



# Hindley-Milner Type Inference

Rob Sison  
UNSW  
Term 3 2024

# Implicitly Typed MinHS

Explicitly typed languages are awkward to use<sup>1</sup>. Ideally, we'd like the compiler to determine the types for us.

## Example

What is the type of this function?

**recfun**  $f\ x = \text{fst } x + 1$

We want the compiler to infer the **most general** type.

---

<sup>1</sup>See Java

# Implicitly Typed MinHS

Start with our polymorphic MinHS, then:

- **remove** type signatures from **recfun**, **let**, etc.
- **remove** explicit **type** abstractions, and type applications (the **@** operator).
- **keep**  $\forall$ -quantified types.

# Implicitly Typed MinHS

Start with our polymorphic MinHS, then:

- **remove** type signatures from **recfun**, **let**, etc.
- **remove** explicit **type** abstractions, and type applications (the **@** operator).
- **keep**  $\forall$ -quantified types.
- **remove** recursive types, as we can't infer types for them.

# Typing Rules

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \text{VAR}$$

# Typing Rules

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \text{VAR}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 \ e_2 : \tau_2} \text{APP}$$

# Typing Rules

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \text{VAR}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 \ e_2 : \tau_2} \text{APP}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\text{Pair } e_1 \ e_2) : \tau_1 \times \tau_2} \text{CONJ}_I$$

# Typing Rules

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \text{VAR}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 \ e_2 : \tau_2} \text{APP}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\text{Pair } e_1 \ e_2) : \tau_1 \times \tau_2} \text{CONJ}_I$$

$$\frac{\Gamma \vdash e_1 : \text{Bool} \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash (\text{If } e_1 \ e_2 \ e_3) : \tau} \text{I}_F$$



# Primitive Operators

For convenience, we treat prim ops as **functions**, and place their types in the environment.

$$(+) : \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}, \Gamma \vdash (\text{App} (\text{App} (+) (\text{Num } 2)) (\text{Num } 1)) : \text{Int}$$

## Functions

$$\frac{x : \tau_1, f : \tau_1 \rightarrow \tau_2, \Gamma \vdash e : \tau_2}{\Gamma \vdash (\text{Recfun } (f.x. e)) : \tau_1 \rightarrow \tau_2} \text{FUNC}$$

## Sum Types

$$\frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash \text{InL } e : \tau_1 + \tau_2} \text{DISJ}_{I1}$$

$$\frac{\Gamma \vdash e : \tau_2}{\Gamma \vdash \text{InR } e : \tau_1 + \tau_2} \text{DISJ}_{I2}$$

Note that we allow the other side of the sum to be **any** type.



# Polymorphism

If we have a polymorphic type, we can instantiate it to any type:

$$\frac{\Gamma \vdash e : \forall a. \tau}{\Gamma \vdash e : \tau[a := \rho]} \text{ALLE}$$

We can quantify over any variable that has not already been used.

$$\frac{\Gamma \vdash e : \tau \quad a \notin TV(\Gamma)}{\Gamma \vdash e : \forall a. \tau} \text{ALLI}$$

(Where  $TV(\Gamma)$  here is all type variables occurring free in the types of variables in  $\Gamma$ )

# The Goal

We want *an algorithm* for type inference:

- With a clear **input** and **output**
- Which *terminates*.
- Which is fully *deterministic*.

# Typing Rules

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\text{Pair } e_1 \ e_2) : \tau_1 \times \tau_2}$$

Can we use the existing typing rules as our algorithm?

`infer :: Context → Expr → Type`

# Typing Rules

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\text{Pair } e_1 \ e_2) : \tau_1 \times \tau_2}$$

Can we use the existing typing rules as our algorithm?

`infer :: Context → Expr → Type`

This approach can work for monomorphic types, but not polymorphic ones. Why not?



