. Montrer que pour tout mot  $w : word$ , l'eng  $w$  entraîne que  $occureI3\ w \neq Z0$ .
9. En déduire que : l'eng `word_MU`.

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