Magistère
Informatique et
Télécommunications
1ère année

Devoir maison

corrigé

Question 1. Si on note π la preuve suivante

$$\dfrac{\dfrac{\dfrac{}{(p \Rightarrow ((p \Rightarrow p) \Rightarrow p)) \Rightarrow ((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p))}\,\mathcal{S} \qquad \dfrac{}{p \Rightarrow ((p \Rightarrow p) \Rightarrow p)}\,\mathcal{K}}{(p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p)}\,\mathsf{MP} \qquad \dfrac{}{p \Rightarrow (p \Rightarrow p)}\,\mathcal{K}}{p \Rightarrow p}\,\mathsf{MP}$$

on obtient comme terme de preuve associée $\Phi(\pi) = (\mathcal{S}\ \mathcal{K}\ \mathcal{K})$.

Question 2.

$$\forall f_1, f_2 \in \mathcal{F}, \qquad \dfrac{}{\mathcal{K} : f_1 \Rightarrow (f_2 \Rightarrow f_1)}\,T_{\mathcal{K}}$$

$$\forall f_1, f_2, f_3 \in \mathcal{F}, \qquad \dfrac{}{\mathcal{S} : (f_1 \Rightarrow (f_2 \Rightarrow f_3)) \Rightarrow ((f_1 \Rightarrow f_2) \Rightarrow (f_1 \Rightarrow f_3))}\,T_{\mathcal{S}}$$

$$\forall t_1, t_2 \in \mathcal{T},\ \forall f_1, f_2 \in \mathcal{F}, \qquad \dfrac{t_1 : f_1 \Rightarrow f_2 \qquad t_2 : f_1}{(t_1\ t_2) : f_2}\,T_{\mathsf{MP}}$$

Question 3. La preuve va se dérouler selon deux parties.

1ère partie : $\boxed{\forall A \in \mathcal{F},\ A \text{ est démontrable} \implies A \text{ est habité}}$.

On montre par induction structurelle[1] sur les arbres de preuve du système de Hilbert, la propriété

$$\mathcal{P}(\pi) = \text{« pour toute formule } A \text{ telle que } \pi \text{ soit une preuve de } A,\ A \text{ est habité. »}$$

– cas de base : π est un axiome. Soit f une formule prouvable par π, f est nécessairement de la forme $A \Rightarrow (B \Rightarrow A)$ ou $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$ avec A, B, C des formules. Dans le premier cas, f est habité par $\mathcal{K}$, dans le deuxième par $\mathcal{S}$.

[1] Il est en effet clair, d'après la définition du système de Hilbert, que les arbres de preuve de ce système ont une structure inductive. Le principe d'induction associée est le suivant : pour toute propriété $\mathcal{P}$ sur les arbres de preuves, si

– pour toute formule f, $\mathcal{P}\left(\dfrac{}{f}\,\mathcal{K}\right)$ est vrai

– pour toute formule f, $\mathcal{P}\left(\dfrac{}{f}\,\mathcal{S}\right)$ est vrai

– pour toute formule f, pour tous arbres π_1, π_2 tels que $P(\pi_1)$ et $\mathcal{P}(\pi_2)$ soient vrais, $\mathcal{P}\left(\dfrac{\pi_1 \quad \pi_2}{f}\,\mathrm{MP}\right)$ est vrai

alors, la propriété $\mathcal{P}$ est vérifiée par tout arbre de preuve.

– cas général : π est de la forme $\dfrac{\pi_1 \quad \pi_2}{f}$ MP avec π_1 une preuve d'une formule $g \Rightarrow f$ et π_2 une preuve de g. Par hypothèse d'induction sur π_1 et π_2, $g \Rightarrow f$ et g sont habités par des termes t_1 et t_2. Donc, d'après la règle T_{MP}, f est habité par le terme $(t_1\ t_2)$.

On a ainsi montré que toute formule démontrable est habitée.