# First Problem

$$\frac{\Gamma \vdash e : \forall a. \tau}{\Gamma \vdash e : \tau[a := \rho]} \text{ALLE}$$

The rule to add a  $\forall$ -quantifier can **always** be applied:

$$\frac{\begin{array}{c} \vdots \\ \hline \Gamma \vdash (\text{Num } 5) : \forall a. \forall b. \text{Int} \end{array}}{\Gamma \vdash (\text{Num } 5) : \forall b. \text{Int}} \text{ALLE}$$

$$\frac{\Gamma \vdash (\text{Num } 5) : \forall b. \text{Int}}{\Gamma \vdash (\text{Num } 5) : \text{Int}} \text{ALLE}$$

Read as an algorithm, the rules are **non-deterministic** – there are many possible rules for a given input. A depth-first search strategy may end up attempting infinite derivations.

## Another Problem

$$\frac{\Gamma \vdash e : \forall a. \tau}{\Gamma \vdash e : \tau[a := \rho]} \text{ALLE}$$

The above rule can be applied at **any time** to a polymorphic type, even if it would break later typing derivations:

$$\frac{\Gamma \vdash \text{fst} : \forall a. \forall b. (a \times b) \rightarrow a \quad \dots}{\Gamma \vdash \text{fst} : (\text{Bool} \times \text{Bool}) \rightarrow \text{Bool} \quad \Gamma \vdash (\text{Pair } 1 \text{ True}) : (\text{Int} \times \text{Bool})} \Gamma \vdash (\text{Apply fst (Pair 1 True)}) : ???$$

We need a way to capture the dependency between the parts of this derivation!

## Yet Another Problem

The rule for **recfun** mentions  $\tau_2$  in both input and output positions.

$$\frac{x : \tau_1, f : \tau_1 \rightarrow \tau_2, \Gamma \vdash e : \tau_2}{\Gamma \vdash (\text{Recfun } (f.x. e)) : \tau_1 \rightarrow \tau_2} \text{FUNC}$$

In order to infer  $\tau_2$  we must provide a context that includes  $\tau_2$  — this is circular. Any guess we make for  $\tau_2$  could be wrong.

## Solution: Unknowns and Unification

We allow types to include *unknowns*, also known as *unification variables* or *schematic variables*. We will call them *flexible* (type) variables. These are placeholders for types that we haven't worked out yet. We shall use  $\alpha, \beta$  etc. for names of type variables,  $\alpha^F, \beta^F$  for flexible type variables, and  $\alpha^R, \beta^R$  for rigid type variables (bound by  $\forall$ -quantifiers).

### Example

$(\text{Int} \times \alpha^F) \rightarrow \beta^F$  is the type of a function from tuples where the left side is `Int`, but no other details of the type have been determined yet.

As we encounter situations where two types should be equal, we *unify* the two types to determine what the unknown variables should be.

# Unification

Our rules for unification will be specified by the judgement:

$$\Gamma_1 \vdash \tau_1 \sim \tau_2 \implies \Gamma_2$$

which are defined such that:

- 1  $\Gamma_1$  and  $\Gamma_2$  contain the same type variables;
- 2  $\Gamma_2$  is **more informative** than  $\Gamma_1$  in the sense that declared type variables have been given definitions in order for  $\tau_1 \sim \tau_2$  to hold:  $\Gamma_1 \sqsubseteq \Gamma_2$
- 3 The information increase is **minimal** (most general) in the sense that it makes the least commitment in order to solve the equation: any other solution  $\Gamma_1 \sqsubseteq \Gamma'$  factors through  $\Gamma_1 \sqsubseteq \Gamma_2$ .

## Back to Type Inference

To keep track of the solutions to unification problems in context, we will **decompose** the typing judgement to allow for an additional output — an updated typing context which represents the *minimal information increase* over the input context (obtained via unification rules!) in order to infer the type of the expression.

**Inputs** Expression, Context

**Outputs** Type, Context

We will write this as  $\Gamma \vdash e \Rightarrow \tau \vdash \Gamma$ , to make clear what are the inputs and outputs.

