DCC888 – Type Inference

Name: _____ ID: _____

1. The figure below contains the derivation tree of the type-checking rules for the program $((\lambda y.\lambda x.(yx))(\lambda a.a))1$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{y{:}int\to int \in \Gamma, y{:}int\to int, x{:}int}{\Gamma, y{:}T_1\to T_2, x{:}T_1 \vdash y{:}\ int\to int} \quad \dfrac{x{:}int \in \Gamma, y{:}int\to int, x{:}int}{\Gamma, y{:}T_1\to T_2, x{:}T_1 \vdash x{:}\ int}}{\Gamma, y{:}int\to int, x{:}int \vdash y\ x{:}\ int\to int} \quad \dfrac{a{:}int \in \Gamma, a{:}int}{\Gamma, a{:}int \vdash a{:}\ int}}{\Gamma, y{:}int\to int \vdash (\lambda x{:}int.(y\ x)){:}\ int\to int} \quad \dfrac{}{\Gamma \vdash \lambda a{:}int.a{:}\ int\to int}}{\Gamma \vdash ((\lambda y{:}int\to int.(\lambda x{:}int.(y\ x))){:}\ (int\to int)\to int} \quad \dfrac{}{\Gamma \vdash 1 : int}}{\Gamma \vdash ((\lambda y{:}int\to int.(\lambda x{:}int.(y\ x)))\ (\lambda a{:}int.a)){:}\ int\to int}}{\Gamma \vdash ((\lambda y{:}int\to int.(\lambda x{:}int.(y\ x)))\ (\lambda a{:}int.a))\ 1 : int}$$

Each horizontal line represents the application of one of the type-checking rules present in Figure 1: T-True, T-False, T-Nat, T-Var, T-Abs, T-App, T-Add, T-Lth, T-And, and T-If. Name each one of those horizontal lines with the correct type-checking rule.

$$\text{true: bool}\ \ [\text{T-True}] \qquad \text{false: bool}\ \ [\text{T-False}] \qquad \text{n: int}\ \ [\text{T-Nat}] \qquad \frac{x{:}T \in \Gamma}{\Gamma \vdash x{:}T}\ [\text{T-Var}]$$

$$\frac{\Gamma, x{:}T_1 \vdash E{:}T_2}{\Gamma \vdash (\lambda x{:}T.E){:}\ T_1\to T_2}\ [\text{T-Abs}] \qquad \frac{\Gamma \vdash E_1{:}T\to T_1 \quad \Gamma \vdash E_2{:}T}{\Gamma \vdash E_1E_2{:}\ T_1}\ [\text{T-App}]$$

$$\frac{\Gamma \vdash E_1{:}\ int \quad \Gamma \vdash E_2{:}\ int}{\Gamma \vdash E_1 + E_2{:}\ int}\ [\text{T-Add}] \qquad \frac{\Gamma \vdash E_1{:}\ int \quad \Gamma \vdash E_2{:}\ int}{\Gamma \vdash E_1 < E_2{:}\ bool}\ [\text{T-Lth}]$$

$$\frac{\Gamma \vdash E_1{:}\ bool \quad \Gamma \vdash E_2{:}\ bool}{\Gamma \vdash E_1\ and\ E_2{:}\ bool}\ [\text{T-And}] \qquad \frac{\Gamma \vdash E_1{:}\ bool \quad \Gamma \vdash E_2{:}\ T \quad \Gamma \vdash E_3{:}\ T}{\Gamma \vdash \text{if } E_1 \text{ then } E_2 \text{ else } E_3 : T}\ [\text{T-If}]$$

Figure 1: Type-Checking Rules for the simply-typed Lambda Calculus.

