

HM(X): Type Inference with Constraint Types ^[1]

Marius Weidner

Chair of Programming Languages, University of Freiburg

July 17, 2023

Outline

- 1 HM: Polymorphic Lambda Calculus with Full Type Inference
- 2 Constraint Systems: Formulate Constraints on Types
- 3 $\text{HM}(X)$: HM Parametrized over Constraint Systems X
- 4 $\text{HM}(\mathcal{R})$: Instantiating $\text{HM}(X)$ with Polymorphic Records
- 5 Meta Theory & Conclusion: Properties of $\text{HM}(X)$ and beyond $\text{HM}(X)$

- Let Polymorphism
- Full Type Inference
- Principle Type Property
 - ▶ most general type is inferred

Example: Polymorphic Identity Function

```
let id = λx. x in      :  $\forall \alpha. \alpha \rightarrow \alpha$   
id 42                  : Int  
id "Hello World"      : Str
```

Example: Polymorphic List Constructors

```
let nil = .. :  $\forall \alpha. \alpha \rightarrow \text{List } \alpha$   
let cons = .. :  $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \text{List } \alpha \rightarrow \text{List } \alpha$ 
```

Hindley Milner Syntax

$$\begin{aligned} e &::= x \\ &| \lambda x. e \\ &| e \ e \\ &| \mathbf{let} \ x = e \ \mathbf{in} \ e \end{aligned}$$
$$\begin{aligned} \tau &::= \alpha \\ &| \tau \rightarrow \tau \end{aligned}$$
$$\begin{aligned} \sigma &::= \tau \\ &| \forall \alpha. \sigma \end{aligned}$$

Hindley Milner Typing

$$\frac{\Gamma \vdash e : \sigma \quad \Gamma, x : \sigma \vdash e' : \tau'}{\Gamma \vdash \mathbf{let} \ x = e \ \mathbf{in} \ e' : \tau'} \text{ Let}$$

$$\frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{free}(\Gamma)}{\Gamma \vdash e : \forall \alpha. \sigma} \forall \text{ Intro}$$

Example: Instance Relation

$$\forall \alpha. \alpha \rightarrow \alpha \quad \sqsubseteq \quad \text{Int} \rightarrow \text{Int}$$

$$\frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma} \forall \text{ Elim}$$

Outlook: A Program with Constraint Types

Example: Record Subtyping Constraints

```
let max = λr1. λr2.                               : ∀α.(α ≤ {key:Int})∀β.(β ≤ {key:Int}) ⇒  
  r1.key ≥ r2.key in                                α → β → Bool  
max {key = 17, foo = "bar"}                          : Bool  
  {key = 42, bar = "foo"}
```

Constraint Syntax

$$\begin{aligned} C &::= \perp \\ &| \top \\ &| C \wedge C \\ &| \exists \alpha. C \\ &| \tau = \tau \\ &| \dots \end{aligned}$$

Constraint Conditions

Entailment

$C \vdash D$ iff C implies D

$C = D$ iff $C \vdash D \wedge D \vdash C$

$\vdash C$ iff $\top \vdash C$

Existential Quantification

$C \vdash \exists \alpha. C$

$C \vdash D$ implies $\exists \alpha. C \vdash \exists \alpha. D$

$\exists \alpha. (C \wedge \exists \alpha. D) = \exists \alpha. C \wedge \exists \alpha. D$

$\exists \alpha. \exists \beta. C = \exists \beta. \exists \alpha. C$

Equality

$\vdash (\alpha = \alpha)$

$(\alpha = \beta) \vdash (\beta = \alpha)$

$(\alpha = \beta) \wedge (\beta = \gamma) \vdash (\alpha = \gamma)$

Substitution

$(\tau = \tau') \vdash (T[\tau] = T[\tau'])$

$[\tau/\alpha]C = \exists \alpha. C \wedge (\alpha = \tau)$ for α not free in τ

Satisfiability

$sat(C)$ iff $\vdash \exists \vec{\alpha}.C$ for $\vec{\alpha}$ free in C

Constraints in Solved Form

$C \in \mathcal{S}$ implies $sat(C)$

$C \in \mathcal{S}$ and $C \vdash (\tau = \tau')$ implies $\vdash (\tau = \tau')$

$C \in \mathcal{S}$ implies $\exists \alpha.C \in \mathcal{S}$

HM(X) Syntax

$e ::= x$

| $\lambda x.e$

| $e\ e$

| **let** $x = e$ **in** e

$\tau ::= \alpha$

| $\tau \rightarrow \tau$

$\sigma ::= \tau$

| $\forall \vec{\alpha}. C \Rightarrow \tau$ for $C \in \mathcal{S}$

HM(X) Typing

$$\frac{C, \Gamma \vdash e : \sigma \quad C, (\Gamma, x : \sigma) \vdash e' : \tau'}{C, \Gamma \vdash \mathbf{let} \ x = e \ \mathbf{in} \ e' : \tau'} \text{ Let}$$

