CS 476 – Programming Language Design

William Mansky

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Constraint-Based Type Inference

- We can do this in two steps:
  - First, gather all the constraints on type variables
  - Second, find a solution to the constraints
- For step 1, we need constraints for each typing rule:

$$\frac{\Gamma \vdash l_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash l_2 : \tau_1}{\Gamma \vdash l_1\ l_2 : \tau_2} \quad \Rightarrow \quad \frac{\Gamma \vdash l_1 : \tau_1 \quad \Gamma \vdash l_2 : \tau_2}{\Gamma \vdash l_1\ l_2 : \tau \mid \{\tau_1 = \tau_2 \to \tau\}}$$

- $\Gamma \vdash l : \tau \mid S$ means "l has type τ in context Γ, as long as constraints S are satisfied"

Unification

- Now we have to *solve* the constraints

let unify (c : constraints) : (ident -> typ option) = …

- Unification produces a *substitution* of types for type variables

unify $\{\tau_3 = \mathrm{int}, \tau_4 = \mathrm{int}, \tau_1 = \tau_2 \rightarrow \tau_3, \tau_1 = \mathrm{int} \rightarrow \tau_4\}$ = …

- Exercise: How would you solve this unification problem? How would you figure out the values of all the type variables?

Constraint-Based Type Inference

- Now we have to *solve* the constraints

let unify (c : constraints) : (ident -> typ option) = ...

- Unification produces a *substitution* of types for type variables

unify $\{\tau_3 = \mathrm{int}, \tau_4 = \mathrm{int}, \tau_1 = \tau_2 \rightarrow \tau_3, \tau_1 = \mathrm{int} \rightarrow \tau_4\}$ =

$\{\tau_3 = \mathrm{int}, \tau_4 = \mathrm{int}, \tau_1 = \mathrm{int} \rightarrow \mathrm{int}, \tau_2 = \mathrm{int}\}$

```
let type_of (gamma : context) (e : exp) =
        let (t, c) = get_constraints gamma e in
        let s = unify c in apply_subst s t
```

Unification

- Input: a set of *constraints* of the form $L = R$, where L and R are types with type variables in them
- Output: a *substitution*, a map from type variables to types (which still may have variables in them)
- The output substitution σ should solve all the constraints: for each $L = R$ in the input, $[\sigma]L$ is exactly the same as $[\sigma]R$

The Unification Algorithm

- Pick a constraint $L = R$ from the current set S
- Apply one of the following rules, as appropriate:
    1. Discard
    2. Substitute left
    3. Substitute right
    4. Decompose
- Update the constraint set S and the substitution σ accordingly
- Repeat on the remaining constraints

The Unification Algorithm: Discard

- Applies when the constraint is of the form $T = T$
- Action: remove the constraint from S, while leaving σ and the rest of S unchanged

$$S\colon \{\text{int} = \text{int}, \tau_1 = \tau_2, \dots\}$$

$$\sigma\colon \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \tau_5 \to \tau_6, \dots\}$$

The Unification Algorithm: Discard

- Applies when the constraint is of the form $T = T$
- Action: remove the constraint from S, while leaving σ and the rest of S unchanged

$$S: \{\text{int} = \text{int}, \tau_1 = \tau_2, \ldots\} \Rightarrow \{\tau_1 = \tau_2, \ldots\}$$

$$\sigma: \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \tau_5 \to \tau_6, \ldots\}$$

The Unification Algorithm: Substitute (L)

- Applies when the constraint is of the form $x = T$
- Action: add $\{x \mapsto T\}$ to σ, and apply it to the rest of σ and S

$$S\colon \{\tau_5 = \text{bool}, \tau_1 = \text{int} \to \tau_5, \dots\}$$

$$\sigma\colon \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \tau_5 \to \tau_6, \dots\}$$

The Unification Algorithm: Substitute (L)

- Applies when the constraint is of the form $x = T$
- Action: add $\{x \mapsto T\}$ to σ, and apply it to the rest of σ and S

$$S\text{: } \{\tau_5 = \text{bool}, \tau_1 = \text{int} \to \tau_5, \dots\} \Rightarrow \{\tau_1 = \text{int} \to \text{bool}, \dots\}$$

$$\sigma\text{: } \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \tau_5 \to \tau_6, \dots\} \Rightarrow \{\tau_5 \mapsto \text{bool}, \tau_4 \mapsto \text{bool} \to \tau_6, \dots\}$$

The Unification Algorithm: Substitute (L)

- Applies when the constraint is of the form $x = T$
- Action: add $\{x \mapsto T\}$ to σ, and apply it to the rest of σ and S

$$S: \{\tau_5 = \tau \to \tau_5, \tau_1 = \text{int} \to \tau_5, \dots\}$$

$$\sigma: \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \tau_5 \to \tau_6, \dots\}$$

- “Occurs check”: x must not be free in T

The Unification Algorithm: Substitute (L)