2ème partie : $\boxed{\forall A \in \mathcal{F},\ A \text{ est habité} \implies A \text{ est démontrable}}$.

On va s'intéresser à l'énoncé équivalent suivant

$$\forall t \in \mathcal{T},\ \forall A \in \mathcal{F},\ (t : A)\ \Rightarrow\ (\vdash f)$$

On peut alors prouver par induction structurelle sur les termes, la propriété

$$\mathcal{P}(t) = \text{« pour toute formule } A \text{ telle que } t \text{ habite } A,\ A \text{ est démontrable. »}$$

– cas de base : $t = \mathcal{K}$ ou $t = \mathcal{S}$. Si f est une formule habitée par t, f est nécessairement de la forme $A \Rightarrow (B \Rightarrow A)$ ou $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$ avec A, B, C des formules. Dans le premier cas, f est démontrable par l'axiome $\mathcal{K}$, dans le deuxième par $\mathcal{S}$.

– cas général : t est de la forme $(t_1\ t_2)$ avec, par hypothèse d'induction, $(\forall f_1 \in \mathcal{F},\ t_1 : f_1\ \Rightarrow\ \vdash f_1)$ et $(\forall f_2 \in \mathcal{F},\ t_2 : f_2\ \Rightarrow\ \vdash f_2)$. On considère une formule f habités par t et on cherche à prouver que f est démontrable. D'après les règles de typage que nous avons définies, puisque $(t_1\ t_2) : f$, il existe nécessairement une formule g telle que la règle T_{MP} soit vérifiée :

$$\dfrac{t1 : g \Rightarrow f \quad t_2 : g}{(t_1\ t_2) : f}\, T_{\mathsf{MP}}$$

En utilisant l'hypothèse d'induction faite sur t_1, on peut affirmer que $g \Rightarrow f$ est démontrable $(\vdash g \Rightarrow f)$. De la même manière, en utilisant l'hypothèse d'induction faite sur t_2, on peut affirmer que g est démontrable $(\vdash g)$. La règle MP nous permet alors d'affirmer que f est démontrable.

$$\dfrac{\vdash g \Rightarrow f \qquad \vdash g}{\vdash f}\, \mathsf{MP}$$

On a ainsi démontré que toute formule habitée est démontrable.

Question 4.

$$\mathrm{Types}(\mathcal{K}) = \{A \Rightarrow B \Rightarrow A \mid A, B \in \mathcal{F}\}$$

Les types les plus simples de $\mathrm{Types}(\mathcal{K})$ sont ceux de la forme $p \Rightarrow q \Rightarrow p$ avec p et q des variables propositionnelles **distinctes**.

$$\mathrm{Types}(\mathcal{K}) = \{p \Rightarrow q \Rightarrow p \mid p, q \in \mathcal{P},\ p \neq q\}$$

Pour définir la relation $\preceq$, on définit tout d'abord la notion de substitution sur les formules.

Définition : *On appelle* substitution *une liste de couples* $(p_i, g_i)_{1\leq i\leq n}$*, avec pour chaque* i*,* $p_i \in \mathcal{P}$ *et* $g_i \in \mathcal{F}$ *et* n *un entier quelconque. On impose de plus que les* $(p_i)_{1\leq i\leq n}$ *soient distincts. Une telle substitution est notée* $[g_1/p_1, \ldots, g_n/p_n]$.

Définition : *Étant données une formule* f *et une substitution* $\sigma = [g_1/p_1, \ldots, g_n/p_n]$ l'application de la substitution σ sur la formule f (notée $\sigma(f)$) *est définie inductivement par*
- $\sigma(p) = g$ *si* $(p, g) \in \sigma$,
- $\sigma(q) = q$ *si* $(p, g) \notin \sigma$,
- $\sigma(f_1 \Rightarrow f_2) = \sigma(f_1) \Rightarrow \sigma(f_2)$

Définition : *Étant données deux formules* f_1 *et* f_2*,* f_1 est plus simple que f_2 (notée $f_1 \preceq f_2$) *si et seulement si il existe une substitution* σ *telle que* $f_2 = \sigma(f_1)$.

