

Today, we look at an example term and derive the constraints for it. We also discuss what having a solution for these constraints means.

## 1 Review

The constraint typing relation

$$\Gamma \vdash e : t \mid_X C$$

states that  $e$  has type  $t$  under the context  $\Gamma$  when the constraints in the set  $C$  are fulfilled.  $X$  is the collection of type variables that have been introduced in the subderivations, which is necessary to create fresh type variables.

The set of constraints is a set of type variable-type pairs, stating what type variables should be equal to what types.

## 2 Type Inference Example

Find the constraints for the term  $\lambda x.(x(\lambda y.y))$ .

We begin building our constraint typing derivation from the bottom to the top: Our term is a lambda abstraction, so the CT-ABS rule applies.

$$\frac{\Gamma, (x : O_1) \vdash x.(\lambda y.y) : O_2 \mid_X C}{\emptyset \vdash \lambda x.(x(\lambda y.y)) : O_1 \rightarrow O_2 \mid_X C}$$

$O_1$  and  $O_2$  are used as placeholders and may be instantiated to concrete types in the derivation process. The  $X$  subscripts mean that the sets of type

variables introduced are the same in the antecedent and the consequence. The same applies to the set of constraints  $C$ .

The subterm in the antecedent,  $x(\lambda y.y)$ , is an application, so the CT-APP rule applies. This rule introduces a new free type variable, which we call  $T$ . This forces the placeholder  $O_2$  to be  $T$ .

The set of type variables introduced in this rule is  $X_1 \cup X_2 \cup \{T\}$ , where  $X_1$  and  $X_2$  are the sets of type variables introduced in the two subderivations. The constraints for this rule are  $C_1 \cup C_2 \cup \{O_3 = O_4 \rightarrow T\}$ . Above, we said that the set of type variables and the constraints have to be the same in the antecedent and consequence of the CT-ABS rule. This forces  $X = X_1 \cup X_2 \cup \{T\}$  and  $C = C_1 \cup C_2 \cup \{O_3 = O_4 \rightarrow T\}$  in the consequence of CT-ABS.

$$\frac{\frac{(x : O_1) \vdash x : O_3|_{X_1} C_1 \quad (x : O_1) \vdash \lambda y.y : O_4|_{X_2} C_2}{(x : O_1) \vdash x.(\lambda y.y) : T|_{X_1 \cup X_2 \cup \{T\}} C_1 \cup C_2 \cup \{O_3 = O_4 \rightarrow T\}}}{\emptyset \vdash \lambda x.(x(\lambda y.y)) : O_1 \rightarrow T|_{X_1 \cup X_2 \cup \{T\}} C_1 \cup C_2 \cup \{O_3 = O_4 \rightarrow T\}}$$

The first subterm of the application is a variable, so CT-VAR applies. In the typing context, we find that  $x : O_1$ , so  $O_3$  becomes  $O_1$ . The set of type variables and the set of constraints in this rule are empty, so both  $X_1$  and  $C_1$  are empty everywhere in the typing derivation.

$$\frac{\frac{\frac{(x : O_1) \in \Gamma}{(x : O_1) \vdash x : O_1|\emptyset\{\}} \quad (x : O_1) \vdash \lambda y.y : O_4|_{X_2} C_2}{(x : O_1) \vdash x.(\lambda y.y) : T|_{\emptyset \cup X_2 \cup \{T\}} \{\} \cup C_2 \cup \{O_1 = O_4 \rightarrow T\}}}{\emptyset \vdash \lambda x.(x(\lambda y.y)) : O_1 \rightarrow T|_{\emptyset \cup X_2 \cup \{T\}} \{\} \cup C_2 \cup \{O_1 = O_4 \rightarrow T\}}$$

In the following steps, unnecessary  $\{\}$  or  $\emptyset$  will be dropped.