- Applies when the constraint is of the form $x = T$
- Action: add $\{x \mapsto T\}$ to σ, and apply it to the rest of σ and S

$$S\text{: } \{\tau_5 = \tau \rightarrow \tau_5, \tau_1 = \text{int} \rightarrow \tau_5, \dots\} \Rightarrow \text{fail}$$

$$\sigma\text{: } \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \tau_5 \rightarrow \tau_6, \dots\}$$

- “Occurs check”: x must not be free in T

The Unification Algorithm: Substitute (R)

- Applies when the constraint is of the form $T = x$
- Action: add $\{x \mapsto T\}$ to σ, and apply it to the rest of σ and S

$$S\text{: } \{\text{bool} = \tau_5, \tau_1 = \text{int} \to \tau_5, \dots\} \Rightarrow \{\tau_1 = \text{int} \to \text{bool}, \dots\}$$

$$\sigma\text{: } \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \tau_5 \to \tau_6, \dots\} \Rightarrow \{\tau_5 \mapsto \text{bool}, \tau_4 \mapsto \text{bool} \to \tau_6, \dots\}$$

- “Occurs check”: x must not be free in T

The Unification Algorithm: Decompose

- Applies when the constraint is of the form
$$T(\tau_1, \dots \tau_n) = T(\upsilon_1, \dots, \upsilon_n)$$
- Action: add $\tau_1 = \upsilon_1, \dots, \tau_n = \upsilon_n$ to S

S: $\{\tau_6 \to \tau_2 = \tau_5 \to \mathrm{int}, \tau_1 = \mathrm{int} \to \tau_5, \dots\}$

σ: $\{\tau_3 \mapsto \mathrm{int}, \tau_4 \mapsto \tau_5 \to \tau_6, \dots\}$

The Unification Algorithm: Decompose

- Applies when the constraint is of the form
$$T(\tau_1, \ldots \tau_n) = T(\upsilon_1, \ldots, \upsilon_n)$$
- Action: add $\tau_1 = \upsilon_1, \ldots, \tau_n = \upsilon_n$ to S

$S\colon \{\tau_6 \to \tau_2 = \tau_5 \to \mathrm{int}, \tau_1 = \mathrm{int} \to \tau_5, \ldots\} \Rightarrow$

$\{\tau_6 = \tau_5, \tau_2 = \mathrm{int}, \tau_1 = \mathrm{int} \to \tau_5, \ldots\}$

$\sigma\colon \{\tau_3 \mapsto \mathrm{int}, \tau_4 \mapsto \tau_5 \to \tau_6, \ldots\}$

The Unification Algorithm: Decompose

- Applies when the constraint is of the form

$$T(\tau_1, \ldots \tau_n) = T(\upsilon_1, \ldots, \upsilon_n)$$

- Action: add $\tau_1 = \upsilon_1, \ldots, \tau_n = \upsilon_n$ to S

S: $\{\tau_6 \rightarrow \tau_2 = \tau_5 * \text{int}, \tau_1 = \text{int} \rightarrow \tau_5, \ldots\}$

σ: $\{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \tau_5 \rightarrow \tau_6, \ldots\}$

The Unification Algorithm: Decompose

- Applies when the constraint is of the form
$$T(\tau_1, \ldots \tau_n) = T(\upsilon_1, \ldots, \upsilon_n)$$
- Action: add $\tau_1 = \upsilon_1, \ldots, \tau_n = \upsilon_n$ to S

$$S\colon \{\tau_6 \to \tau_2 = \tau_5 * \mathrm{int}, \tau_1 = \mathrm{int} \to \tau_5, \ldots\} \Rightarrow \mathrm{fail}$$
$$\sigma\colon \{\tau_3 \mapsto \mathrm{int}, \tau_4 \mapsto \tau_5 \to \tau_6, \ldots\}$$

- If the constructors or number of arguments are different, no solution exists

The Unification Algorithm

- Pick a constraint $L = R$ from the current set S
- Apply one of the following rules, as appropriate:
    1. Discard
    2. Substitute left
    3. Substitute right
    4. Decompose
- Update the constraint set S and the substitution σ accordingly
- Repeat on the remaining constraints
- When finished, σ will unify all the original constraints

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Constraint-Based Type Inference

- Step 1: gather constraints, outputs pair (τ, S) such that if S can be solved, τ is the type of the expression
- Step 2: unify constraints S, obtain solving substitution σ
- Step 3: apply σ to τ to get the type of the expression

```
let type_of (gamma : context) (e : exp) =
        let (t, c) = get_constraints gamma e in
        let s = unify c in apply_subst s t
```