## One More Concern: Generalisation

$$\frac{\Gamma \vdash e : \tau \quad a \notin TV(\Gamma)}{\Gamma \vdash e : \forall a. \tau} \text{ALL}_I$$

We can generalise a type to a polymorphic type by introducing a  $\forall$  at any point. We want to restrict this to only occur in a *syntax-directed* way.

Consider this example:

**let**  $f = (\text{recfun } f \ x = (x, x))$  **in**  $(\text{fst } (f \ 4), \text{fst } (f \ \text{True}))$

Where should generalisation happen?

## Solution: Let-generalisation

To make type inference tractable, we will generalise only in **let** expressions.



## Solution: Let-generalisation

To make type inference tractable, we will generalise only in **let** expressions.

This means that **let** expressions are now not just sugar for a function application. They actually play a vital role, as the place where generalisation happens.

# The New Type Inference Judgement

$$\Gamma_1 \vdash e \Rightarrow \tau \dashv \Gamma_2$$

**Purpose** Keep track of the solutions to unification problems in context;

Output context represents *minimal information increase* over the input context in order to infer the type of the expression.

**Inputs** Expression, Context

**Outputs** Type, Context



## Example Part 1

PRIM

$$\Gamma \vdash \text{fst} \Rightarrow ? \vdash ?$$

## Example Part 1

$$\frac{\text{PRIM} \quad \text{primOpType}(\text{fst}) = \forall \alpha \beta. (\alpha^{\text{R}} \times \beta^{\text{R}}) \rightarrow \alpha^{\text{R}}}{\Gamma \vdash \text{fst} \implies ? \vdash ?}$$

## Example Part 1

$$\frac{\text{PRIM} \quad \text{primOpType}(\text{fst}) = \forall \alpha \beta. (\alpha^{\text{R}} \times \beta^{\text{R}}) \rightarrow \alpha^{\text{R}}}{\Gamma \vdash \text{fst} \implies (\alpha^{\text{F}} \times \beta^{\text{F}}) \rightarrow \alpha^{\text{F}} \dashv \Gamma \cdot \alpha \cdot \beta}$$

NB: Must introduce *fresh* names for the flexible type variables ( $\alpha$ ,  $\beta$  reused here just for convenience!)

## Example Part 2

$$\frac{\text{PRIM} \quad \text{primOpType}(\text{fst}) = \forall \alpha \beta. (\alpha^{\text{R}} \times \beta^{\text{R}}) \rightarrow \alpha^{\text{R}}}{\Gamma \vdash \text{fst} \implies (\alpha^{\text{F}} \times \beta^{\text{F}}) \rightarrow \alpha^{\text{F}} \dashv \Gamma \cdot \alpha \cdot \beta}$$

$$\Gamma_2 = \Gamma \cdot \alpha \cdot \beta$$

$$\text{PROD} \quad \frac{\begin{array}{l} \Gamma_2 \vdash 1 \Rightarrow \text{Int} \dashv \Gamma_2 \quad \Gamma_2 \vdash \text{True} \Rightarrow \text{Bool} \dashv \Gamma_2 \\ \Gamma_2 \cdot \rho \vdash \rho^F \sim \text{Int} \times \text{Bool} \Rightarrow \Gamma_2 \cdot \rho := \text{Int} \times \text{Bool} \end{array}}{\Gamma_2 \vdash (\text{Pair } 1 \text{ True}) \Rightarrow \rho^F \dashv \Gamma_2 \cdot \rho := \text{Int} \times \text{Bool}}$$

## Example Part 3

$$\Gamma_3 = \Gamma_2 \cdot \rho := \text{Int} \times \text{Bool}$$

APP

$$\frac{\Gamma \vdash \text{fst} \Rightarrow (\alpha^F \times \beta^F) \rightarrow \alpha^F \vdash \Gamma_2 \quad \Gamma_2 \vdash (\text{Pair 1 True}) \Rightarrow \rho^F \vdash \Gamma_3 \quad \Gamma_3 \cdot \omega \vdash (\alpha^F \times \beta^F) \rightarrow \alpha^F \sim \rho^F \rightarrow \omega^F \Rightarrow ?}{\Gamma \vdash \text{Apply fst (Pair 1 True)} \Rightarrow ? \vdash ?}$$