2. Build derivation trees, using the type-checking rules in Figure 1 for each one of the following programs:

   (a) $(\lambda x.x)a$

   (b) $(\lambda x.x < x)a$

   (c) $(\lambda x.\text{if } x < x \text{ then } x \text{ else } x)a$

   (d) $\lambda y.(\lambda x.x)y$

3. Use the rules in Figure 2 to generate constraints for the programs below. The type of the entire program is τ_0. Therefore, the first call to the gen function starts with τ_0 as its third parameter. We are providing initial Γ environments (Γ_0) for each item. Γ_0 is the first parameter of the gen function:

   (a) $(\lambda x : \tau_1.x)a, \Gamma_0 = \{a \mapsto \text{bool}\}$

   (b) $(\lambda x : \tau_1.x < x)a, \Gamma_0 = \{a \mapsto \text{int}\}$

   (c) $(\lambda x : \tau_1.\text{if } x < x \text{ then } x \text{ else } x)a, \Gamma_0 = \{a \mapsto \text{int}\}$

   (d) $\lambda y : \tau_2.(\lambda x : \tau_1.x)y, \Gamma_0 = \emptyset$

   As an example, we are showing the derivation tree for $\lambda y.(\lambda x.x)y$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\text{fresh}(\alpha 1)} \quad \dfrac{\dfrac{\dfrac{}{\text{fresh}(\alpha 2)} \quad \dfrac{\dfrac{}{\text{fresh}(\alpha 3)} \quad \{y{\mapsto}\boldsymbol{\tau}2,\ x{\mapsto}\boldsymbol{\tau}1\}(x) = \boldsymbol{\tau}1}{\text{gen}(\{y{\mapsto}\boldsymbol{\tau}2,\ x{\mapsto}\boldsymbol{\tau}1\},\ x,\ \alpha 3) = \alpha 3 \equiv \boldsymbol{\tau}1 = \mathsf{K1}} \quad \{y{\mapsto}\boldsymbol{\tau}2\}(y) = \boldsymbol{\tau}2}{\text{gen}(\{y{\mapsto}\boldsymbol{\tau}2\},\ (\lambda x{:}\boldsymbol{\tau}1.x),\ \alpha 2{\to}\alpha 1) = \alpha 2{\to}\alpha 1 \equiv \boldsymbol{\tau}1{\to}\alpha 3 \wedge \mathsf{K1} = \mathsf{K2}} \quad \text{gen}(\{y{\mapsto}\boldsymbol{\tau}2\},\ y,\ \alpha 2) = \alpha 2 \equiv \boldsymbol{\tau}2 = \mathsf{K3}}{\text{gen}(\{y{\mapsto}\boldsymbol{\tau}2\},\ (\lambda x{:}\boldsymbol{\tau}1.x)\ y,\ \alpha 1) = \mathsf{K2} \wedge \mathsf{K3} = \mathsf{K4}}}{\text{gen}(\{\},\ \lambda y{:}\boldsymbol{\tau}2.\ (\lambda x{:}\boldsymbol{\tau}1.x)\ y,\ \boldsymbol{\tau}0) = \boldsymbol{\tau}0 \equiv \boldsymbol{\tau}2{\to}\alpha 1 \wedge \mathsf{K4}}$$

$$\frac{c \text{ is integer literal}}{\text{gen}(\mathbf{\Gamma}, c, \tau) = \tau \equiv \text{int}} \qquad \frac{\text{gen}(\mathbf{\Gamma}, E_1, \text{bool}) = K_1 \quad \text{gen}(\mathbf{\Gamma}, E_2, \tau) = K_2 \quad \text{gen}(\mathbf{\Gamma}, E_3, \tau) = K_3}{\text{gen}(\mathbf{\Gamma}, \text{if } E_1 \text{ then } E_2 \text{ else } E_3, \tau) = K_1 \wedge K_2 \wedge K_3}$$

$$\text{gen}(\mathbf{\Gamma}, \text{true}, \tau) = \tau \equiv \text{bool}$$

$$\text{gen}(\mathbf{\Gamma}, \text{false}, \tau) = \tau \equiv \text{bool}$$

$$\frac{\text{gen}(\mathbf{\Gamma}, E_1, \text{int}) = K_1 \quad \text{gen}(\mathbf{\Gamma}, E_2, \text{int}) = K_2}{\text{gen}(\mathbf{\Gamma}, E_1 + E_2, \tau) = \tau \equiv \text{int} \wedge K_1 \wedge K_2}$$

