Université Bordeaux 1. Master d'informatique.
Logique J1IN7M12

Décembre 2012

Sujet de M. Castéran Tous documents autorisés ; durée conseillée : 1h30.
Les 3 exercices sont indépendants

À lire avant d'écrire quoi que ce soit *Le critère d'évaluation sera la qualité et la clarté de vos explications. Les commandes* Coq *non accompagnées d'explications ne seront pas prises en compte ; il vaut mieux expliquer simplement la structure de vos preuves ; par exemple vous pouvez donner des explications de la forme «* En appliquant telle tactique, les nouveaux buts suivants sont engendrés *». Si vous oubliez la syntaxe précise d'une tactique, ou le nom d'une fonction ou d'un théorème, signalez-le sur votre copie.*

1 Exercice

On considère une variable `P` de type `Prop`. Donner des preuves (sans utiliser de tactique automatique) des énoncés suivants :

```
Lemma L1 : (P → ∼P) → ∼P.

Lemma L2 : (forall Q: Prop, P → Q) ↔ ∼P.
```

2 Exercice

On considère les declarations suivantes :

```
Section S2.
Variable A : Type.
Variable P : A → Prop.

Hypothesis H : exists a:A, (P a ∧ forall b:A, P b → a = b).
```

En déduire le lemme suivant (sans utiliser de tactiques automatiques, genre `intuition` ou `first_order`).

```
Lemma L2 : forall x y:A, P x → P y → x = y.
```

3 Exercice

On définit la fonction suivante :

3.1

```
Require Import List.

Fixpoint repeat {A:Type}(a:A)(n:nat) : list A :=
 match n with 0 => nil
            | S p => a :: repeat  a p
 end.
```

Que permet-elle de calculer ?

3.2

Définir en **Coq** le prédicat `Is_repeat` signifiant « La liste l est obtenue par la répétition un nombre quelconque de fois du même élément ». Par exemple, les listes ci-dessous satisfont ce prédicat :

- `33::33::33::nil`
- `false::false::false::false::nil`
- `@nil nat`

En revanche `33::42::33::42::nil`, ne satisfait pas `Is_repeat`.

3.3

Définir en **Coq** une fonction booléenne `is_repeat_nat : list nat → bool` permettant de tester *par calcul* si une liste d'entiers naturels est obtenue la répétition un nombre quelconque de fois du même nombre. *On rappelle que la fonction* `beq_nat : nat → nat → bool` *permet de tester l'égalité de deux entiers naturels.*

3.4

Proposer une version *polymorphe* de la fonction précédente. On donnera son type, sa définition, et on l'appelera `is_repeat`.

3.5

Quel est l'énoncé d'un lemme exprimant la correction de `is_repeat` ? *La preuve de ce lemme est facultative*