Licence STS, semestre 4 — 2014–15
Mathématiques pour l'Informatique 2 — 25 mars 2015
`http://www.lri.fr/~paulin/MathInfo2`

Devoir maison - Préparation au TP noté

Ce devoir est un entraînement pour le TP noté du 7 avril et doit pouvoir être résolu en moins de 2h30 (à l'exception de la dernière question de l'exercice 4, qui est plus difficile). Vous pouvez traiter dès maintenant les deux premiers exercices. Les techniques d'induction utiles pour les troisième et quatrième exercices seront vues en TP lors de la séance du 30-31 mars. Pour faire l'ensemble, vous récupérerez le squelette disponible à l'adresse suivante :
`http://www.lri.fr/~paulin/MathInfo2/dm.v`
Vous devez renvoyer ce fichier complété à votre chargé de TP (`blsk@lri.fr` ou `leon.gondelman@lri.fr`) au plus tard le mardi 7 avril à midi. Vous pouvez travailler seul ou à deux. Dans ce dernier cas, ne renvoyez qu'un fichier pour deux. N'oubliez pas de faire apparaître le nom ou les deux noms en tête du fichier renvoyé.

Exercice 1 (Logique propositionnelle)
Parmi les énoncés suivants, cinq sont vrais. Trouver lesquels, les traduire en Coq et les démontrer. Expliquer pourquoi les autres ne sont pas valides.

(a) $(P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow P \Rightarrow R$
(b) $((P \wedge Q) \Rightarrow R) \Rightarrow (P \Rightarrow Q \Rightarrow R)$
(c) $(P \Rightarrow Q \Rightarrow R) \Rightarrow (P \Rightarrow R)$
(d) $(P \vee Q) \Rightarrow (P \wedge Q)$
(e) $(P \wedge Q) \Rightarrow (P \vee Q)$
(f) $(P \vee Q) \Rightarrow (Q \vee P)$
(g) $(P \vee Q) \Rightarrow (\neg P) \Rightarrow Q$
(h) $(P \Rightarrow Q) \Rightarrow \neg(Q \Rightarrow P)$

Exercice 2 (Quantificateurs)
On se donne un ensemble A et une relation binaire R sur A. Soient les formules

- $F_1 : \forall x, \exists y, (R(x,y) \wedge R(y,x))$
- $F_2 : \forall x, \exists y, (R(x,y) \vee R(y,x))$
- $F_3 : \forall\, x\, y\, z, (R(x,y) \wedge R(y,z) \Rightarrow R(x,z))$
- $F_4 : \exists x, R(x,x)$

1. Traduire ces formules en Coq.
2. Montrer que $F_1 \Rightarrow F_2$
3. Montrer que $F_1 \Rightarrow (F_3 \Rightarrow F_4)$

Exercice 3 (Relations inductives)
Dans un réseau social, deux personnes peuvent décider d'être des « amis ». On modélise cela par un ensemble X d'utilisateurs du réseau et une relation binaire `ami` sur X, telle que `ami x y` est vrai si x et y sont amis.

On souhaite définir la relation « être lié à » (notée `lien`) telle que « x est lié à y » si et seulement si y est un ami de x ou bien si y est lui-même lié à un ami de x. Formellement, on utilisera les deux règles suivantes :

$$\frac{\texttt{ami x y}}{\texttt{lien x y}} \qquad\qquad \frac{\texttt{ami x z} \qquad \texttt{lien z y}}{\texttt{lien x y}}$$

1. Définir la relation `lien` comme une relation inductive (`Inductive lien : X -> X -> Prop := ...`).
2. Montrer à l'aide d'une induction que la propriété suivante est vérifiée :
   `Lemma ami_droite : forall x y z, lien x z -> ami z y -> lien x y`

3. La relation `ami` est symétrique, c'est-à-dire que l'on a :
   `Axiom ami_sym : forall x y, ami x y -> ami y x`
   Montrer à l'aide d'une induction que la relation `lien` est aussi symétrique :
   `Theorem lien_sym : forall x y, lien x y -> lien y x`

Exercice 4 (Récurrence forte et diviseurs premiers)
Le but de cet exercice est de prouver que tout entier naturel strictement supérieur à 1 admet un diviseur premier. Pour cela, il nous faudra établir d'abord le principe de récurrence forte. Pour prouver certaines propriétés arithmétiques, vous pourrez utiliser la tactique `omega`.

Pour tout prédicat `P` sur les entiers, on écrit `upto P n` si on a `P m` pour tout entier `m` tel que `m<n`.

1. Prouver les deux lemmes auxiliaires suivants :
   `Lemma upto_forall : forall P, (forall a, upto P a) -> (forall n, P n)`
   `Lemma rec_upto : forall P, (forall n, upto P n -> P n) -> (forall a, upto P a)`
   Note : l'un de ces deux lemmes nécessite une induction sur un entier (`induction a`).
2. En déduire la validité du principe de récurrence forte :
   `Lemma rec_forte : forall P, (forall n, upto P n -> P n) -> (forall n, P n)`
3. En utilisant la récurrence forte (`apply rec_forte with (n := ...)`), prouver le théorème d'existence d'un diviseur premier :
   `Theorem diviseur_premier : forall n, n>1 -> exists d, divise d n /\ premier d`
   Note : vous pouvez utiliser sans les démontrer les axiomes `premier_ou_divisible` (tout nombre entier `n>1` est premier ou admet un diviseur différent de `1` et de `n`) et `division_transitive` (la divisibilité est transitive), qui sont disponibles dans le squelette.