Functional Programming
Exercises Week 11
(for January 13, 2017)

4. Solve the unification problem given by the following equations

$$\begin{aligned} \alpha_1 &\approx \alpha_2 \to \alpha_3 \to \alpha_4 && \text{(a)}\\ \alpha_2 &\approx \mathsf{list}(\alpha_5) && \text{(b)}\\ \alpha_4 &\approx \alpha_3 && \text{(c)}\\ \alpha_3 &\approx \mathsf{list}(\alpha_6) && \text{(d)}\\ \alpha_7 &\approx \alpha_8 \to \mathsf{list}(\alpha_8) \to \mathsf{list}(\alpha_8) && \text{(e)}\\ \alpha_9 &\approx \alpha_0 \to \mathsf{list}(\alpha_0) \to \mathsf{list}(\alpha_0) && \text{(f)}\\ \alpha_0 &\approx \alpha_8 && \text{(g)}\\ \alpha_2 &\approx \mathsf{list}(\alpha_8) && \text{(h)}\\ \mathsf{list}(\alpha_8) &\approx \alpha_4 && \text{(i)}\\ \alpha_4 &\approx \mathsf{list}(\alpha_0) && \text{(k)} \end{aligned}$$

Solution. We have the following derivation:

$$\begin{aligned} & \text{(a–k)}\\ \Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_1/\alpha_2\to\alpha_3\to\alpha_4\}} & \text{(b–k)}\\ \Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_2/\mathsf{list}(\alpha_5)\}} & \text{(c–g)}; \mathsf{list}(\alpha_5) \approx \mathsf{list}(\alpha_8); \text{(i)}; \text{(k)}\\ \Rightarrow^{(\mathsf{d}_1)}_{\iota} & \text{(c–g)}; \alpha_5 \approx \alpha_8; \text{(i)}; \text{(k)}\\ \Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_5/\alpha_8\}} & \text{(c–g)}; \text{(i)}; \text{(k)}\\ \Rightarrow^{(\mathsf{v}_2)}_{\{\alpha_3/\alpha_4\}} & \alpha_4 \approx \mathsf{list}(\alpha_6); \text{(e–g)}; \text{(i)}; \text{(k)}\\ \Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_4/\mathsf{list}(\alpha_6)\}} & \text{(e–g)}; \mathsf{list}(\alpha_8) \approx \mathsf{list}(\alpha_6); \mathsf{list}(\alpha_6) \approx \mathsf{list}(\alpha_0)\\ \Rightarrow^{(\mathsf{d}_1)}_{\iota} & \text{(e–g)}; \alpha_8 \approx \alpha_6; \mathsf{list}(\alpha_6) \approx \mathsf{list}(\alpha_0)\\ \Rightarrow^{(\mathsf{d}_1)}_{\iota} & \text{(e–g)}; \alpha_8 \approx \alpha_6; \alpha_6 \approx \alpha_0\\ \Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_6/\alpha_0\}} & \text{(e–g)}; \alpha_8 \approx \alpha_0\\ \Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_8/\alpha_0\}} & \alpha_7 \approx \alpha_0 \to \mathsf{list}(\alpha_0) \to \mathsf{list}(\alpha_0); \text{(f)}; \alpha_0 \approx \alpha_0\\ \Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_7/\alpha_0\to\mathsf{list}(\alpha_0)\to\mathsf{list}(\alpha_0)\}} & \text{(f)}; \alpha_0 \approx \alpha_0\\ \Rightarrow^{(\mathsf{t})}_{\iota} & \text{(f)}\\ \Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_9/\alpha_0\to\mathsf{list}(\alpha_0)\to\mathsf{list}(\alpha_0)\}} & \square \end{aligned}$$

The corresponding unifier is

$$\{\alpha_1/\mathsf{list}(\alpha_0) \to \mathsf{list}(\alpha_0) \to \mathsf{list}(\alpha_0), \alpha_2/\mathsf{list}(\alpha_0), \alpha_5/\alpha_0, \alpha_3/\mathsf{list}(\alpha_0), \alpha_4/\mathsf{list}(\alpha_0), \alpha_6/\alpha_0, \alpha_8/\alpha_0, \alpha_7/\alpha_0 \to \mathsf{list}(\alpha_0) \to \mathsf{list}(\alpha_0), \alpha_9/\alpha_0 \to \mathsf{list}(\alpha_0) \to \mathsf{list}(\alpha_0)\}$$

5. ($\star$) Define the type checking and type inference rules for a let-rec expression:

```
let rec f x = expr in f
```

and to a let expression with multiple bindings:

```
let x = expr and y = expr in expr
```

Solution.

- For the let-rec expression of the above formulation, a type checking rule would be
$$\frac{E, x : \tau_1, f : \tau_1 \to \tau_2 \vdash e : \tau_2}{E \vdash \texttt{let rec } f\ x \texttt{ = } e \texttt{ in } f : \tau_1 \to \tau_2}$$
and the type inference rule would be
$$\frac{E \rhd \texttt{let rec } f\ x \texttt{ = } e \texttt{ in } f : \tau}{E, x : \alpha_1, f : \alpha_1 \to \alpha_2 \rhd e : \alpha_2;\ \tau \approx \alpha_1 \to \alpha_2}$$
A nicer formulation of let-rec would be
```
let rec x = expr in expr
```
with the rules
$$\frac{E, x : \tau_1 \vdash e_1 : \tau_1 \quad E, x : \tau_1 \vdash e_2 : \tau_2}{E \vdash \texttt{let rec } x \texttt{ = } e_1 \texttt{ in } e_2 : \tau_2}$$
$$\frac{E \rhd \texttt{let rec } x \texttt{ = } e_1 \texttt{ in } e_2 : \tau}{E, x : \alpha_1 \rhd e_1 : \alpha_1;\ E, x : \alpha_1 \rhd e_2 : \tau}$$

- For let expression with multiple bindings, a type checking rule would be
$$\frac{E \vdash e_1 : \tau_1 \quad E \vdash e_2 : \tau_2 \quad E, x : \tau_1, y : \tau_2 \vdash e_3 : \tau_3}{E \vdash \texttt{let } x \texttt{ = } e_1 \texttt{ and } y \texttt{ = } e_2 \texttt{ in } e_3 : \tau_3}$$
and the type inference rule would be
$$\frac{E \vdash \texttt{let } x \texttt{ = } e_1 \texttt{ and } y \texttt{ = } e_2 \texttt{ in } e_3 : \tau}{E \vdash e_1 : \alpha_1;\ E \vdash e_2 : \alpha_2;\ E, x : \alpha_1, y : \alpha_2 \vdash e_3 : \tau}$$