

Type Inference with Polarization: $\forall \pm \exists$

Ilya Kaysin

Supervisor:
Dr Neel Krishnaswami



UNIVERSITY OF
CAMBRIDGE

Types

False : Bool
239 : Int
[False] : [Bool]
[2, 3] : [Int]

length : [Bool] → Int

Types

True : Bool
239 : Int
[False] : [Bool]
[2, 3] : [Int]

length : [Bool] → Int

Types

True : Bool

42 : Int

[False] : [Bool]

[2, 3] : [Int]

length : [Bool] → Int

Types

True : Bool

42 : Int

[False] : [Bool]

[2, 3, 9] : [Int]

length : [Bool] → Int

Types

True : Bool

42 : Int

[False] : [Bool]

[2, 3, 9] : [Int]

length : [Bool] → Int

length [True, False] = 2

Types

True : Bool
42 : Int
[False] : [Bool]
[2, 3, 9] : [Int]

length : [Bool] → Int

length [True, False] = 2

length [2, 3, 9] =  TypeError 

Types

True : Bool
42 : Int
[False] : [Bool]
[2, 3, 9] : [Int]

length : [Int] → Int

length [True, False] = 2

length [2, 3, 9] =  TypeError

Types

True : Bool
42 : Int
[False] : [Bool]
[2, 3, 9] : [Int]

length : [Int] → Int

length [True, False] = 💀TypeError💀

length [2, 3, 9] = 3

Types: \forall

True : Bool

42 : Int

[False] : [Bool]

[2, 3, 9] : [Int]

length : $\forall a. [a] \rightarrow \text{Int}$

length [True, False] =

length [2, 3, 9] =

Types: \forall

True : Bool

42 : Int

[False] : [Bool]

[2, 3, 9] : [Int]

length : $\forall a. [a] \rightarrow \text{Int}$

length [True, False] = 2

length [2, 3, 9] = 3

Types: \forall

True : Bool
42 : Int
[False] : [Bool]
[2, 3, 9] : [Int]

length : $\forall a. [a] \rightarrow \text{Int}$

$\leqslant [Bool] \rightarrow \text{Int}$
 $\leqslant [\text{Int}] \rightarrow \text{Int}$

length [True, False] = 2

length [2, 3, 9] = 3

Types: \forall

True : Bool
42 : Int
[False] : [Bool]
[2, 3, 9] : [Int]

length : $\forall a. [a] \rightarrow \text{Int}$

$\leqslant [Bool] \rightarrow \text{Int}$
 $\leqslant [\text{Int}] \rightarrow \text{Int}$

length [True, False] = 2

length [2, 3, 9] = 3

Types: \forall

True : Bool
42 : Int
[False] : [Bool]
[2, 3, 9] : [Int]

length : $\forall a. [a] \rightarrow \text{Int}$

$\leqslant [Bool] \rightarrow \text{Int}$
 $\leqslant [\text{Int}] \rightarrow \text{Int}$

length [True, False] = 2

length [2, 3, 9] = 3

Subtyping (\leq)

is the key to type inference

$P \leq Q$ = any **expr : P** is safe to use where **Q** is expected

$$\text{If: } [T/a]P \leq Q$$

$$\text{Then: } \forall a. P \leq Q$$

$$P \leq P$$

$$N \leq M$$

$$P \geq Q$$

$$P \approx Q$$

$$P \rightarrow N \leq Q \rightarrow M$$

$$[P] \leq [Q]$$

Types: 3

initial state	old state	output	new state	
				Stream of booleans
				F, T, F, T, ...

Types: \exists

initial state	old state	output	new state	
				Stream of booleans
Int × (Int → Bool × Int)				
(0,	$\lambda s. (\text{odd } s, s + 1)$)			F, T, F, T, ...
(F,	$\lambda s. (s, \text{not } s)$)			F, T, F, T, ...
Bool × (Bool → Bool × Bool)				Stream of booleans

Types: 3

initial state **old** state output **new** state
Int × (**Int** → **Bool** × **Int**) Stream of booleans

¶

||

Bool × (**Bool** → **Bool** × **Bool**) Stream of booleans

Types: 3

initial state	old state	output	new state	
				Stream of booleans
				Stream of booleans
				Stream of booleans

Int × (Int → Bool × Int)

$\exists a. a \times (a \rightarrow \text{Bool} \times a)$

Bool × (Bool → Bool × Bool)

Types: \exists

initial state	old state	output	new state	
				Stream of booleans
				Stream of booleans
				Stream of booleans

Goal: developing a type checking algorithm for polymorphic systems with existential types

Objectives:

1. Develop the declarative system specifying the **decidable segment** of the language
2. Devise a sound and complete **subtyping** algorithm
3. Apply the subtyping for the type **inference**

The Algorithm

?

$$\forall a. [a] \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow \text{Int}$$

Subtyping → Inference

The Algorithm

?