## Example Part 3

$$\Gamma_3 = \Gamma_2 \cdot \rho := \text{Int} \times \text{Bool}$$

$$\frac{\Gamma_3 \cdot \omega \vdash \alpha^F \times \beta^F \sim \rho^F \implies ? \quad ? \vdash \alpha^F \sim \omega^F \implies ?}{\Gamma_3 \cdot \omega \vdash (\alpha^F \times \beta^F) \rightarrow \alpha^F \sim \rho^F \rightarrow \omega^F \implies ?}$$

## Example Part 3A

$$\Gamma_2 = \Gamma \cdot \alpha \cdot \beta$$

$$\Gamma_3 = \Gamma_2 \cdot \rho := \text{Int} \times \text{Bool}$$

$$\Gamma_4 = \Gamma_3 \cdot \omega$$

$$\frac{\frac{\frac{\Gamma_2 \vdash \text{Int} \sim \alpha^F \Rightarrow ? \quad ? \vdash \text{Bool} \sim \beta^F \Rightarrow ?}{\Gamma_2 \cdot [] \vdash \text{Int} \times \text{Bool} \sim \alpha^F \times \beta^F \Rightarrow ?}}{\Gamma_3 \mid [] \vdash \rho^F \sim \alpha^F \times \beta^F \Rightarrow ?} \text{SUBST}$$

$$\frac{\Gamma_3 \mid [] \vdash \rho^F \sim \alpha^F \times \beta^F \Rightarrow ?}{\Gamma_4 \mid [] \vdash \rho^F \sim \alpha^F \times \beta^F \Rightarrow ?} \text{SKIP-TY}$$

$$\frac{\Gamma_4 \mid [] \vdash \rho^F \sim \alpha^F \times \beta^F \Rightarrow ?}{\Gamma_4 \vdash \alpha^F \times \beta^F \sim \rho^F \Rightarrow ?} \text{INST}$$



# Example Part 3A

$$\Gamma_2 = \Gamma \cdot \alpha \cdot \beta$$

$$\Gamma_5 = \Gamma \cdot \alpha := \text{Int} \cdot \beta$$

$$\Gamma_3 = \Gamma_2 \cdot \rho := \text{Int} \times \text{Bool}$$

$$\Gamma_4 = \Gamma_3 \cdot \omega$$

$$\Gamma_2 \vdash \text{Int} \sim \alpha^F \implies \Gamma_5 \quad ? \vdash \text{Bool} \sim \beta^F \implies ?$$

$$\Gamma_2 \vdash \text{Int} \times \text{Bool} \sim \alpha^F \times \beta^F \implies ?$$

$$\Gamma_3 \mid [] \vdash \rho^F \sim \alpha^F \times \beta^F \implies ?$$

$$\Gamma_4 \mid [] \vdash \rho^F \sim \alpha^F \times \beta^F \implies ?$$

$$\Gamma_4 \vdash \alpha^F \times \beta^F \sim \rho^F \implies ?$$

## Example Part 3A

$$\Gamma_2 = \Gamma \cdot \alpha \cdot \beta$$

$$\Gamma_5 = \Gamma \cdot \alpha := \text{Int} \cdot \beta$$

$$\Gamma_3 = \Gamma_2 \cdot \rho := \text{Int} \times \text{Bool} \quad \Gamma_6 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool}$$

$$\Gamma_4 = \Gamma_3 \cdot \omega$$

$$\frac{\Gamma_2 \vdash \text{Int} \sim \alpha^F \Rightarrow \Gamma_5 \quad \Gamma_5 \vdash \text{Bool} \sim \beta^F \Rightarrow \Gamma_6}{\Gamma_2 \vdash \text{Int} \times \text{Bool} \sim \alpha^F \times \beta^F \Rightarrow \Gamma_6}$$