Question 5. On prouve la propriété demandée par induction structurelle sur le terme t, en prenant comme propriété

$$\mathcal{P}(t) = \text{« } \forall f_1, f_2 \in \mathcal{F},\ \text{si } t : f_1 \text{ et } f_1 \preceq f_2 \text{ alors } t : f_2 \text{ »}$$

- case de base 1 : $t = \mathcal{K}$. Soient f_1 et f_2 telles que $\mathcal{K} : f_1$ et $f_1 \preceq f_2$, montrons qu'on a $\mathcal{K} : t_2$. Puisque $f_1 \in \mathrm{Types}(\mathcal{K})$, il existe $A, B \in \mathcal{F}$ telles que $f_1 = A \Rightarrow B \Rightarrow A$. Puisque $f_1 \preceq f_2$, il existe une substitution σ telle que

$$\begin{aligned} f_2 &= \sigma(f_1) \\ &= \sigma(A \Rightarrow B \Rightarrow A) \\ &= \sigma(A) \Rightarrow \sigma(B) \Rightarrow \sigma(A) \qquad \text{par définition de l'application d'une substitution} \end{aligned}$$

  On en déduit que f_2 appartient à $\mathrm{Types}(\mathcal{K})$. On a donc bien $\mathcal{K} : t_2$.
- cas de base 2 : $t = \mathcal{S}$, la preuve est similaire au cas précédent.
- cas général : $t = (t_1\ t_2)$ avec t_1 et t_2 qui vérifient les hypothèse d'induction suivantes

$$\forall g_1, g_2 \in \mathcal{F},\ \text{si } t_1 : g_1 \text{ et } g_1 \preceq g_2 \text{ alors } t_1 : g_2$$

  et

$$\forall h_1, h_2 \in \mathcal{F},\ \text{si } t_2 : h_1 \text{ et } h_1 \preceq h_2 \text{ alors } t_2 : h_2$$

  Soient f_1 et f_2 telles que $(t_1\ t_2) : f_1$ et $f_1 \preceq f_2$, montrons que $(t_1\ t_2) : f_2$.
  D'après la définition de la règle T_{MP}, il existe nécessairement g_1 telle que $t_1 : g_1 \Rightarrow f_1$ et $t_2 : g_1$. On sait, de plus, qu'il existe une substitution σ telle que $f_2 = \sigma(f_1)$.
  Si on considère les formules g_1 et $\sigma(g_1)$, on a $g_1 \preceq \sigma(g_1)$ et $t_2 : g_1$, on peut donc en déduire, en utilisant l'hypothèse d'induction faite sur t_2, que $t_2 : \sigma(g_1)$.
  Si on considère les formules $g_1 \Rightarrow f_1$ et $\sigma(g_1) \Rightarrow f_2$, on a $g_1 \Rightarrow f_1 \preceq \sigma(g_1) \Rightarrow f_2$ (car $\sigma(g_1 \Rightarrow f_1) = \sigma(g_1) \Rightarrow f_2$) et $t_1 : g_1 \Rightarrow f_1$, on peut donc en déduire, en utilisant l'hypothèse d'induction faite sur t_1, que $t_1 : \sigma(g_1) \Rightarrow f_2$.
  La règle de typage T_{MP} permet alors d'affirmer que $(t_1\ t_2)$ admet le type f_2.

$$\frac{t_1 : \sigma(g_1) \Rightarrow f_2 \quad t_2 : \sigma(g_1)}{(t_1\ t_2) : f_2}\, T_{\mathsf{MP}}$$

On a ainsi démontré la propriété demandée.