Constraint-Based Type Inference: Rules

$$\frac{(n \text{ is an integer literal})}{\Gamma \vdash n : \text{int} \mid \{\}} \qquad \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau \mid \{\}}$$

$$\frac{\Gamma \vdash l_1 : \tau_1 \mid S_1 \quad \Gamma \vdash l_2 : \tau_2 \mid S_2}{\Gamma \vdash l_1 \texttt{+} l_2 : \text{int} \mid \{\tau_1 = \text{int}, \tau_2 = \text{int}\} \cup S_1 \cup S_2}$$

$$\frac{\Gamma[x \mapsto \tau_1] \vdash l : \tau_2 \mid S \quad \tau_1 \text{ fresh}}{\Gamma \vdash (\texttt{fun}\ x\ \texttt{->}\ l) : \tau_1 \to \tau_2 \mid S}$$

The Unification Algorithm

- Pick a constraint $L = R$ from the current set S
- Apply one of the following rules, as appropriate:
  1. Discard
  2. Substitute left
  3. Substitute right
  4. Decompose
- Update the constraint set S and the substitution σ accordingly
- Repeat on the remaining constraints
- When finished, σ will unify all the original constraints

Constraint-Based Type Inference: Example

$$\{\} \vdash (\texttt{fun f -> fun x -> f x + f 3}) : \tau_1 \to \tau_2 \to \text{int} \mid S_1$$

$$S_1 = \{\tau_3 = \text{int}, \tau_4 = \text{int}, \tau_1 = \tau_2 \to \tau_3, \tau_1 = \text{int} \to \tau_4\}$$

Constraint-Based Type Inference: Example

$\{\} \vdash$ `(fun f -> fun x -> f x + f 3)` $: \tau_1 \to \tau_2 \to \mathrm{int} \mid S_1$

$$S_1 = \{\tau_4 = \mathrm{int}, \tau_1 = \tau_2 \to \tau_3, \tau_1 = \mathrm{int} \to \tau_4\}$$

$$\sigma = \{\tau_3 \mapsto \mathrm{int}\}$$

Constraint-Based Type Inference: Example

$\{\} \vdash$ `(fun f -> fun x -> f x + f 3)` $: \tau_1 \rightarrow \tau_2 \rightarrow \text{int} \mid S_1$

$$S_1 = \{\tau_4 = \text{int}, \tau_1 = \tau_2 \rightarrow \text{int}, \tau_1 = \text{int} \rightarrow \tau_4\}$$

$$\sigma = \{\tau_3 \mapsto \text{int}\}$$

Constraint-Based Type Inference: Example

$\{\} \vdash$ `(fun f -> fun x -> f x + f 3)` $: \tau_1 \to \tau_2 \to \text{int} \mid S_1$

$$S_1 = \{\tau_1 = \tau_2 \to \text{int}, \tau_1 = \text{int} \to \text{int}\}$$

$$\sigma = \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \text{int}\}$$

Constraint-Based Type Inference: Example

$$\{\} \vdash \texttt{(fun f -> fun x -> f x + f 3)} : \tau_1 \to \tau_2 \to \text{int} \mid S_1$$

$$S_1 = \{\tau_2 \to \text{int} = \text{int} \to \text{int}\}$$

$$\sigma = \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \text{int}, \tau_1 \mapsto \tau_2 \to \text{int}\}$$

Constraint-Based Type Inference: Example

$\{\} \vdash$ `(fun f -> fun x -> f x + f 3)` $: \tau_1 \to \tau_2 \to \mathrm{int} \mid S_1$

$$S_1 = \{\tau_2 = \mathrm{int}, \mathrm{int} = \mathrm{int}\}$$

$$\sigma = \{\tau_3 \mapsto \mathrm{int}, \tau_4 \mapsto \mathrm{int}, \tau_1 \mapsto \tau_2 \to \mathrm{int}\}$$

Constraint-Based Type Inference: Example

$$\{\} \vdash (\texttt{fun f -> fun x -> f x + f 3}) : \tau_1 \rightarrow \tau_2 \rightarrow \text{int} \mid S_1$$

$$S_1 = \{\text{int} = \text{int}\}$$

$$\sigma = \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \text{int}, \tau_1 \mapsto \text{int} \rightarrow \text{int}, \tau_2 \mapsto \text{int}\}$$

Constraint-Based Type Inference: Example

$$\{\} \vdash (\texttt{fun f -> fun x -> f x + f 3}) : \tau_1 \to \tau_2 \to \text{int} \mid S_1$$

$$S_1 = \{\}$$

$$\sigma = \{\tau_3 \mapsto \text{int}, \tau_4 \mapsto \text{int}, \tau_1 \mapsto \text{int} \to \text{int}, \tau_2 \mapsto \text{int}\}$$

$$[\sigma](\tau_1 \to \tau_2 \to \text{int}) = (\text{int} \to \text{int}) \to \text{int} \to \text{int}$$

$$\{\} \vdash (\texttt{fun f -> fun x -> f x + f 3}) : (\text{int} \to \text{int}) \to \text{int} \to \text{int}$$

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.