---


$$\Gamma_3 \mid [] \vdash \rho^F \sim \alpha^F \times \beta^F \Rightarrow ?$$


---


$$\Gamma_4 \mid [] \vdash \rho^F \sim \alpha^F \times \beta^F \Rightarrow ?$$


---


$$\Gamma_4 \vdash \alpha^F \times \beta^F \sim \rho^F \Rightarrow ?$$

## Example Part 3A

$$\Gamma_2 = \Gamma \cdot \alpha \cdot \beta$$

$$\Gamma_5 = \Gamma \cdot \alpha := \text{Int} \cdot \beta$$

$$\Gamma_3 = \Gamma_2 \cdot \rho := \text{Int} \times \text{Bool}$$

$$\Gamma_6 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool}$$

$$\Gamma_4 = \Gamma_3 \cdot \omega$$

$$\Gamma_7 = \Gamma_6 \cdot \rho := \text{Int} \times \text{Bool}$$

$$\frac{\Gamma_2 \vdash \text{Int} \sim \alpha^F \Rightarrow \Gamma_5 \quad \Gamma_5 \vdash \text{Bool} \sim \beta^F \Rightarrow \Gamma_6}{\Gamma_2 \vdash \text{Int} \times \text{Bool} \sim \alpha^F \times \beta^F \Rightarrow \Gamma_6}$$

$$\frac{\Gamma_3 \mid [] \vdash \rho^F \sim \alpha^F \times \beta^F \Rightarrow \Gamma_7}{\Gamma_4 \mid [] \vdash \rho^F \sim \alpha^F \times \beta^F \Rightarrow \Gamma_7 \cdot \omega}$$

$$\frac{\Gamma_4 \vdash \alpha^F \times \beta^F \sim \rho^F \Rightarrow \Gamma_7 \cdot \omega}{\Gamma_4 \vdash \alpha^F \times \beta^F \sim \rho^F \Rightarrow \Gamma_7 \cdot \omega}$$

## Example Part 3

$$\Gamma_4 = \Gamma_3 \cdot \omega$$

$$\Gamma_6 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool}$$

$$\Gamma_7 = \Gamma_6 \cdot \rho := \text{Int} \times \text{Bool}$$

$$\frac{\Gamma_4 \vdash \alpha^F \times \beta^F \sim \rho^F \implies \Gamma_7 \cdot \omega \quad ? \vdash \alpha^F \sim \omega^F \implies ?}{\Gamma_4 \vdash (\alpha^F \times \beta^F) \rightarrow \alpha^F \sim \rho^F \rightarrow \omega^F \implies ?}$$

## Example Part 3B

$$\Gamma_4 = \Gamma_3 \cdot \omega$$

$$\Gamma_6 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool}$$

$$\Gamma_7 = \Gamma_6 \cdot \rho := \text{Int} \times \text{Bool}$$

$$\frac{\Gamma_4 \vdash \alpha^F \times \beta^F \sim \rho^F \implies \Gamma_7 \cdot \omega \quad \Gamma_7 \cdot \omega \vdash \alpha^F \sim \omega^F \implies ?}{\Gamma_4 \vdash (\alpha^F \times \beta^F) \rightarrow \alpha^F \sim \rho^F \rightarrow \omega^F \implies ?}$$



## Example Part 3B

$$\Gamma_4 = \Gamma_3 \cdot \omega$$

$$\Gamma_6 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool}$$

$$\Gamma_7 = \Gamma_6 \cdot \rho := \text{Int} \times \text{Bool}$$

$$\Gamma_8 = \Gamma_7 \cdot \omega := \alpha^F$$

$$\frac{\Gamma_4 \vdash \alpha^F \times \beta^F \sim \rho^F \implies \Gamma_7 \cdot \omega \quad \Gamma_7 \cdot \omega \vdash \alpha^F \sim \omega^F \implies \Gamma_8}{\Gamma_4 \vdash (\alpha^F \times \beta^F) \rightarrow \alpha^F \sim \rho^F \rightarrow \omega^F \implies \Gamma_8}$$