$$\frac{\mathbf{\Gamma}[x] = \tau_x}{\text{gen}(\mathbf{\Gamma}, x, \tau) = \tau \equiv \tau_x} \qquad \frac{\text{gen}(\mathbf{\Gamma}, E_1, \text{int}) = K_1 \quad \text{gen}(\mathbf{\Gamma}, E_2, \text{int}) = K_2}{\text{gen}(\mathbf{\Gamma}, E_1 < E_2, \tau) = \tau \equiv \text{bool} \wedge K_1 \wedge K_2}$$

$$\frac{\text{fresh}(\alpha) \quad \text{gen}(\mathbf{\Gamma} \cup \{x \mapsto \tau_x\}, E, \alpha) = K}{\text{gen}(\mathbf{\Gamma}, \lambda x{:}\tau_x.E, \tau) = \tau \equiv \tau_x \to \alpha \wedge K} \qquad \frac{\text{gen}(\mathbf{\Gamma}, E_1, \text{bool}) = K_1 \quad \text{gen}(\mathbf{\Gamma}, E_2, \text{bool}) = K_2}{\text{gen}(\mathbf{\Gamma}, E_1 \text{ and } E_2, \tau) = \tau \equiv \text{bool} \wedge K_1 \wedge K_2}$$

$$\frac{\text{fresh}(\alpha) \quad \text{gen}(\mathbf{\Gamma}, E_1, \alpha \to \tau) = K_1 \quad \text{gen}(\mathbf{\Gamma}, E_2, \alpha) = K_2}{\text{gen}(\mathbf{\Gamma}, E_1 E_2, \tau) = K_1 \wedge K_2}$$

Figure 2: Constraint generation rules.

4. Solve the constraints produced in Exercise (3). Show each step of the resolution process. A solution to each constraint system is a set of equivalence classes. Add arrows between the equivalence classes that you produce for each set of constraints. These arrows indicate dependence relations between equivalence classes. We say that class Q_1 depends on class Q_2 if Q_1 uses, in a type constructor[1] some variable that exists in Q_2. An example for the constraints produced by $\text{gen}(\emptyset, \lambda y.(\lambda x.x)y, \tau_0)$ is shown below:

   1) $\alpha 2 \to \alpha 1 \equiv \boldsymbol{\tau} 1 \to \alpha 3 \wedge \boldsymbol{\tau} 0 \equiv \boldsymbol{\tau} 2 \to \alpha 1 \wedge \alpha 3 \equiv \boldsymbol{\tau} 1 \wedge \alpha 2 \equiv \boldsymbol{\tau} 2$

   2) $\alpha 2 \equiv \boldsymbol{\tau} 1 \wedge \alpha 1 \equiv \alpha 3 \wedge \boldsymbol{\tau} 0 \equiv \boldsymbol{\tau} 2 \to \alpha 1 \wedge \alpha 3 \equiv \boldsymbol{\tau} 1 \wedge \alpha 2 \equiv \boldsymbol{\tau} 2$

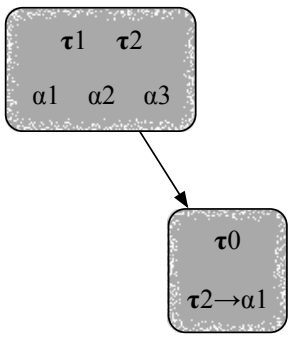
   3) $\{\{\boldsymbol{\tau} 0, \boldsymbol{\tau} 2 \to \alpha 1\}\}: \alpha 2 \equiv \boldsymbol{\tau} 1 \wedge \alpha 1 \equiv \alpha 3 \wedge \alpha 3 \equiv \boldsymbol{\tau} 1 \wedge \alpha 2 \equiv \boldsymbol{\tau} 2$

