Typed Lambda Calculus (3/3)

Eduardo Bonelli

"There may, indeed, be other applications of the system other than its use as a logic"

Alonzo Church, 1932

February 21, 2019

Contents

Type Inference

- Transform terms without type annotations or with partial type annotations into a typable term
- The missing type information must be inferred
- Benefits
  - programmer may omit some type declarations
  - does not affect performance: type inference takes place at compile time

Type Inference

- Specially useful in languages with polymorphism
- We start by restricting our study to $\lambda_V^{\mathbb{B},\to}$
- Even though $\lambda_V^{\mathbb{B},\to}$ is not polymorphic, it suffices to introduce many important notions in type inference
- We will address type inference for let-polymorphism later

The Type Inference Problem

Modified $\lambda_V^{\mathbb{B},\rightarrow}$ syntax – Removal of type annotations

$$
\begin{array}{rcl}
M & ::= & x \\
& | & \mathit{true} \mid \mathit{false} \mid \mathit{if}\ M\ \mathit{then}\ P\ \mathit{else}\ Q \\
& | & 0 \mid \mathit{succ}(M) \mid \mathit{pred}(M) \mid \mathit{iszero}(M) \\
& | & \lambda x : \sigma . M \mid M\, N \mid \\
& | & \mathit{fix}\ M
\end{array}
$$

The Type Inference Problem

Modified $\lambda_V^{\mathbb{B},\rightarrow}$ syntax – Removal of type annotations

$$
\begin{array}{lcl}
M & ::= & x \\
 & | & \mathit{true} \mid \mathit{false} \mid \mathit{if}\ M\ \mathit{then}\ P\ \mathit{else}\ Q \\
 & | & 0 \mid \mathit{succ}(M) \mid \mathit{pred}(M) \mid \mathit{iszero}(M) \\
 & | & \lambda x.M \mid M\,N \mid \\
 & | & \mathit{fix}\ M
\end{array}
$$

Erasing Function

The function $\mathit{Erase}(\cdot)$, given a term in $\lambda_V^{\mathbb{B},\to}$, erases all type annotations

$\mathit{Erase}(\lambda x : \mathbb{N}.\lambda f : \mathbb{N} \to \mathbb{N}.f\ x) = \lambda x.\lambda f.f\ x$

The Type Inference Problem - Definition

Given a term U without type annotations, find a standard term (i.e. one with type annotations) M s.t.

1. $\Gamma \triangleright M : \sigma$, for some Γ and σ; and
2. $Erase(M) = U$

Examples

- For $U = \lambda x.x + 5$ we take $M = \lambda x : \mathbb{N}.x + 5$ (note: no other possibility)

The Type Inference Problem - Definition

Given a term U without type annotations, find a standard term (i.e. one with type annotations) M s.t.

1. $\Gamma \triangleright M : \sigma$, for some Γ and σ; and
2. $Erase(M) = U$

Examples

- For $U = \lambda x.x + 5$ we take $M = \lambda x : \mathbb{N}.x + 5$ (note: no other possibility)
- For $U = \lambda x.\lambda f.f\ x$ we take $M_{\sigma,\tau} = \lambda x : \sigma.\lambda f : \sigma \to \tau.f\ x$ (there is a $M_{\sigma,\tau}$ for each σ, τ)

The Type Inference Problem - Definition

Given a term U without type annotations, find a standard term (i.e. one with type annotations) M s.t.

1. $\Gamma \triangleright M : \sigma$, for some Γ and σ; and
2. $Erase(M) = U$

Examples

- For $U = \lambda x.x + 5$ we take $M = \lambda x : \mathbb{N}.x + 5$ (note: no other possibility)
- For $U = \lambda x.\lambda f.f\ x$ we take $M_{\sigma,\tau} = \lambda x : \sigma.\lambda f : \sigma \to \tau.f\ x$ (there is a $M_{\sigma,\tau}$ for each σ, τ)
- For $U = \lambda x.\lambda f.f\ (f\ x)$ we take $M_\sigma = \lambda x : \sigma.\lambda f : \sigma \to \sigma.f\ (f\ x)$ (there is a M_σ for each σ)

The Type Inference Problem - Definition

Given a term U without type annotations, find a standard term (i.e. one with type annotations) M s.t.