## Example Part 3

$$\Gamma_4 = \Gamma_3 \cdot \omega$$

$$\Gamma_6 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool}$$

$$\Gamma_7 = \Gamma_6 \cdot \rho := \text{Int} \times \text{Bool}$$

$$\Gamma_8 = \Gamma_7 \cdot \omega := \alpha^F$$

$$\frac{\Gamma_4 \vdash \alpha^F \times \beta^F \sim \rho^F \implies \Gamma_7 \cdot \omega \quad \Gamma_7 \cdot \omega \vdash \alpha^F \sim \omega^F \implies \Gamma_8}{\Gamma_4 \vdash (\alpha^F \times \beta^F) \rightarrow \alpha^F \sim \rho^F \rightarrow \omega^F \implies \Gamma_8}$$

$$\Gamma_8 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool} \cdot \rho := \text{Int} \times \text{Bool} \cdot \omega := \alpha^F$$

# Example

$$\Gamma_2 = \Gamma \cdot \alpha \cdot \beta$$

$$\Gamma_3 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool} \cdot \rho := \text{Int} \times \text{Bool}$$

$$\Gamma_8 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool} \cdot \rho := \text{Int} \times \text{Bool} \cdot \omega := \alpha^F$$

APP

$$\frac{\begin{array}{l} \Gamma \vdash \text{fst} \Rightarrow (\alpha^F \times \beta^F) \rightarrow \alpha^F \dashv \Gamma_2 \quad \Gamma_2 \vdash (\text{Pair } 1 \text{ True}) \Rightarrow \rho^F \dashv \Gamma_3 \\ \Gamma_3 \cdot \omega \vdash (\alpha^F \times \beta^F) \rightarrow \alpha^F \sim \rho^F \rightarrow \omega^F \Rightarrow \Gamma_8 \end{array}}{\Gamma \vdash \text{Apply fst (Pair 1 True)} \Rightarrow ? \dashv ?}$$

## Example

$$\Gamma_2 = \Gamma \cdot \alpha \cdot \beta$$

$$\Gamma_3 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool} \cdot \rho := \text{Int} \times \text{Bool}$$

$$\Gamma_8 = \Gamma \cdot \alpha := \text{Int} \cdot \beta := \text{Bool} \cdot \rho := \text{Int} \times \text{Bool} \cdot \omega := \alpha^F$$

APP

$$\frac{\begin{array}{l} \Gamma \vdash \text{fst} \Rightarrow (\alpha^F \times \beta^F) \rightarrow \alpha^F \vdash \Gamma_2 \quad \Gamma_2 \vdash (\text{Pair } 1 \text{ True}) \Rightarrow \rho^F \vdash \Gamma_3 \\ \Gamma_3 \cdot \omega \vdash (\alpha^F \times \beta^F) \rightarrow \alpha^F \sim \rho^F \rightarrow \omega^F \Rightarrow \Gamma_8 \end{array}}{\Gamma \vdash \text{Apply fst (Pair 1 True)} \Rightarrow \omega^F \vdash \Gamma_8}$$

# Unification Example

**Demo:** See Notes on course website after the lecture.

## Summary

- We've started examining a variant of algorithm  $\mathcal{W}$  (originally due to Damas & Milner, variant thanks to Gundry) for type inference which tracks flexible variables and their instantiations using typing contexts;
- This algorithm is restricted to the Hindley-Milner subset of decidable polymorphic instantiations, and requires that polymorphism is top-level — polymorphic functions are not first class;
- The rest of the rules will be given in the specification for Assignment 2 — out soon.