Question 6. On donne l'algorithme en pseudo-syntaxe Ocaml.

```
let rec infere_type t =
  match t with
    K       => let p = fresh() and q = fresh()  in
                 true,p ⇒ q ⇒ p
  | S       => let p = fresh() and q = fresh() and r = fresh()  n
                 true,(p ⇒ q ⇒ r) ⇒ (p ⇒ q) ⇒ (p ⇒ r)
  | (t1 t2) => match (infere_type t1,infere_type t2) with
                  ((true,f1⇒f2),(true,f3)) =>
                      match unification f1 f3 with
                        (true,σ) => true,σ(f2)
                      | (false,_) => false,∅
               | _ => false,∅
infere_type : terme -> bool * formule
```

Avec `unification : formule -> formule -> bool * substitution` une fonction qui calcule l'unificateur principal des deux formules données en argument et $\emptyset$ une formule quelconque.

Question 7. On suppose que la fonction `unification` utilisée est correcte, c'est à dire :

$$\forall f_1, f_2 \in \mathcal{F},\ (\texttt{unification}\ f_1\ f_2 = (\texttt{true}, \sigma)) \Leftrightarrow \left(\begin{array}{l} \sigma(f_1) = \sigma(f_2) \wedge \\ \forall \sigma', \sigma'(f_1) = \sigma'(f_2) \Rightarrow \exists \beta,\ \sigma' = \beta \circ \sigma \end{array} \right)$$

et

$$\forall f_1, f_2 \in \mathcal{F},\ (\texttt{unification}\ f_1\ f_2 = (\texttt{false}, \_)) \Leftrightarrow \neg\left(\exists \sigma,\ \sigma(f_1) = \sigma(f_2)\right)$$

On démontre alors que `infere_type` est valide, c'est à dire

$$\forall t \in \mathcal{T},\ \text{si } t \text{ est typable, alors } \begin{array}{l} (\texttt{infere\_type}\ t) = (\texttt{true}, f) \\ \text{avec } t : f \text{ et } \forall g \in \mathcal{F},\ t : g \ \Rightarrow\ f \preceq g \end{array} \tag{1}$$

$$\forall t \in \mathcal{T},\ \text{si } t \text{ n'est pas typable, alors } (\texttt{infere\_type}\ t) = (\texttt{false}, \_) \tag{2}$$

Nous allons d'abord nous intéresser à la propriété (2). Nous allons pour cela démontrer le fait suivant

$$\forall t \in \mathcal{T},\ \forall f \in \mathcal{F},\ (\texttt{infere\_type}\ t) = (\texttt{true}, f) \ \Rightarrow\ t : f \tag{3}$$

On réalise, pour cela, une induction structurelle sur t, pour la propriété

$$\mathcal{P}(t) = \text{ « } \forall f \in \mathcal{F},\ (\texttt{infere\_type}\ t) = (\texttt{true}, f) \ \Rightarrow\ t : f \text{ »}$$

- cas de base 1 : $t = \mathcal{K}$. Dans ce cas $(\texttt{infere\_type}\ t) = (\texttt{true}, p \Rightarrow q \Rightarrow p)$ avec p et q des variables propositionnelles. On a bien $\mathcal{K} : p \Rightarrow q \Rightarrow p$.
- cas de base 2 : $t = \mathcal{S}$, cas similaire au cas précédent.

– cas général : t est de la forme $(t_1\ t_2)$ avec t_1 et t_2 des termes vérifiant les hypothèses d'induction $\mathcal{P}(t_1)$ et $\mathcal{P}(t_2)$. On suppose que $(\texttt{infere\_type}\ t)$ est de la forme $(\texttt{true}, f)$ avec f une formule. La définition de `infere_type` implique alors que
  – $(\texttt{infere\_type}\ t_1)$ est de la forme $(\texttt{true}, f_1 \Rightarrow f_2)$
  – $(\texttt{infere\_type}\ t_2)$ est de la forme $(\texttt{true}, f_3)$
  – $(\texttt{unification}\ f_1\ f_3)$ est de la forme $(\texttt{true}, \sigma)$
  – $(\texttt{infere\_type}\ t) = (\texttt{true}, \sigma(f_2))$

