Examen de Logique 1 – HLIN402 – session 1

Michel Leclère

9 mai 2019

Durée : 2h. 1 feuille A4 manuscrite recto-verso autorisée. Pas de calculatrice.

Question 1 (5 points)**.** *Syntaxe*
Soit la formule :

$$F \;=\; \neg((b \to \neg\neg a) \vee (c \to d)) \;\vee\; ((a \wedge \neg c) \wedge (b \vee \neg d \;\to\; b \wedge \neg d))$$

a. Dessinez l'arborescence de F.

b. Dites si F est valide, contingente ou insatisfiable en **justifiant votre réponse**.

c. Quelles sont les sous-formules de F dont le connecteur racine est une négation ? ù $\neg((b \to a) \vee (c \to d))$, $\neg\neg a$, $\neg a$, $\neg c$ et $\neg d$

d. Définissez par induction la fonction `ssFbfNeg` qui associe à toute formule bien formée l'ensemble de ses sous-formules ayant une négation comme connecteur racine.

e. Traduisez en Scheme `ssFbfNeg` (cf. annexe pour rappel des fonctions utiles).

Question 2 (3 points)**.** *Questions de cours*

a. Rappelez la définition de l'insatisfiabilité.

b. Rappelez la définition de la conséquence logique.

c. Comment peut-on reformuler un problème de conséquence logique $\{H_1, H_2, \ldots, H_n\} \models C$ (où les H_i et C sont des formules bien formées) à l'aide de l'insatisfiabilité ?

d. Prouvez que si cette propriété d'insatisfiabilité est établie alors la conséquence logique l'est aussi. **Soignez la présentation de la preuve.**

Question 3 (3 points)**.** *Modélisation*
Soit le règlement suivant d'un club écossais :

1. *Tout membre non écossais porte des chaussettes orange ;*
2. *Tout membre porte une jupe ou ne porte pas de chaussettes orange ;*
3. *Les membres mariés ne sortent pas le dimanche ;*
4. *Un membre sort le dimanche si et seulement si il est écossais ;*
5. *Tout membre qui porte une jupe est écossais et marié ;*
6. *Tout membre écossais porte une jupe*

On cherche à savoir s'il peut y avoir un membre dans ce club, c'est à dire si son règlement est consistant.

a. Modélisez chacune des règles de ce club (après avoir précisé quel sens vous donnez à vos symboles propositionnels).

b. Précisez quel problème de logique des propositions permet de répondre à la question.

c. Résolvez ce problème à l'aide de la méthode de votre choix et répondez à la question.

Question 4 (4 points)**.** *Méthodes de preuve*
Montrez que la formule suivante est **valide** :

$$(((r \vee t) \ \rightarrow \ (r \vee s)) \ \ \rightarrow \ \ (r \ \vee \ (t \rightarrow s)))$$

a. En utilisant la méthode de résolution.

b. En utilisant le système LK des séquents de Gentzen (cf. règles en annexe).

Pour chacune de ces méthodes, vous détaillerez bien les étapes du raisonnement. **Soignez la présentation**.

Question 5 (5 points)**.** *Exercice de synthèse*

On se propose de remplacer les deux règles d'inférence à gauche du système LK par une règle compilée en exploitant l'équivalence sémantique suivante : $(P \leftrightarrow Q) \equiv (P \rightarrow Q) \wedge (Q \rightarrow P)$.

a. Prouver l'équivalence logique précédente.

b. Développer "au maximum" l'arbre de preuve du séquent $\Gamma, (A \rightarrow B) \wedge (B \rightarrow A) \ \vdash \ \Delta$.

c. Prouver que l'utilisation de la règle de superdéduction

$$\frac{\Gamma, A, B \vdash \Delta \qquad \Gamma \vdash A, B, \Delta}{\Gamma, A \leftrightarrow B \vdash \Delta} \leftrightarrow_{\mathsf{g}}$$

est adéquate vis-à-vis de la sémantique de la logique des propositions c'est-à-dire prouver que, soit $\Gamma = \{\gamma_1, \gamma_2, \ldots \gamma_n\}$ et $\Delta = \{\delta_1, \delta_2, \ldots \delta_m\}$, on a :

Si $\gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_n \wedge (A \leftrightarrow B) \models \delta_1 \vee \delta_2 \vee \ldots \vee \delta_m$ *alors*
d'une part, $\gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_n \wedge A \wedge B \models \delta_1 \vee \delta_2 \vee \ldots \vee \delta_m$
et d'autre part, $\gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_n \models A \vee B \vee \delta_1 \vee \delta_2 \vee \ldots \vee \delta_m$.

