Principal type inference with subtyping

Stephen Dolan and Alan Mycroft

April 21, 2016

Computer Laboratory
University of Cambridge

select

$$\texttt{select}\ p\ v\ d = \texttt{if}\ (p\ v)\ \texttt{then}\ v\ \texttt{else}\ d$$

select

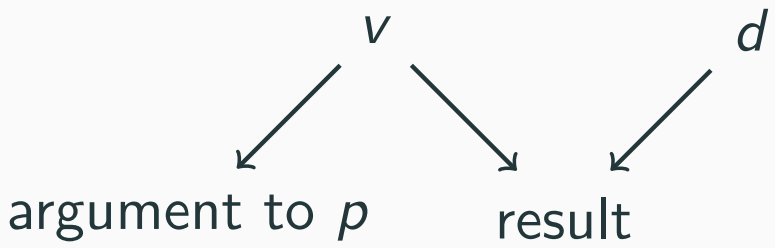
$$\texttt{select}\ p\ v\ d = \texttt{if}\ (p\ v)\ \texttt{then}\ v\ \texttt{else}\ d$$

$$ML : \forall\alpha.(\alpha \to \mathsf{bool}) \to \alpha \to \alpha \to \alpha$$

`select`

$$\texttt{select}\ p\ v\ d = \texttt{if}\ (p\ v)\ \texttt{then}\ v\ \texttt{else}\ d$$

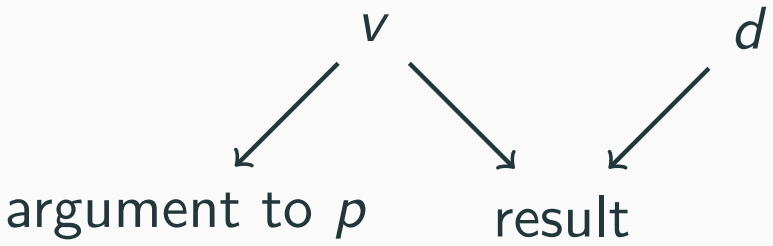
$$ML : \forall\alpha.(\alpha \to \mathsf{bool}) \to \alpha \to \alpha \to \alpha$$



`select`

$$\texttt{select}\ p\ v\ d = \texttt{if}\ (p\ v)\ \texttt{then}\ v\ \texttt{else}\ d$$

$$\mathit{ML} : \forall\alpha.(\alpha \to \mathsf{bool}) \to \alpha \to \alpha \to \alpha$$



$$\mathit{MLsub} : \forall\alpha.(\alpha \to \mathsf{bool}) \to \alpha \to \beta \to (\alpha \sqcup \beta)$$

Defining types

Extensibility

When I add more types,

no programs should break.

Constructing types syntactically

Types:

$$\tau ::= \mathsf{bool} \mid \tau_1 \to \tau_2 \mid \{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\}$$

Constructing types syntactically

Types with subtyping:

$$\tau ::= \mathsf{bool} \mid \tau_1 \to \tau_2 \mid \{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\}$$

$$\frac{}{\mathsf{bool} \leq \mathsf{bool}} \qquad \frac{\tau_1' \leq \tau_1 \quad \tau_2 \leq \tau_2'}{\tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'}$$

$$\frac{}{\{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n, \ldots\} \leq \{\ell_1 : \tau_1, \ldots \ell_n : \tau_n\}}$$

$$\frac{\tau_1 \leq \tau_1' \quad \ldots \quad \tau_n \leq \tau_n'}{\{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\} \leq \{\ell_1 : \tau_1', \ldots, \ell_n : \tau_n'\}}$$

Constructing types syntactically

Types with subtyping, least and greatest types:

$$\tau ::= \mathsf{bool} \mid \tau_1 \to \tau_2 \mid \{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\} \mid \bot \mid \top$$

$$\frac{}{\mathsf{bool} \leq \mathsf{bool}} \qquad \frac{\tau_1' \leq \tau_1 \quad \tau_2 \leq \tau_2'}{\tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'}$$

$$\frac{}{\{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n, \ldots\} \leq \{\ell_1 : \tau_1, \ldots \ell_n : \tau_n\}}$$

$$\frac{\tau_1 \leq \tau_1' \quad \ldots \quad \tau_n \leq \tau_n'}{\{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\} \leq \{\ell_1 : \tau_1', \ldots, \ell_n : \tau_n'\}}$$

$$\frac{}{\top \leq \tau} \qquad \frac{}{\tau \leq \top}$$

Constructing types syntactically

Types with lattice subtyping:

$$\tau ::= \mathsf{bool} \mid \tau_1 \to \tau_2 \mid \{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\} \mid \bot \mid \top$$

$$\frac{}{\mathsf{bool} \leq \mathsf{bool}} \qquad \frac{\tau_1' \leq \tau_1 \quad \tau_2 \leq \tau_2'}{\tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'}$$

$$\frac{}{\{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n, \ldots\} \leq \{\ell_1 : \tau_1, \ldots \ell_n : \tau_n\}}$$

$$\frac{\tau_1 \leq \tau_1' \quad \ldots \quad \tau_n \leq \tau_n'}{\{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\} \leq \{\ell_1 : \tau_1', \ldots, \ell_n : \tau_n'\}}$$

$$\frac{}{\top \leq \tau} \qquad \frac{}{\tau \leq \top}$$

What's going on here?

