Type Systems

Elaine Li
Office Hours Fridays 1:00pm – 2:00pm

Nisarg Patel
Office Hours Mondays 11:00am – 12:00pm

Definition: Substitution

- Substitution, denoted σ, is a mapping from type variables to types
  - e.g. $\sigma = [X \mapsto \mathsf{Nat}, Y \mapsto Z]$
- Substitution rules

$$
\begin{array}{lcl}
\sigma(\mathsf{X}) & = & \begin{cases} \mathsf{T} & \text{if } (\mathsf{X} \mapsto \mathsf{T}) \in \sigma \\ \mathsf{X} & \text{if } \mathsf{X} \text{ is not in the domain of } \sigma \end{cases} \\
\sigma(\mathsf{Nat}) & = & \mathsf{Nat} \\
\sigma(\mathsf{Bool}) & = & \mathsf{Bool} \\
\sigma(\mathsf{T}_1 \to \mathsf{T}_2) & = & \sigma\mathsf{T}_1 \to \sigma\mathsf{T}_2
\end{array}
$$

Definition: Substitution

- Composition

$$\sigma \circ \gamma = \left[\begin{array}{ll} \mathsf{X} \mapsto \sigma(\mathsf{T}) & \text{for each } (\mathsf{X} \mapsto \mathsf{T}) \in \gamma \\ \mathsf{X} \mapsto \mathsf{T} & \text{for each } (\mathsf{X} \mapsto \mathsf{T}) \in \sigma \text{ with } \mathsf{X} \notin dom(\gamma) \end{array} \right]$$

  - Not commutative!
    - e.g. $\sigma = [X \mapsto int]$, $\gamma = [X \mapsto Z]$

Type Unification

$$
\begin{array}{lll}
\mathit{unify}(C) & = & \text{if } C = \emptyset\text{, then } [\,] \\
& & \text{else let } \{\mathsf{S} = \mathsf{T}\} \cup C' = C \text{ in} \\
& & \quad \text{if } \mathsf{S} = \mathsf{T} \\
& & \qquad \text{then } \mathit{unify}(C') \\
& & \quad \text{else if } \mathsf{S} = \mathsf{X} \text{ and } \mathsf{X} \notin \mathit{FV}(\mathsf{T}) \\
& & \qquad \text{then } \mathit{unify}([\mathsf{X} \mapsto \mathsf{T}]C') \circ [\mathsf{X} \mapsto \mathsf{T}] \\
& & \quad \text{else if } \mathsf{T} = \mathsf{X} \text{ and } \mathsf{X} \notin \mathit{FV}(\mathsf{S}) \\
& & \qquad \text{then } \mathit{unify}([\mathsf{X} \mapsto \mathsf{S}]C') \circ [\mathsf{X} \mapsto \mathsf{S}] \\
& & \quad \text{else if } \mathsf{S} = \mathsf{S}_1 \to \mathsf{S}_2 \text{ and } \mathsf{T} = \mathsf{T}_1 \to \mathsf{T}_2 \\
& & \qquad \text{then } \mathit{unify}(C' \cup \{\mathsf{S}_1 = \mathsf{T}_1, \mathsf{S}_2 = \mathsf{T}_2\}) \\
& & \quad \text{else} \\
& & \qquad \mathit{fail}
\end{array}
$$

Figure 22-2: Unification algorithm

Examples: Unification

1) $\{X = Nat, Y = X \longrightarrow X\}$
2) $\{X \longrightarrow Y = Y \longrightarrow Z, Z = U \longrightarrow W\}$
3) $\{Y = Nat \longrightarrow Y\}$
4) $\{Nat \longrightarrow Nat = X \longrightarrow Y\}$
5) $\{Nat = Nat \longrightarrow Y\}$
6) $\{\}$