  On en déduit alors que, par hypothèse d'induction sur t_1, $t_1 : f_1 \Rightarrow f_2$. De même, par hypothèse d'induction sur t_2, $t_2 : f_3$. Enfin, par correction de `unification`, $\sigma(f_1) = \sigma(f_3)$. En utilisant le résultat de la question 5, on peut alors affirmer que $t_1 : \sigma(f_1) \Rightarrow \sigma(f_2)$ et $t_2 : \sigma(f_3)$. Puisque $\sigma(f_1) = \sigma(f_3)$, nous pouvons utiliser la règle MP pour affirmer que $(t_1\ t_2) : \sigma(f_2)$, ce qui démontre $\mathcal{P}(t)$.

(3) est ainsi démontré. Nous n'avons cependant pas montré grand chose si la fonction `infere_type` ne termine pas ! (le faux implique tout et son contraire...). Il est cependant facile de justifier la terminaison de $(\texttt{infere\_type}\ t)$ pour toute entrée t car les appels récursifs se font uniquement sur des sous termes de t (récursion dite structurelle).
La terminaison de cette fonction et la propriété (3) permettent alors d'affirmer que (2) est vérifiée.
Pour (1), nous allons réaliser une induction sur t avec la propriété suivante.

$$\mathcal{P}(t) = \text{« si } t \text{ est typable, alors } \begin{array}{l} (\texttt{infere\_type}\ t) = (\texttt{true}, f) \\ \text{et } \forall g \in \mathcal{F},\ t : g \ \Rightarrow\ f \preceq g \end{array} \text{»}$$

Cette propriété est suffisante car (3) nous assure déjà que `infere_type` retourne un type valide.

– cas de base 1 : $t = \mathcal{K}$. Dans ce cas $(\texttt{infere\_type}\ t) = (\texttt{true}, p \Rightarrow q \Rightarrow p)$ avec p et q des variables propositionnelles **distinctes** (grâce aux propriétés de la fonction `fresh`). Pour toute formule f vérifiant $\mathcal{K} : f$, f est de la forme $A \Rightarrow B \Rightarrow A$ avec A et B des formules, donc $p \Rightarrow q \Rightarrow p \preceq f$ en prenant pour substitution $[A/p, B/q]$ et en utilisant le fait que p et q sont **distinctes**.
– cas de base 2 : $t = \mathcal{S}$, cas similaire au cas précédent.
– cas général : t est de la forme $(t_1\ t_2)$ avec t_1 et t_2 des termes vérifiant les hypothèses d'induction $\mathcal{P}(t_1)$ et $\mathcal{P}(t_2)$. On suppose que t est typable, donc d'après la règle T_{MP}, t_1 et t_2 sont typables. Les hypothèses d'induction $\mathcal{P}(t_1)$ et $\mathcal{P}(t_2)$ nous donnent alors les faits suivants

$$(\texttt{infere\_type}\ t_1) = (\texttt{true}, f_1), \quad \text{et} \quad \forall g_1 \in \mathcal{F},\ t_1 : g_1 \ \Rightarrow\ f_1 \preceq g_1$$

$$(\texttt{infere\_type}\ t_2) = (\texttt{true}, f_2), \quad \text{et} \quad \forall g_2 \in \mathcal{F},\ t_2 : g_2 \ \Rightarrow\ f_2 \preceq g_2$$

  Or, toute formule habitée est de la forme $A \Rightarrow B$. Ce résultat peut se démontrer de la manière suivante :
  – une formule habitée est démontrable dans le système de Hilbert (question 3)
  – le système de Hilbert est correct : tout formule démontrable dans ce système est une tautologie (car les règles du système sont correctes sémantiquement)
  – aucune variable propositionnelle n'est une tautologie