$$\mathcal{F}(A) = \mathsf{Bool}(A) + \mathsf{Func}(A) + \mathsf{Rec}(A)$$

$$\mathsf{Bool}(A) = \mathsf{bool}$$

$$\mathsf{Func}(A) = A \times A$$

$$\mathsf{Rec}(A) = L \rightharpoonup A$$

Take the *initial algebra* of $\mathcal{F} : \mathbf{Set} \to \mathbf{Set}$

What's going on here?

$$\mathcal{F}(A) = \mathsf{Bool}(A) + \mathsf{Func}(A) + \mathsf{Rec}(A)$$

$$\mathsf{Bool}(A) = \mathsf{bool}$$
$$\mathsf{Func}(A) = A^{-} \times A$$
$$\mathsf{Rec}(A) = L \rightharpoonup A$$

Take the *initial algebra* of $\mathcal{F} : \mathbf{Pos} \to \mathbf{Pos}$.

What's going on here?

$$\mathcal{F}(A) = \mathsf{Bool}(A) + \mathsf{Func}(A) + \mathsf{Rec}(A)$$

$$\mathsf{Bool}(A) = (\mathsf{bool})^{\top}_{\bot}$$
$$\mathsf{Func}(A) = (A^{-} \times A)^{\top}_{\bot}$$
$$\mathsf{Rec}(A) = (L \rightharpoonup A)^{\top}_{\bot}$$

Take the *initial algebra* of $\mathcal{F} : \mathbf{Pos}^{\top}_{\bot} \to \mathbf{Pos}^{\top}_{\bot}$.

What's going on here?

$$\mathcal{F}(A) = \mathsf{Bool}(A) + \mathsf{Func}(A) + \mathsf{Rec}(A)$$

$$\mathsf{Bool}(A) = (\mathsf{bool})^{\top}_{\bot}$$
$$\mathsf{Func}(A) = (A^{-} \times A)^{\top}_{\bot}$$
$$\mathsf{Rec}(A) = (L \rightharpoonup A)^{\top}_{\bot}$$

Take a lattice of types, forget that it's a lattice, apply $\mathcal{F} : \mathbf{Pos}^{\top}_{\bot} \to \mathbf{Pos}^{\top}_{\bot}$, and notice that the result happens to be a lattice.

What's going on here?

$$\mathcal{F}(A) = \mathsf{Bool}(A) + \mathsf{Func}(A) + \mathsf{Rec}(A)$$

$$\mathsf{Bool}(A) = (\mathsf{bool})^{\top}_{\bot}$$
$$\mathsf{Func}(A) = (A^{-} \times A)^{\top}_{\bot}$$
$$\mathsf{Rec}(A) = (L \rightharpoonup A)^{\top}_{\bot}$$

Take the *initial algebra* of $\mathcal{F} : \mathbf{Lat} \to \mathbf{Lat}$.

Coproducts

$$\tau ::= \mathsf{bool} \mid \tau_1 \to \tau_2 \mid \{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\}$$

The $\mid$ should be the coproduct. Not always union!

Coproducts

$$\tau ::= \mathsf{bool} \mid \tau_1 \to \tau_2 \mid \{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\}$$

The $\mid$ should be the coproduct. Not always union!

We get some new types:

$$(\tau \to \tau) \sqcap \{\texttt{foo} : \tau\}$$

Coproducts

$$\tau ::= \mathsf{bool} \mid \tau_1 \to \tau_2 \mid \{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\}$$

The $\mid$ should be the coproduct. Not always union!

We get some new types:

$$(\tau \to \tau) \sqcap \{\texttt{foo} : \tau\} \leq \mathsf{bool}$$

Type variables

How about type variables as *metavariables*?

Type variables

How about type variables as *metavariables*?

Pottier's tricky example:

$$\tau = \bot \mid \tau \to \tau \mid \top$$

$$(\bot \to \top) \to \bot \;\leq\; (\alpha \to \bot) \sqcup \alpha$$

This is true, by case analysis...

Type variables

How about type variables as *metavariables*?

Pottier's tricky example:

$$\tau = \bot \mid \tau \to \tau \mid \top$$

$$(\bot \to \top) \to \bot \;\leq\; (\alpha \to \bot) \sqcup \alpha$$

This is true, by case analysis... until we extend the type system

$$\alpha = (\top \xrightarrow{\circ} \bot) \xrightarrow{\circ} \bot$$

Type variables

How about type variables as *metavariables*?

