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

Marius Weidner

Chair of Programming Languages, University of Freiburg

August 5, 2023

Outline

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

Hindley Milner Basics

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

Example: Polymorphic Identity Function

```
let id = λx. x in          : ∀α.α → α
id 42                      : Int
id "Hello World"           : Str
```

Example: Polymorphic List Constructors

```
let nil = ..    : ∀α. List α
let cons = ..   : ∀α.α → List α → List α
```

Hindley Milner Syntax

$$
\begin{aligned}
e \;::=\;& x \\
\mid\;& \lambda x.e \\
\mid\;& e\ e \\
\mid\;& \textbf{let}\ x = e\ \textbf{in}\ e
\end{aligned}
$$

$$
\begin{aligned}
\tau \;::=\;& \alpha \\
\mid\;& \tau \to \tau
\end{aligned}
$$

$$
\begin{aligned}
\sigma \;::=\;& \tau \\
\mid\;& \forall \vec{\alpha}.\tau
\end{aligned}
$$

Hindley Milner Typing

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

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

$$\frac{\Gamma \vdash e : \forall \vec{\alpha}.\tau}{\Gamma \vdash e : [\vec{\tau}/\vec{\alpha}]\tau} \forall \text{ Elim}$$

A Program with Constraint Types

Example: Record Subtyping Constraints

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

Constraint Syntax

$$
\begin{aligned}
C \;::=\; & \bot \\
\mid\; & \top \\
\mid\; & C \wedge C \\
\mid\; & \exists \alpha . C \\
\mid\; & \tau = \tau \\
\mid\; & \cdots
\end{aligned}
$$

Constraint Conditions

Entailment

$$C \vdash D \text{ iff } C \text{ implies } D$$
$$C = D \text{ iff } C \vdash D \wedge D \vdash C$$
$$\vdash C \text{ iff } \top \vdash C$$

Substitution

$$[\tau/\alpha]C = \exists\alpha.C \wedge (\alpha = \tau) \text{ for } \alpha \text{ not free in } \tau$$

Existential Quantification

$$C \vdash \exists\alpha.C$$
$$C \vdash D \text{ implies } \exists\alpha.C \vdash \exists\alpha.D$$
$$\exists\alpha.(C \wedge \exists\alpha.D) = \exists\alpha.C \wedge \exists\alpha.D$$
$$\exists\alpha.\exists\beta.C = \exists\beta.\exists\alpha.C$$

Solved Form

Satisfiability

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

Constraints in Solved Form

$$C \in \mathcal{S} \text{ implies } sat(C)$$
$$C \in \mathcal{S} \text{ and } C \vdash (\tau = \tau') \text{ implies } \vdash (\tau = \tau')$$
$$C \in \mathcal{S} \text{ implies } \exists \alpha.C \in \mathcal{S}$$

HM(X) Syntax

$$\begin{aligned} e \; &::= x \\ &\mid \lambda x.e \\ &\mid e \; e \\ &\mid \textbf{let } x = e \textbf{ in } e \end{aligned}$$

$$\begin{aligned} \tau \; &::= \alpha \\ &\mid \tau \to \tau \end{aligned}$$

$$\begin{aligned} \sigma \; &::= \tau \\ &\mid \forall \vec{\alpha}.C \Rightarrow \tau \text{ for } C \in \mathcal{S} \end{aligned}$$

HM(X) Typing

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

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

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

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

HM($\mathcal{R}$) Syntax

$$
\begin{aligned}
e \;\; ::= \;\; & \ldots \\
& \mid \{l_i = e_i\} \text{ for } i \in \mathbb{N} \\
& \mid e.l
\end{aligned}
$$

$$
\begin{aligned}
\tau \;\; ::= \;\; & \ldots \\
& \mid \{l_i : \tau_i\} \text{ for } i \in \mathbb{N}
\end{aligned}
$$

$$
\begin{aligned}
C \;\; ::= \;\; & \ldots \\
& \mid \tau \leq \{l : \tau'\}
\end{aligned}
$$

HM($\mathcal{R}$) Constraint Conditions

Subtyping

$$\vdash \{l_1 : \tau_1, .., l_n : \tau_n\} \leq \{l_i : \tau_i\}$$
$$\tau \leq \{l : \tau_1\} \wedge \tau \leq \{l : \tau_2\} \vdash \tau_1 = \tau_2$$
$$\{.., l : \tau_1, ..\} \leq \{l : \tau_2\} \vdash \tau_1 = \tau_2$$
$$\{l_1 : \tau_1, .., l_n : \tau_n\} \leq \{l : \tau\} \vdash \perp$$

Restrict Recusion

$\exists \alpha . \alpha \leq \{l : \tau\} = \top$ for α not free in τ

$\exists \alpha . \alpha \leq \{l : \tau\} = \perp$ for α free in τ

Ordering

$$\vdash \{l_i : \tau_i\} = \{l_{\pi(i)} : \tau_{\pi(i)}\}$$

where π is a permutation

Solved Form

$\alpha \leq \{l : \tau\} \in \mathcal{S}$

where all $\alpha \leq \{l : \tau\}$ non-recursive

HM($\mathcal{R}$) Example

Example: Nested Record Subtyping Constraints

```
let unwrap = λr.                : ∀α.∃β(α ≤{outer:β}∧β ≤{inner:Int}) ⇒
  r.outer.inner in                α → Int
unwrap {                        : Int
  outer = {
    inner = 42,
    bar = "foo"
  },
  foo = "bar"
}
```

Meta Theoretical Properties & Outlook

- Full Type Inference: 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: HM(X) is sound, i.e. all valid typing judgements $C, \Gamma \vdash e : \sigma$ where $sound(C)$ imply $e \hookrightarrow e' \vee val(e)$ (*progress*) and $e \hookrightarrow e' \rightarrow C, \Gamma \vdash e' : \sigma$ (*subject reduction*) [3]
- 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-TAPO4>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. URL: https://www.sciencedirect.com/science/article/pii/S1571066104807795.