  On peut ainsi affirmer que f_1 est de la forme $A \Rightarrow B$. Le calcul de $(\texttt{infere\_type}\ t)$ nécessite donc celui de $(\texttt{unification}\ A\ f_2)$.
  Or t est typable donc il existe $g_1, g_2 \in \mathcal{F}$ telles que

$$\frac{t_1 : g_2 \Rightarrow g_1 \quad t_2 : g_2}{t : g_1}\, T_{\mathsf{MP}}$$

Par hypothèses d'induction, $A \Rightarrow B \preceq g_2 \Rightarrow g_1$ et $f_2 \preceq g_2$. Il existe donc des substitutions σ_1 et σ_2 telles que

$$\sigma_1(A) = g_2, \quad \sigma_1(B) = g_1 \quad \text{et } \sigma_2(f_2) = g_2$$

A et f_2 sont donc unifiables. $(\texttt{unification}\ A\ f_2)$ est donc de la forme $(\texttt{true}, \sigma)$ avec σ un unificateur principal de A et f_2. On a ainsi démontré que $(\texttt{infere\_type}\ t)$ est bien de la forme $(\texttt{true}, f)$ avec $f = \sigma(B)$.

Il nous reste à démontrer que $\sigma(B)$ est un des types les plus simples de t. Si g_1 est un type valide pour t, il existe nécessairement une formule g_2 telle que

$$\frac{t_1 : g_2 \Rightarrow g_1 \quad t_2 : g_2}{t : g_1} T_{\mathsf{MP}}$$

Comme précédemment, on arrive alors aux égalités suivantes

$$\sigma_1(A) = g_2, \quad \sigma_1(B) = g_1 \quad \text{et } \sigma_2(f_2) = g_2$$

Grâce à l'utilisation de la fonction `fresh()`, on est assuré que les formules $A \Rightarrow B$ et f_2 n'ont aucune variable en commun. Ceci nous permet d'affirmer que σ_1 et σ_2 peuvent être choisies de façon à ce qu'elle commutent : $\sigma_1 \circ \sigma_2 = \sigma_2 \circ \sigma_1$. On a, de plus, $\sigma_2(A \Rightarrow B) = A \Rightarrow B$ et $\sigma_1(f_2) = f_2$.

La substitution $\sigma' = \sigma_1 \circ \sigma_2$ vérifie ainsi $\sigma'(A) = \sigma'(f_2)$: c'est un unificateur de A et f_2 donc il existe une substitution β vérifiant $\sigma' = \beta \circ \sigma$.

σ' vérifie de plus $\sigma'(B) = g_1$, donc $g_1 = \beta(\alpha(B))$: on a ainsi $\alpha(B) \preceq g_1$. CQFD !

Question 8.

$$(\mathcal{S}\ (\mathcal{S}\ (\mathcal{K}\ \mathcal{S})\ (\mathcal{S}\ (\mathcal{S}\ (\mathcal{K}\ \mathcal{S})\ (\mathcal{K}\ \mathcal{K}))\ (\mathcal{K}\ \mathcal{K})))\ (\mathcal{S}\ (\mathcal{K}\ \mathcal{K})\ (\mathcal{S}\ \mathcal{K}\ \mathcal{K}))) : p \Rightarrow (p \Rightarrow q) \Rightarrow q$$

avec p et q des variables propositionnelles distinctes.

Question 9.

```
let verifie_type t f =
  match infere_type t with
    (true,g) => match unification g f with
                       (true,σ) => σ(g)=f
                     | (false,_) => false
  | (false,_) => false
```

On doit vérifier que f est supérieure à un des plus petits types de t (si t est typable). On utilise pour cela une propriété forte de l'implémentation de l'algorithme d'unification que nous avons réalisé en TP (que nous transposons ici aux formules du système de Hilbert)

$$\forall f_1, f_2 \in \mathcal{F},\ (\texttt{unification}\ f_1\ f_2) = (\texttt{true}, f_1) \Longleftrightarrow f_1 \preceq f_2$$

Pour une autre implémentation de l'unification, le résultat subsiste à renommage près des variables propositionnelles.