The second subterm of the application is another lambda abstraction, and therefore CT-ABS applies. We therefore know that  $\lambda y.y$ 's type,  $O_4$ , is an arrow type, which we change to  $O_5 \rightarrow O_6$  everywhere in the derivation. Just like in the first use of the CT-ABS rule, the set of type variables and the set of constraints must be the same in the antecedent and the consequence.

$$\frac{\frac{(x : O_1) \in \Gamma}{(x : O_1) \vdash x : O_1 |_{\emptyset} \{ \}} \quad \frac{(x : O_1), (y : O_5) \vdash y : O_6 |_{X_2} C_2}{(x : O_1) \vdash \lambda y. y : (O_5 \rightarrow O_6) |_{X_2} C_2}}{(x : O_1) \vdash x. (\lambda y. y) : T |_{X_2 \cup \{T\}} C_2 \cup \{O_1 = (O_5 \rightarrow O_6) \rightarrow T\}}}{\emptyset \vdash \lambda x. (x (\lambda y. y)) : O_1 \rightarrow T |_{X_2 \cup \{T\}} C_2 \cup \{O_1 = (O_5 \rightarrow O_6) \rightarrow T\}}$$

The subterm of the second CT-ABS rule,  $y$  is a variable, so CT-VAR applies again. We find that  $y$ 's type is  $O_5$ , so we change the placeholder  $O_6$  to match it. The type variable and constraint sets,  $X_2$  and  $C_2$  are empty, so we change those throughout the derivation, too. To simplify the derivation object, we have completely dropped these empty sets already where possible.

$$\frac{\frac{(x : O_1) \in \Gamma}{(x : O_1) \vdash x : O_1 |_{\emptyset} \{ \}} \quad \frac{(y : O_5) \in \Gamma}{(x : O_1), (y : O_5) \vdash y : O_5 |_{\emptyset} \{ \}}}{(x : O_1) \vdash \lambda y. y : (O_5 \rightarrow O_5) |_{\emptyset} \{ \}}}{\frac{(x : O_1) \vdash x. (\lambda y. y) : T |_{\{T\}} \{O_1 = (O_5 \rightarrow O_5) \rightarrow T\}}{\emptyset \vdash \lambda x. (x (\lambda y. y)) : O_1 \rightarrow T |_{\{T\}} \{O_1 = (O_5 \rightarrow O_5) \rightarrow T\}}}}$$

This is our finished constraint typing derivation. We have one constraint for the term  $\lambda x. (x (\lambda y. y))$ :

$$O_1 = (O_5 \rightarrow O_5) \rightarrow T$$

Interestingly, the placeholders  $O_1$  and  $O_5$  do not get instantiated to a concrete type; they act as metavariables.

### 3 Solutions to Constraints

What does it mean to have a solution for a set of constraints? The set of constraints is a set of pairs that equate type variables and types.

Let  $\sigma$  be a substitution, i. e. a set of pairs of type variables and closed types:  $\{(V_1 := T_1), (V_2 := T_2), \dots\}$ .

A set of constraints  $C$  is satisfied if there exists a substitution  $\sigma$  that equates both sides of all constraints in  $C$ :

$$\exists \text{ solution} \Leftrightarrow \exists \sigma, V_i[\sigma] \equiv T_i[\sigma], \forall (V_i := T_i) \in C$$

Examples of substitutions for the constraint typing relation  $\emptyset \vdash \lambda x.(x(\lambda y.y)) : O_1 \rightarrow T|_{\{T\}}\{O_1 = (O_5 \rightarrow O_5) \rightarrow T\}$ :

$$\begin{aligned} \sigma_1 &= \{(T := \mathbf{int}), \quad (O_1 := (O_5 \rightarrow O_5) \rightarrow \mathbf{int})\} \\ \sigma_2 &= \{(T := \mathbf{int} \rightarrow \mathbf{int}), \quad (O_1 := (O_5 \rightarrow O_5) \rightarrow (\mathbf{int} \rightarrow \mathbf{int}))\} \end{aligned}$$

Even after these substitutions, the placeholder  $O_5$  will not have been assigned a concrete type. It still is a metavariable. For any choice of  $O_5$ , the constraints can be fulfilled; therefore, the term therefore is well-typed for any choice of  $O_5$ . It becomes apparent that this constraint typing system does not give constraints for all terms.