$$\frac{C \wedge D, \Gamma \vdash e : \tau \quad \vec{\alpha} \notin \text{free}(\Gamma) \cup \text{free}(C)}{C \wedge \exists \vec{\alpha}. D, \Gamma \vdash e : \forall \vec{\alpha}. D \Rightarrow \tau} \forall \text{ Intro}$$

$$\frac{C, \Gamma \vdash e : \tau \quad \vdash (\tau = \tau')}{C, \Gamma \vdash e : \tau'} \text{ Eq}$$

$$\frac{C, \Gamma \vdash e : \forall \vec{\alpha}. D \Rightarrow \tau' \quad C \vdash^e [\vec{\tau}/\vec{\alpha}] D}{C, \Gamma \vdash e : [\vec{\tau}/\vec{\alpha}] \tau'} \forall \text{ Elim}$$

HM(\mathcal{R}) Syntax

$e ::= \dots$

| $\{l_i = e_i\} \text{ for } i \in \mathbb{N}$

| $e.l$

$\tau ::= \dots$

| $\{l_i : \tau_i\} \text{ for } i \in \mathbb{N}$

$c ::= \dots$

| $\tau \leq \{l : \tau'\}$

HM(\mathcal{R}) Constraint Conditions

Subtyping

$$\begin{aligned} &\vdash \{l_1 : \tau_1, \dots, l_n : \tau_n\} \leq \{l_i : \tau_i\} \\ &\tau \leq \{l : \tau_1\} \wedge \tau \leq \{l : \tau_2\} \vdash \tau_1 = \tau_2 \\ &\{\dots, l : \tau_1, \dots\} \leq \{l : \tau_2\} \vdash \tau_1 = \tau_2 \\ &\{l_1 : \tau_1, \dots, l_n : \tau_n\} \leq \{l : \tau\} \vdash \perp \end{aligned}$$

Restrict Recursion

$$\begin{aligned} &\exists \alpha. \alpha \leq \{l : \tau\} = \top \text{ for } \alpha \text{ not free in } \tau \\ &\exists \alpha. \alpha \leq \{l : \tau\} = \perp \text{ for } \alpha \text{ free in } \tau \end{aligned}$$

Ordering

$$\begin{aligned} &\vdash \{l_i : \tau_i\} = \{l_{\pi(i)} : \tau_{\pi(i)}\} \\ &\text{where } \pi \text{ is permutation} \end{aligned}$$

Solved Form

$$\begin{aligned} &\tau \leq \{l : \tau\} \in \mathcal{S} \text{ implies } \tau = \alpha \\ &\text{where } \alpha \leq \{l : \tau\} \text{ non-recursive} \end{aligned}$$

HM(\mathcal{R}) Advanced Example

Example: Nested Record Subtyping Constraints

```
let unwrap =  $\lambda r.$                                 :  $\forall \alpha. \exists \beta (\alpha \leq \{\text{outer} : \beta\} \wedge \beta \leq \{\text{inner} : \text{Int}\}) \Rightarrow$   
  r.outer.inner in                                   $\alpha \rightarrow \text{Int}$   
unwrap {                                           :  $\text{Int}$   
  outer = {  
    inner = 42,  
    bar = "foo"  
  },  
  foo = "bar"  
}
```

Meta Theoretical Properties & Outlook

- Full Type Inference: $\text{HM}(X)$ comes with a type inference algorithm that
 - ▶ solves typing problems by translating them to constraint problems
 - ▶ preserves the *principal type property*, if X fulfills the *principal constraint property*
- Type Soundness: $\text{HM}(X)$ is sound, i.e. all valid typing judgements $C, \Gamma \vdash e : \sigma$ where $\text{sound}(C)$ imply $e \hookrightarrow e' \vee \text{val}(e)$ (*progress*) and $e \hookrightarrow e' \rightarrow C, \Gamma \vdash e' : \sigma$ (*subject reduction*) [3]
- $\text{SHM}(X)$: extension with subtyping constraint $\tau \leq \tau'$, small extension to type inference algorithm required

References

- [1] Martin Odersky, Martin Sulzmann, and Martin Wehr. “Type Inference with Constrained Types”. In: *TAPOS 5* (Jan. 1999), pp. 35–55. DOI: [10.1002/\(SICI\)1096-9942\(199901/03\)5:1<35::AID-TAP04>3.0.CO;2-4](https://doi.org/10.1002/(SICI)1096-9942(199901/03)5:1<35::AID-TAP04>3.0.CO;2-4).
- [2] Benjamin C. Pierce. *Advanced Topics in Types and Programming Languages*. The MIT Press, 2004. ISBN: 0262162288.
- [3] Christian Skalka and François Pottier. “Syntactic Type Soundness for HM(X)”. In: *Electronic Notes in Theoretical Computer Science 75* (2003). TIP'02, International Workshop in Types in Programming, pp. 61–74. ISSN: 1571-0661. DOI: [https://doi.org/10.1016/S1571-0661\(04\)80779-5](https://doi.org/10.1016/S1571-0661(04)80779-5). URL: <https://www.sciencedirect.com/science/article/pii/S1571066104807795>.