

## TP 3 - Définitions par clôture

Un squelette de fichier Coq vous est fourni sur la page du cours.

### Exercice 1 (Parité)

En Coq, une définition par clôture se construit comme une définition **inductive**. Par exemple, la définition des entiers pairs par les deux règles suivantes :

$$\frac{}{\text{pair}(0)} \quad \frac{\text{pair}(x)}{\text{pair}(x + 2)}$$

s'écrit en Coq comme ceci :

```
Inductive pair : nat -> Prop :=
| pair_0 : pair 0
| pair_ss : forall x, pair x -> pair (S (S x)).
```

Chacune des règles est nommée, ce qui permet de les utiliser dans les raisonnements. Par exemple, si l'on doit prouver `pair 4` la commande `apply pair_ss` nous ramène à prouver `pair 2`.

Quand on a une hypothèse construite par clôture, comme `x : nat` ou `H : even x`, on peut raisonner dessus par induction, en utilisant la tactique `induction x` ou `induction H`. Le principe d'induction utilisé est généré automatiquement par Coq. Pour afficher celui généré pour `pair`, on utilisera la commande `Check pair_ind`.

1. Définir de la même façon le prédicat `impair`.
2. Prouver que le successeur d'un entier pair est impair, et vice versa.  
(Attention, doit-on raisonner par induction sur l'entier `x` ou sur le prédicat `pair x` ?)
3. Prouver que tout entier est soit pair soit impair.

### Exercice 2 (Mots)

On se donne un ensemble de lettres contenant au moins les symboles `a` et `b`. On utilise ensuite les listes de Coq pour définir un mot comme une liste de caractères :

```
Variable char : Set.
Variable a : char.
Variable b : char.
```

```
Require Import List.
Definition word : Set := list char.
```

Les listes de Coq sont définies comme des termes. La signature comporte une constante correspondant à la liste vide `nil` et de l'opération d'ajout en tête de liste `cons` qu'on peut noter `::`.

Par exemple, `cons a (cons b nil)` (aussi noté `a::b::nil`) représente la liste `[a, b]`.

On dispose de la fonction prédéfinie `app` (aussi notée `++`) qui calcule la concaténation de deux listes.

Par exemple, `app (a::b::nil) (b::nil) = (a::b::nil)++(b::nil) = a::b::b::nil` (`[a, b, b]`).

1. On représente les ensembles de mots par leurs prédicats caractéristiques (un prédicat qui est vrai pour un mot si et seulement si ce mot appartient bien à l'ensemble considéré). Par exemple, l'ensemble ne contenant que le mot vide est représenté par le prédicat suivant :

Definition pempty (w:word) : Prop := (w = nil).

et l'ensemble ne contenant que le mot à une lettre c :

Definition pchar (c:char) (w:word) : Prop := (w = cons c nil).

La concaténation de deux ensembles de mots  $S_1$  et  $S_2$  est définie comme l'ensemble :

$$S_1.S_2 = \{ w \text{ tels que } w = w_1w_2 \text{ avec } w_1 \in S_1 \text{ et } w_2 \in S_2 \}$$

Définir en Coq l'opérateur pconcat tel que pour deux prédicats p1 et p2 (word -> Prop) correspondant à des ensembles de mots, pconcat p1 p2 représente la concaténation de ces ensembles.

(En cas de doute sur la formalisation de cette définition, on peut utiliser des théorèmes tests, par exemple on peut vérifier qu'avec notre définition, on représente bien que le mot (ab) appartient à la concaténation {a}.{b} de {a} et {b}.)

2. Montrer que la concaténation d'un ensemble  $S$  avec l'ensemble vide, est égale à  $S$  :

Theorem pconcat\_empty : forall p w, pconcat p pempty w -> p w.

(On utilisera la tactique simpl, qui permet, par exemple, de remplacer nil ++ l par l. Pour le calcul opposé, on utilisera le lemme suivant (chargé depuis le module list ).

app\_nil\_end : forall (l : word), l = l ++ nil

3. Définir par clôture l'opération "étoile" : pour un ensemble de mots  $S$ ,

$$S^* = \{ w \text{ tels que } w = w_1 \dots w_n \text{ avec } n \geq 0, w_1 \in S, \dots w_n \in S \}$$

(Encore une fois, un théorème test peut aider à vérifier votre formalisation.)

4. Montrer ensuite les théorèmes suivants :

Theorem concat\_star :

forall (p : word -> Prop) w1 w2, p w1 -> pstar p w2 -> pstar p (app w1 w2).

Theorem star\_concat :

forall (p : word -> Prop) w1 w2, pstar p w1 -> p w2 -> pstar p (app w1 w2).

(L'un de ces théorèmes doit être une conséquence immédiate de votre formalisation et l'autre se montre par induction sur la construction étoile. On utilisera également le lemme suivant à propos de la concaténation.)

app\_ass : forall (l m n : word), (l ++ m) ++ n = l ++ m ++ n

5. En déduire finalement que pour tout ensemble de mots  $S$ , on a  $S^*.S = S^*$  :

Theorem star\_pconcat :

forall (p : word -> Prop) w, pconcat (pstar p) p w -> pstar p w.