Pottier's tricky example:

$$\tau = \bot \mid \tau \to \tau \mid \top$$

$$(\bot \to \top) \to \bot \;\leq\; (\alpha \to \bot) \sqcup \alpha$$

This is true, by case analysis... until we extend the type system

$$\alpha = (\top \overset{\circ}{\to} \bot) \overset{\circ}{\to} \bot$$

We need type variables as a *free algebra*. Syntactically, add them as opaque indeterminates:

$$\tau ::= \mathsf{bool} \mid \tau_1 \to \tau_2 \mid \{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\} \mid \alpha$$

Inferring types

Polarity

We have lots of types, but they're not all useful all the time.

$\tau_1 \sqcup \tau_2$ describes an output that may be τ_1 or may be τ_2.

$\tau_1 \sqcap \tau_2$ describes an input that must be τ_1 and must be τ_2

Polarity

We have lots of types, but they're not all useful all the time.

$\tau_1 \sqcup \tau_2$ describes an output that may be τ_1 or may be τ_2.

$\tau_1 \sqcap \tau_2$ describes an input that must be τ_1 and must be τ_2

Use *positive types* τ^+ for outputs, and *negative types* τ^- for inputs:

$$\begin{aligned} \tau^+ &::= \alpha \mid \tau^+ \sqcup \tau^+ \mid \bot \mid \mathsf{unit} \mid \tau^- \to \tau^+ \\ \tau^- &::= \alpha \mid \tau^- \sqcap \tau^- \mid \top \mid \mathsf{unit} \mid \tau^+ \to \tau^- \end{aligned}$$

When does ML unify?

Two inputs: introduce $\sqcap$

Two outputs: introduce $\sqcup$

Output used as an input: constraint $\tau^+ \leq \tau^-$

When does ML unify?

Two inputs: introduce $\sqcap$

Two outputs: introduce $\sqcup$

Output used as an input: constraint $\tau^+ \leq \tau^-$

Handling $\sqcup$ and $\sqcap$ is easy, thanks to polarity:

$$\tau_1 \sqcup \tau_2 \leq \tau_3 \text{ iff } \tau_1 \leq \tau_3, \tau_2 \leq \tau_3$$

$$\tau_1 \leq \tau_2 \sqcap \tau_3 \text{ iff } \tau_1 \leq \tau_2, \tau_1 \leq \tau_3$$

Atomic constraints

How do we get rid of $\alpha \leq \tau^-$ or $\tau^+ \leq \alpha$? Not substitution!

Atomic constraints

How do we get rid of $\alpha \leq \tau^-$ or $\tau^+ \leq \alpha$? Not substitution!

$$\forall \neq \Lambda$$

Atomic constraints

How do we get rid of $\alpha \leq \tau^-$ or $\tau^+ \leq \alpha$? Not substitution!

$$\forall \neq \Lambda$$

ML's $\forall$ is more like a set comprehension. These are the same:

$$\forall\alpha\forall\beta.\ \alpha \to \beta \to \alpha$$
$$\forall\beta\forall\alpha.\ \alpha \to \beta \to \alpha$$

because these are the same

$$\{\alpha \to \beta \to \alpha \mid \alpha, \beta \text{ types}\}$$
$$\{\alpha \to \beta \to \alpha \mid \beta, \alpha \text{ types}\}$$

Preserving the set of instances

It's easy to remove constraints from set comprehensions:

Preserving the set of instances

It's easy to remove constraints from set comprehensions:

$$\{n^2 + 17 \mid n \in \mathbb{N},\ n \geq 5\}$$

Preserving the set of instances

It's easy to remove constraints from set comprehensions:

$$\begin{aligned} &\{n^2 + 17 \mid n \in \mathbb{N},\ n \geq 5\} \\ = &\{\max(n, 5)^2 + 17 \mid n \in \mathbb{N}\} \end{aligned}$$

Preserving the set of instances

It's easy to remove constraints from set comprehensions:

$$\{n^2 + 17 \mid n \in \mathbb{N},\ n \geq 5\}$$
$$= \{\mathsf{max}(n, 5)^2 + 17 \mid n \in \mathbb{N}\}$$

Thinking of type schemes as set comprehensions, we can do the same:

$$\alpha \to \alpha \mid \alpha \leq \{\texttt{isgood} : \mathsf{bool}\}$$

Preserving the set of instances

It's easy to remove constraints from set comprehensions:

$$\begin{aligned} &\{n^2 + 17 \mid n \in \mathbb{N},\ n \geq 5\} \\ =\ &\{\mathsf{max}(n, 5)^2 + 17 \mid n \in \mathbb{N}\} \end{aligned}$$

