Preuves assistées par ordinateur

Examen du 29 mai 2015 – durée : 3h

Documents autorisés

Le barème ci-dessous est indicatif et est susceptible d'être modifié.

Exercice 1 (≈ 6 points) – Déduction naturelle

Les séquents suivants sont-ils dérivables en déduction naturelle ? Si oui, en donner une dérivation. Si non, donner un contre-modèle.

1. $(A \Rightarrow C) \vee (B \Rightarrow C) \ \vdash \ (A \wedge B) \Rightarrow C$
2. $(A \wedge B) \Rightarrow C \ \vdash \ (A \Rightarrow C) \vee (B \Rightarrow C)$
3. $\neg(\exists x\, A(x)) \ \vdash \ \forall x\, (\neg A(x))$
4. $\forall x\, (\neg A(x)) \ \vdash \ \neg(\exists x\, A(x))$
5. $(\forall x\, A(x)) \wedge (\forall x\, B(x)) \ \vdash \ \forall x\, (A(x) \wedge B(x))$
6. $\text{PA} \vdash \ A(S\ 0) \ \wedge \ \forall y\, (A(y) \Rightarrow A(S\ y)) \ \Rightarrow \ \forall x \ \ (x \neq 0 \ \Rightarrow \ A(x))$

On rappelle que PA désigne la théorie de Peano.

Exercice 2 : Théories du 1er ordre (≈ 8 pts)

On utilise ici les règles de la déduction naturelle classique, et le tiers-exclus peut être utilisé librement sans être redémontré à partir de la règle de l'absurde. Soit $\mathcal{T}$ une théorie du premier ordre comprenant un symbole de prédicat p d'arité 2 et les trois axiomes suivants :

$$\begin{array}{ll} (A_1) & \forall xy,\ p(x,y) \Rightarrow p(y,x) \\ (A_2) & \forall xyz,\ p(x,y) \Rightarrow p(y,z) \Rightarrow p(x,z) \\ (A_3) & \forall xyz,\ \neg p(x,y) \Rightarrow \neg p(y,z) \Rightarrow \neg p(x,z) \end{array}$$

L'objectif de cet exercice est de montrer que le prédicat p est alors uniforme, c'est-à-dire soit valide pour tous les couples de points, soit invalide partout. Les questions intermédiaires peuvent être utilisées dans les questions suivantes même si elles n'ont pas été finies.

1. On propose une formulation alternative à l'axiome A_3 :

   $$(A'_3) \qquad \forall xyz,\ p(x,z) \Rightarrow (p(x,y) \vee p(y,z))$$

   Démontrer (informellement) le séquent $\vdash A'_3 \Leftrightarrow A_3$. Lequel des deux sens du $\Leftrightarrow$ reste vrai en logique intuitionniste ?
2. À l'aide de modèles bien choisis, montrer que p n'est plus nécessairement uniforme dès qu'un des trois axiomes A_1, A_2 ou A_3 est invalide.
3. Démontrer (informellement) le séquent suivant : $\vdash_{\mathcal{T}} \ \forall xy,\ p(x,x) \Leftrightarrow p(x,y)$

4. Montrer (informellement puis formellement) que dès qu'on a un point x satisfaisant $p(x, x)$, alors p est valide pour tous les couples de points.
5. Conclure en montrant (informellement) que p est uniforme. On pourra raisonner sur l'existence ou non d'un point x tel que $p(x, x)$.
6. Même en partant de A'_3 au lieu de A_3, ce résultat d'uniformité vous semble-t'il démontrable en logique intuitionniste ? Expliquer votre point de vue.

Exercice 3 ($\approx$ 8 points) – Formalisation des langages rationnels en Coq

Le but de cet exercice est de formaliser un fragment de la théorie des langages rationnels en Coq. Les réponses consisteront en une suite de définitions et d'énoncés de lemmes/théorèmes, exprimés dans le langage de Coq, *sans les scripts de preuves correspondants*.

Pour certaines questions, il peut être utile d'introduire des définitions intermédiaires.

Un mot sur un alphabet A est une suite finie d'éléments de A, notée $x_1x_2x_3 \dots x_n$ (par juxtaposition), le mot vide étant noté quant à lui ε. La longueur d'un mot u est notée $|u|$, et la concaténation de deux mots u et v (dans cet ordre) est notée uv.

A – Préliminaires

On suppose le développement Coq paramétré par une variable A : `Type` représentant l'alphabet.

1. Définir un type de données `word` représentant le type des mots ainsi qu'une fonction `length : word → nat` calculant la longueur d'un mot.
2. Définir une fonction `append : word → word → word` qui effectue la concaténation de deux mots. Énoncer (en Coq) ses principales propriétés.

B – Opérations sur les langages

Un langage est un ensemble de mots quelconque, cet ensemble pouvant être infini. On propose de représenter ces langages en Coq via le type `lang := word → Prop` des fonctions caractéristiques de ces ensembles.

3. Définir le langage vide `empty : lang`.
4. Définir la fonction `singleton : A → lang` qui pour une lettre a de l'alphabet A retourne le langage $\{a\}$ réduit au seul mot (à une lettre) a.
5. Définir un prédicat `In : word → lang → Prop` exprimant si un mot appartient à un langage.
6. L'inclusion d'un langage L_1 dans un autre langage L_2 est noté $L_1 \subset L_2$. Définir des relations `subset : lang → lang → Prop` et `equal : lang → lang → Prop` exprimant respectivement l'inclusion et l'égalité de deux langages, et énoncer leurs principales propriétés.
7. L'union de deux langages L_1 et L_2 est notée $L_1 \cup L_2$. Définir une fonction `union : lang → lang → lang` calculant l'union de deux langages, et énoncer ses principales propriétés.
8. La concaténation de deux langages L_1 et L_2 est définie par $L_1L_2 = \{uv \mid u \in L_1,\ v \in L_2\}$. La puissance n-ième d'un langage L est définie par $L^0 = \{\varepsilon\},\ L^{n+1} = L^nL$; et son étoile par $L^* = \bigcup_{n\in\mathbb{N}} L^n$. Définir des fonctions `concat : lang → lang → lang` (concaténation de deux langages), `power : lang → nat → lang` (puissance n-ième d'un langage) et `star : lang → lang` (étoile d'un langage). Énoncer leurs principales propriétés.

C – Expressions régulières

Les expressions régulières sont définies par la syntaxe suivante (avec a élément de A) :

$$E \ := \ \varnothing \mid a \mid E* \mid E_1E_2 \mid E_1 \cup E_2$$

Chaque expression régulière représente un langage de manière évidente : ainsi l'expression régulière $\varnothing$ représente le langage vide, l'expression régulière a le langage $\{a\}$ réduit au seul mot (à une lettre) a, l'expression E_1E_2 la concaténation des langages représentés par les expressions régulières E_1 et E_2, etc. On dit d'un langage qu'il est *rationnel* s'il peut être représenté par une expression régulière.

9. Définir le type `regexp` des expressions régulières.
10. Définir la fonction `lang_of_reg : regexp → lang` qui à chaque expression régulière associe le langage rationnel correspondant.
11. Définir le prédicat `is_rational : lang → Prop` caractérisant les langages rationnels.