   4) $\{\{\boldsymbol{\tau} 0, \boldsymbol{\tau} 2 \to \alpha 1\}, \{\boldsymbol{\tau} 1, \alpha 2\}\}: \alpha 1 \equiv \alpha 3 \wedge \alpha 3 \equiv \boldsymbol{\tau} 1 \wedge \alpha 2 \equiv \boldsymbol{\tau} 2$

   5) $\{\{\boldsymbol{\tau} 0, \boldsymbol{\tau} 2 \to \alpha 1\}, \{\boldsymbol{\tau} 1, \alpha 2\}, \{\alpha 1, \alpha 3\}\}: \alpha 3 \equiv \boldsymbol{\tau} 1 \wedge \alpha 2 \equiv \boldsymbol{\tau} 2$

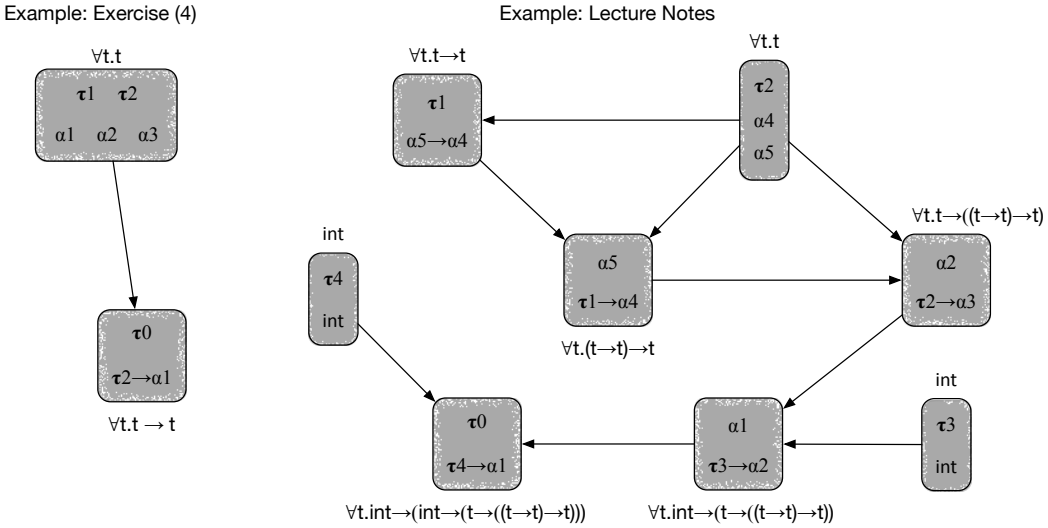
   6) $\{\{\boldsymbol{\tau} 0, \boldsymbol{\tau} 2 \to \alpha 1\}, \{\boldsymbol{\tau} 1, \alpha 1, \alpha 2, \alpha 3\}\}: \alpha 2 \equiv \boldsymbol{\tau} 2$

   7) $\{\{\boldsymbol{\tau} 0, \boldsymbol{\tau} 2 \to \alpha 1\}, \{\boldsymbol{\tau} 1, \boldsymbol{\tau} 2, \alpha 1, \alpha 2, \alpha 3\}\}:$



[1]The only type constructor that we have is the arrow type

5. Find names for each equivalence class that is part of each one of the four constraint systems solved in Exercise (4). To help you, we are showing names for the example used in Exercise (4), and also for a constraint system seen in the lecture notes (available in the slides).



Once you find the right type names for each equivalence class, rename the programs, so that each program passes the type verification rules. We have completed the last item for you, using the solution found in the example from Exercise (4):

(a) $(\lambda x : \tau_1.x)a$

(b) $(\lambda x : \tau_1.x < x)a$

(c) $(\lambda x : \tau_1.\text{if } x < x \text{ then } x \text{ else } x)a$

(d) $\lambda y : \tau_2.(\lambda x : \tau_1.x)y \quad \mapsto \quad \lambda \forall t.y : t.(\lambda x : t \to t.x)y$