Thinking of type schemes as set comprehensions, we can do the same:

$$\begin{aligned} &\alpha \to \alpha \mid \alpha \leq \{\texttt{isgood} : \mathsf{bool}\} \\ =\ &(\alpha \sqcap \{\texttt{isgood} : \mathsf{bool}\}) \to (\alpha \sqcap \{\texttt{isgood} : \mathsf{bool}\}) \end{aligned}$$

Preserving the set of instances

It's easy to remove constraints from set comprehensions:

$$\begin{aligned} &\{n^2 + 17 \mid n \in \mathbb{N},\ n \geq 5\} \\ =\ &\{\mathsf{max}(n, 5)^2 + 17 \mid n \in \mathbb{N}\} \end{aligned}$$

Thinking of type schemes as set comprehensions, we can do the same:

$$\begin{aligned} &\alpha \to \alpha \mid \alpha \leq \{\texttt{isgood} : \mathsf{bool}\} \\ =\ &(\alpha \sqcap \{\texttt{isgood} : \mathsf{bool}\}) \to (\alpha \sqcap \{\texttt{isgood} : \mathsf{bool}\}) \\ =\ &(\alpha \sqcap \{\texttt{isgood} : \mathsf{bool}\}) \to \alpha \end{aligned}$$

Example

$$\mathsf{filter\_good} : \mathsf{list}[\{\texttt{isgood} : \mathsf{bool}\} \sqcap \alpha] \to \mathsf{list}[\alpha]$$

$$\mathsf{things} : \mathsf{list}[\{\texttt{isgood} : \mathsf{bool}, \texttt{x} : \mathsf{int}\}]$$

Example

$$\mathsf{filter\_good} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}\} \sqcap \alpha] \to \mathsf{list}[\alpha]$$

$$\mathsf{things} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\}]$$

$$\mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}\} \sqcap \alpha] \to \mathsf{list}[\alpha] \leq \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\}] \to \beta$$

Example

$$\mathsf{filter\_good} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}\} \sqcap \alpha] \to \mathsf{list}[\alpha]$$

$$\mathsf{things} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\}]$$

$$\mathsf{list}[\alpha] \leq \beta$$

$$\mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\}] \leq \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}\} \sqcap \alpha]$$

Example

$$\mathsf{filter\_good} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}\} \sqcap \alpha] \to \mathsf{list}[\alpha]$$

$$\mathsf{things} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\}]$$

$$\mathsf{list}[\alpha] \leq \beta$$

$$\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\} \leq \{\mathtt{isgood} : \mathsf{bool}\} \sqcap \alpha$$

Example

$$\mathsf{filter\_good} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}\} \sqcap \alpha] \to \mathsf{list}[\alpha]$$

$$\mathsf{things} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\}]$$

$$\mathsf{list}[\alpha] \leq \beta$$

$$\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\} \leq \{\mathtt{isgood} : \mathsf{bool}\}$$

$$\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\} \leq \alpha$$

Example

$$\mathsf{filter\_good} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}\} \sqcap \alpha] \to \mathsf{list}[\alpha]$$

$$\mathsf{things} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\}]$$

$$\mathsf{list}[\alpha] \leq \beta$$

$$\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\} \leq \alpha$$

Example

$$\mathsf{filter\_good} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}\} \sqcap \alpha] \to \mathsf{list}[\alpha]$$

$$\mathsf{things} : \mathsf{list}[\{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\}]$$

$$\beta^+ \mapsto \beta \sqcup \mathsf{list}[\alpha]$$

$$\alpha^+ \mapsto \alpha \sqcup \{\mathtt{isgood} : \mathsf{bool}, \mathtt{x} : \mathsf{int}\}$$

Example

$$\mathsf{filter\_good} : \mathsf{list}[\{\texttt{isgood} : \mathsf{bool}\} \sqcap \alpha] \to \mathsf{list}[\alpha]$$

$$\mathsf{things} : \mathsf{list}[\{\texttt{isgood} : \mathsf{bool}, \texttt{x} : \mathsf{int}\}]$$

$$\beta \sqcup \mathsf{list}[\alpha \sqcup \{\texttt{isgood} : \mathsf{bool}, \texttt{x} : \mathsf{int}\}]$$

Example

$$\mathsf{filter\_good} : \mathsf{list}[\{\texttt{isgood} : \mathsf{bool}\} \sqcap \alpha] \to \mathsf{list}[\alpha]$$

$$\mathsf{things} : \mathsf{list}[\{\texttt{isgood} : \mathsf{bool}, \texttt{x} : \mathsf{int}\}]$$

$$\mathsf{list}[\{\texttt{isgood} : \mathsf{bool}, \texttt{x} : \mathsf{int}\}]$$

Questions?

http://www.cl.cam.ac.uk/~sd601/mlsub

stephen.dolan@cl.cam.ac.uk