$$[\hat{\alpha}] \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow \text{Int}$$

$$\forall a. [a] \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow \text{Int}$$

Subtyping → Inference

The Algorithm

$$\frac{\begin{array}{c} ? \\ \hline [\hat{\alpha}] \geq [\text{Bool}] & \text{Int} \leq \text{Int} \end{array}}{\begin{array}{c} [\hat{\alpha}] \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow \text{Int} \\ \hline \forall a. [a] \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow \text{Int} \end{array}}$$

Subtyping → Inference

The Algorithm

$$\begin{array}{c} \hat{\alpha} \approx \text{Bool} \\ \hline [\hat{\alpha}] \geq [\text{Bool}] \quad \text{Int} \leq \text{Int} \\ \hline [\hat{\alpha}] \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow \text{Int} \\ \hline \forall a. [a] \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow \text{Int} \end{array}$$

Unification → Subtyping → Inference

The Algorithm

?

$$\forall a. a \rightarrow a \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int}$$

.....

Unification → Subtyping → Inference

The Algorithm

?

$$\hat{a} \rightarrow \hat{a} \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int}$$

$$\forall a. a \rightarrow a \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int}$$

.....

Unification → Subtyping → Inference

The Algorithm

$$\frac{\begin{array}{c} ? \\ \hline \hat{\alpha} \geq [\text{Bool}] \end{array}}{\hat{\alpha} \rightarrow \hat{\alpha} \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int}}$$
$$\frac{\begin{array}{c} ? \\ \hline \hat{\alpha} \geq [\text{Int}] \end{array} \quad \text{Int} \leq \text{Int}}{\hat{\alpha} \rightarrow \hat{\alpha} \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int}}$$
$$\frac{}{\forall a. a \rightarrow a \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int}}$$

Unification → Subtyping → Inference

The Algorithm

$$\hat{\alpha} = [\text{Bool}] \vee [\text{Int}]$$

$$\hat{\alpha} \geq [\text{Bool}] \quad \hat{\alpha} \geq [\text{Int}] \quad \text{Int} \leq \text{Int}$$

$$\hat{\alpha} \rightarrow \hat{\alpha} \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int}$$

$$\forall a. a \rightarrow a \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int}$$



The Algorithm

$$\begin{array}{c} \text{[} \underline{\text{ - }} \text{]} \\ \swarrow \qquad \searrow \\ \hat{\alpha} = [\text{Bool}] \vee [\text{Int}] \\ \hline \hat{\alpha} \geq [\text{Bool}] \quad \hat{\alpha} \geq [\text{Int}] \quad \text{Int} \leq \text{Int} \\ \hline \hat{\alpha} \rightarrow \hat{\alpha} \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int} \\ \hline \forall a. a \rightarrow a \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int} \end{array}$$

Anti-unification $\longrightarrow P \vee Q$ \longrightarrow Subtyping \rightarrow Inference
Unification \longrightarrow

The Algorithm

$$\frac{\exists \beta . [\beta]}{\hat{\alpha} = [\text{Bool}] \vee [\text{Int}]}$$
$$\frac{\hat{\alpha} \geq [\text{Bool}] \quad \hat{\alpha} \geq [\text{Int}] \quad \text{Int} \leq \text{Int}}{\hat{\alpha} \rightarrow \hat{\alpha} \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int}}$$
$$\frac{}{\forall a . a \rightarrow a \rightarrow \text{Int} \leq [\text{Bool}] \rightarrow [\text{Int}] \rightarrow \text{Int}}$$

Anti-unification $\longrightarrow P \vee Q$

Unification \longrightarrow Subtyping \rightarrow Inference

The Algorithm

Does it work? NO!

The Algorithm

Does it work? NO! Unless we restrict the system:

Polarization

Does it work? NO! Unless we restrict the system:

Positive: $a^+ [P]$ $\exists a^+. P \downarrow N$

Negative: $a^- P \rightarrow N$ $\forall a^-. N \uparrow P$

Polarization

Does it work? NO! Unless we restrict the system:

Positive: $a^+ [P] \quad \exists a^+. P \downarrow N$

Negative: $a^- P \rightarrow N \quad \forall a^-. N \uparrow P$

Shifts are invariant:

$$\frac{P \approx Q}{\uparrow P \leq \uparrow Q} \qquad \frac{N \approx M}{\downarrow N \geq \downarrow M}$$

\hat{a}^\pm occur only on one side \Rightarrow unification = matching

a^- are “protected” by $\downarrow \quad \Rightarrow \quad N \wedge M$ is not required

Summary

- Inference and \leq for a large segment of System F with \exists
- Anti-unification and unification are applied
- Polarization reveals dualities:

Data	Computations
\exists -types	\forall -types
Supertypes	Subtypes
Checking	Inference
Anti-Unification	Unification

Plans

- Submit to POPL'24
- Categorization
- Dependent Types
- Mechanization