$$
\begin{array}{lll}
\mathit{unify}(C) & = & \text{if } C = \emptyset \text{, then } [\,] \\
& & \text{else let } \{\mathsf{S} = \mathsf{T}\} \cup C' = C \text{ in} \\
& & \quad \text{if } \mathsf{S} = \mathsf{T} \\
& & \quad\quad \text{then } \mathit{unify}(C') \\
& & \quad \text{else if } \mathsf{S} = \mathsf{X} \text{ and } \mathsf{X} \notin \mathit{FV}(\mathsf{T}) \\
& & \quad\quad \text{then } \mathit{unify}([\mathsf{X} \mapsto \mathsf{T}]C') \circ [\mathsf{X} \mapsto \mathsf{T}] \\
& & \quad \text{else if } \mathsf{T} = \mathsf{X} \text{ and } \mathsf{X} \notin \mathit{FV}(\mathsf{S}) \\
& & \quad\quad \text{then } \mathit{unify}([\mathsf{X} \mapsto \mathsf{S}]C') \circ [\mathsf{X} \mapsto \mathsf{S}] \\
& & \quad \text{else if } \mathsf{S} = \mathsf{S}_1 \to \mathsf{S}_2 \text{ and } \mathsf{T} = \mathsf{T}_1 \to \mathsf{T}_2 \\
& & \quad\quad \text{then } \mathit{unify}(C' \cup \{\mathsf{S}_1 = \mathsf{T}_1, \mathsf{S}_2 = \mathsf{T}_2\}) \\
& & \quad \text{else} \\
& & \quad\quad \mathit{fail}
\end{array}
$$

Examples: Unification

2) $\{X \longrightarrow Y = Y \longrightarrow Z, Z = U \longrightarrow W\}$

unify({**X** $\longrightarrow$ **Y = Y** $\longrightarrow$ **Z**, Z = U $\longrightarrow$ W})

= unify({Z = U -> W} U {X = Y, Y = Z})

= unify({**Z = U -> W**, X = Y, Y = Z})

= unify([Z $\mapsto$ U -> W]C') ·[Z $\mapsto$ U -> W]

= unify({X = Y, **Y = U -> W**}) ·[Z $\mapsto$ U -> W]

= unify([Y $\mapsto$ U -> W]C') ·[Y $\mapsto$ U -> W] ·[Z $\mapsto$ U -> W]

= unify({**X = U -> W**}) ·[Y $\mapsto$ U -> W] ·[Z $\mapsto$ U -> W]

= unify([X $\mapsto$ Y]C') ·[X $\mapsto$ Y] ·[Y $\mapsto$ U -> W] ·[Z $\mapsto$ U -> W]

= unify([X $\mapsto$ Y][]) ·[X $\mapsto$ Y] ·[Y $\mapsto$ U -> W] ·[Z $\mapsto$ U -> W]

= unify([]) ·[X $\mapsto$ Y] ·[Y $\mapsto$ U -> W] ·[Z $\mapsto$ U -> W]

= [] ·[X $\mapsto$ Y] ·[Y $\mapsto$ U -> W] ·[Z $\mapsto$ U -> W]

= [X $\mapsto$ Y] ·[Y $\mapsto$ U -> W] ·[Z $\mapsto$ U -> W]

$$
\begin{array}{lll}
unify(C) & = & \text{if } C = \emptyset, \text{ then } [\,] \\
& & \text{else let } \{S = T\} \cup C' = C \text{ in} \\
& & \quad \text{if } S = T \\
& & \quad\quad \text{then } unify(C') \\
& & \quad \text{else if } S = X \text{ and } X \notin FV(T) \\
& & \quad\quad \text{then } unify([X \mapsto T]C') \circ [X \mapsto T] \\
& & \quad \text{else if } T = X \text{ and } X \notin FV(S) \\
& & \quad\quad \text{then } unify([X \mapsto S]C') \circ [X \mapsto S] \\
& & \quad \text{else if } S = S_1 \to S_2 \text{ and } T = T_1 \to T_2 \\
& & \quad\quad \text{then } unify(C' \cup \{S_1 = T_1, S_2 = T_2\}) \\
& & \quad \text{else} \\
& & \quad\quad \mathit{fail}
\end{array}
$$

Definition: Inhabitation

- By the Curry-Howard correspondence, type inhabitation is equivalent to propositional validity
- A logical formula is valid if and only if the formula is provable (by constructive logic)

LOGIC	PROGRAMMING LANGUAGES
propositions	types
proposition $P \supset Q$	type P→Q
proposition $P \wedge Q$	type P × Q (see §11.6)
proof of proposition P	term `t` of type P
proposition P is provable	type P is inhabited (by some term)

Examples: Inhabitation

1. a * b -> c
2. c
3. (a -> b) -> (a -> c) -> a -> b * c
4. (a -> (b -> c)) -> ((a -> b) -> (a -> c))
5. (a * b -> c) -> (a -> c, b -> c) either 🌶️