TD 12

Exercice 1. Appliquez l'algorithme d'unification « naïf » (exponentiel) vu en cours (voir Figure 1) aux systèmes d'équations suivants. Pouvez vous donner un unificateur autre que le mgu ?

$$
\begin{array}{rlll}
(E \cup \{f(s_1,\dots,s_m)\doteq f(t_1,\dots,t_m)\},\theta) & \to (E \cup \{s_1\doteq t_1,\dots,s_m\doteq t_m\},\theta) & & \text{(Dec)}\\
(E \cup \{f(s_1,\dots,s_m)\doteq g(t_1,\dots,t_n)\},\theta) & \to \mathsf{Fail} & \text{si } f \neq g & \text{(DecFail)}\\
(E \cup \{x\doteq x\},\theta) & \to (E,\theta) & & \text{(Triv)}\\
(E \cup \{x\doteq t\},\theta) & \to (E[x:=t],\theta[x:=t]) & \text{si } x \notin \mathrm{fv}(t) & \text{(Bind)}\\
(E \cup \{t\doteq x\},\theta) & \to (E[x:=t],\theta[x:=t]) & \text{si } x \notin \mathrm{fv}(t) & \text{(Bind')}\\
(E \cup \{x\doteq t\},\theta) & \to \mathsf{Fail} & \text{si } t \neq x \in \mathrm{fv}(t) & \text{(Check)}\\
(E \cup \{t\doteq x\},\theta) & \to \mathsf{Fail} & \text{si } t \neq x \in \mathrm{fv}(t) & \text{(Check')}
\end{array}
$$

FIGURE 1 – Algorithme d'unification de ROBINSON.

1. $\{y\doteq f(x,z), y\doteq f(\dot{3},\dot{5})\}$
2. $\{f(g(x))\doteq f(z), g(z)\doteq g(g(\dot{3}))\}$
3. $\{\mathsf{a}(x,x)\doteq \mathsf{a}(\mathsf{int},\mathsf{a}(\mathsf{int},\mathsf{int}))\}$
4. $\{f(x)\doteq f(f(f(x)))\}$
5. $\{\mathsf{a}(\mathsf{a}(x,\mathsf{int}),\mathsf{int})\doteq \mathsf{a}(y,z), \mathsf{a}(\mathsf{int},y)\doteq \mathsf{a}(z,t)\}$
6. $\{x\doteq \mathsf{a}(x,\mathsf{int})\}$
7. $\{\alpha\doteq\beta\to\beta, \beta\doteq\gamma\to\gamma, \gamma\doteq\delta\to\delta\}$

Exercice 2. Associez à chaque expression pureML ci-dessous un système d'équations en appliquant l'algorithme de HINDLEY (voir Figure 2). Donnez « de tête » son mgu lorsqu'il existe.

1. $x\dot{+}\dot{3}$
2. $\dot{3}\,\dot{4}$
3. $\texttt{letrec}\ f(x) = x\dot{+}\dot{3}\ \texttt{in}\ f\ y$
4. $\texttt{letrec}\ f(x) = \texttt{if}\ x = 0\ \texttt{then}\ \dot{0}\ \texttt{else}\ f\ (x\dot{+}(\dot{-}\dot{1}))\ \texttt{in}\ f\ y$

Exercice 3 (Unification en temps polynomial)**.**

1. Montrez que l'algorithme classique vu en cours (c.f. Figure 1) peut nécessiter un temps exponentiel.
2. Proposez une structure de donnée pour les mgu qui contourne le problème évoqué à la question précédente.
3. Proposez une modification des règles de l'algorithme naïf adapté à cette nouvelle structure.

Exercice 4.

1. Donnez un exemple de terme clos de pureML qui ne type pas en monomorphic pureML.
2. Donnez un exemple de terme clos qui ne type pas en pureML mais qui ne se réduit pas à Wrong.

Pour une expression de monomorphic-pureML rectifiée u_0, c'est-à-dire que chaque variable n'est liée qu'au plus une fois, l'algorithme crée une variable de type fraîche α_t pour chaque sous-terme t de u_0. L'algorithme construit un *système d'équations de types* E_{u_0} en ajoutant inductivement les ensembles d'équations pour chaque sous-terme :

$\{\alpha_{\dot{n}} \doteq \texttt{int}\}$	pour $\dot{n}$
$\{\alpha_u \doteq \alpha_v \to \alpha_{uv}\}$	pour uv
$\{\alpha_x \doteq \alpha_u, \alpha_{\texttt{let } x=u \texttt{ in} v} \doteq \alpha_v\}$	pour $\texttt{let}\, x = u\, \texttt{in}\, v$
$\{\alpha_f \doteq \alpha_x \to \alpha_u, \alpha_{\texttt{letrec } f(x)=u \texttt{ in} v} \doteq \alpha_v\}$	pour $\texttt{letrec}\, f(x) = u\, \texttt{in}\, v$
$\{\alpha_u \doteq \texttt{int}, \alpha_v \doteq \texttt{int}, \alpha_{u \dot{+} v} \doteq \texttt{int}\}$	pour $u \dot{+} v$
$\{\alpha_u \doteq \texttt{int}, \alpha_{\dot{-}u} \doteq \texttt{int}\}$	pour $\dot{-}u$
$\{\alpha_u \doteq \texttt{int}, \alpha_{\texttt{if } u=0 \texttt{ then} v \texttt{ else} w} \doteq \alpha_v, \alpha_{\texttt{if } u=0 \texttt{ then} v \texttt{ else} w} \doteq \alpha_w\}$	pour $\texttt{if}\, u = 0\, \texttt{then}\, v\, \texttt{else}\, w$

On ajoute enfin un contexte de typage Γ_{u_0} formé des hypothèses $x : \alpha_x$ quand x parcourt les variables libres de u_0. On calcule alors le mgu θ de E_{u_0} et on obtient un jugement de typage $\Gamma_{u_0}\theta, \Delta \vdash u_0 : \alpha_{u_0}\theta$ où Δ type les variables non libres de u_0.

FIGURE 2 – Algorithme de typage de HINDLEY.

Exercice 5. En imaginant la généralisation naturelle des règles de typage de pureML, typez le programme suivant :

```
let r = ref (fun x -> x) in
  r := (fun n -> n+1);
  !r "abc" ;;
```

Exercice 6. Écrire en OCaml une fonction `length` pour le type suivant :

```
type 'a mycroft =
  | Nil
  | Cont of 'a * ('a list) mycroft
```

Discutez.