A Annexe règles d'inférence du système LK

$$\frac{}{\Gamma, P \vdash \Delta, P} \ ax$$

$$\frac{}{\Gamma \vdash \Delta, \top} \ \top_{\mathsf{d}}$$

$$\frac{}{\Gamma, \bot \vdash \Delta} \ \bot_{\mathsf{g}}$$

$$\frac{\Gamma \vdash \Delta, P}{\Gamma, \neg P \vdash \Delta} \ \neg_{\mathsf{g}}$$

$$\frac{\Gamma, P \vdash \Delta}{\Gamma \vdash \Delta, \neg P} \ \neg_{\mathsf{d}}$$

$$\frac{\Gamma, P, Q \vdash \Delta}{\Gamma, P \wedge Q \vdash \Delta} \ \wedge_{\mathsf{g}}$$

$$\frac{\Gamma \vdash \Delta, P \qquad \Gamma \vdash \Delta, Q}{\Gamma \vdash \Delta, P \wedge Q} \ \wedge_{\mathsf{d}}$$

$$\frac{\Gamma, P \vdash \Delta \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \vee Q \vdash \Delta} \ \vee_{\mathsf{g}}$$

$$\frac{\Gamma \vdash \Delta, P, Q}{\Gamma \vdash \Delta, P \vee Q} \ \vee_{\mathsf{d}}$$

$$\frac{\Gamma \vdash \Delta, P \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \rightarrow Q \vdash \Delta} \ \rightarrow_{\mathsf{g}}$$

$$\frac{\Gamma, P \vdash \Delta, Q}{\Gamma \vdash \Delta, P \rightarrow Q} \ \rightarrow_{\mathsf{d}}$$

$$\frac{\Gamma \vdash \Delta, P \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \leftrightarrow Q \vdash \Delta} \ \leftrightarrow_{\mathsf{g}_1}$$

$$\frac{\Gamma \vdash \Delta, Q \qquad \Gamma, P \vdash \Delta}{\Gamma, P \leftrightarrow Q \vdash \Delta} \ \leftrightarrow_{\mathsf{g}_2}$$

$$\frac{\Gamma, P \vdash \Delta, Q \qquad \Gamma, Q \vdash \Delta, P}{\Gamma \vdash \Delta, P \leftrightarrow Q} \ \leftrightarrow_{\mathsf{d}}$$

B Annexe : fonctions utiles TP Scheme

```
(define (neg? f) (eq? f '!))

(define (and? f) (eq? f '^))

(define (or? f) (eq? f 'v))

(define (imp? f) (eq? f '->))

(define (equ? f) (eq? f '<->))

(define (top? f) (eq? f 'Top))

(define (bot? f) (eq? f 'Bot))

(define (symbLog? f) (or (top? f) (bot? f) (and? f) (or? f) (neg? f) (imp? f) (equ

(define (conBin? f) (or (and? f) (or? f) (imp? f) (equ? f)))

(define (symbProp? f) (and (symbol? f) (not (symbLog? f))))

(define (atomicFbf? f) (or (symbProp? f) (top? f) (bot? f)))

(define (fbf? f)
  (cond ((atomicFbf? f)     #t )
        ((list? f) (cond ((and (= (length f) 2) (neg? (car f)))    (fbf? (cadr f))
                         ((and (= (length f) 3) (conBin? (car f))) (and (fbf? (cad
                         (else #f)))
        (else #f)))

(define (conRac f) (car f))

(define (fils f) (cadr f))

(define (filsG f) (cadr f))

(define (filsD f) (caddr f))

(define (negFbf? f) (and (not (atomicFbf? f)) (neg? (conRac f))))

(define (andFbf? f) (and (not (atomicFbf? f)) (and? (conRac f))))

(define (orFbf? f) (and (not (atomicFbf? f)) (or? (conRac f))))

(define (impFbf? f) (and (not (atomicFbf? f)) (imp? (conRac f))))

(define (equFbf? f) (and (not (atomicFbf? f)) (equ? (conRac f))))
```

Vous pouvez également utiliser les fonctions de manipulation d'ensemble utilisées dans le TP `(set-union ens1 ens2)`, `(set-intersect ens1 ens2)`, `(set-subtract ens1 ens2)`, `(set-add ens val)`, `(set-remove ens val)`, `(set=? ens1 ens2)`.