1. $\Gamma \triangleright M : \sigma$, for some Γ and σ; and
2. $Erase(M) = U$

Examples

- For $U = \lambda x.x + 5$ we take $M = \lambda x : \mathbb{N}.x + 5$ (note: no other possibility)
- For $U = \lambda x.\lambda f.f\ x$ we take $M_{\sigma,\tau} = \lambda x : \sigma.\lambda f : \sigma \to \tau.f\ x$ (there is a $M_{\sigma,\tau}$ for each σ, τ)
- For $U = \lambda x.\lambda f.f\ (f\ x)$ we take $M_\sigma = \lambda x : \sigma.\lambda f : \sigma \to \sigma.f\ (f\ x)$ (there is a M_σ for each σ)
- For $U = xx$ there is no M with the desired property

The Type Checking Problem

type checking $\neq$ type inference

Type checking

Given a standard term M determine whether there exist Γ and σ s.t. $\Gamma \triangleright M : \sigma$ is derivable.

- Easier than type inference
- Simply follow the syntactic structure of M to reconstruct typing judgement
- Essentially equivalent to determining, given Γ and σ, if $\Gamma \triangleright M : \sigma$ is derivable.

Type Variables

- Given $\lambda x.\lambda f.f\,(f\,x)$, for each σ,
  $M_\sigma = \lambda x:\sigma.\lambda f:\sigma \to \sigma.f\,(f\,x)$ is a possible solution
- How may we provide a unique expression that encompasses all of them? Using type variables
  - All solutions may be represented with
    $$\lambda x : s.\lambda f : s \to s.f\,(f\,x)$$
  - "s" is a type variable that models an arbitrary type expression

Type Variables

- Type expressions of $\lambda_V^{\mathbb{B},\to}$ are extended with type variables $s, t, u, \ldots$

$$\sigma \quad ::= \quad s \,|\, \mathbb{N} \,|\, \mathbb{B} \,|\, \sigma \to \tau$$

Examples

- $s \to t$
- $\mathbb{N} \to \mathbb{N} \to t$
- $\mathbb{B} \to t$

Type Substitution

Function from type variables to type expressions. We use S and T for type substitutions.

- A substitution S may be applied to:
  1. a type expression σ ($S\sigma$)
  2. a term M (SM)
  3. typing context $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ ($S\Gamma$ defined below)

$$S\Gamma \stackrel{\text{def}}{=} \{x_1 : S\sigma_1, \ldots, x_n : S\sigma_n\}$$

Type Substitution - Additional Notions

- Supporting set $\{t \mid St \neq t\}$
  - Variables that S "affects"
- We use the notation $\{\sigma_1/t_1, \ldots, \sigma_n/t_n\}$ for substitutions
- The substitution whose supporting set is $\emptyset$ is the identity substitution (id)

Instance of a Typing Judgement

A typing judgement $\Gamma' \rhd M' : \sigma'$ is an instance of $\Gamma \rhd M : \sigma$ if there exists a type substitution S s.t.

$$\Gamma' \supseteq S\Gamma,\ M' = SM \text{ and } \sigma' = S\sigma$$

Property

If $\Gamma \rhd M : \sigma$ is derivable, then any of its instances too

Inference Function $\mathbb{W}(\cdot)$

Define a function $\mathbb{W}(\cdot)$ s.t. given a term U without type annotations it enjoys:

Correctness $\mathbb{W}(U) = \Gamma \triangleright M : \sigma$ implies

- $Erase(M) = U$ and
- $\Gamma \triangleright M : \sigma$ is derivable

Completeness If $\Gamma \triangleright M : \sigma$ is derivable and $Erase(M) = U$, then

- $\mathbb{W}(U)$ is successful and
- produces a judgement $\Gamma' \triangleright M' : \sigma'$ s.t. $\Gamma \triangleright M : \sigma$ is an instance of it (we say that $\mathbb{W}(\cdot)$ computes the principal type)

Unification

- The inference algorithm analyzes a term (without type annotations) through its subterms
- Once it obtains type information for them it
  1. Consistency Determines if the information obtained for each subterm is consistent
  2. Synthesis Synthesizes information about the original term via that of its subterms

Example

Consider $x\, y + x(y + 1)$

- From $x\, y$ we know $x :: s \to t$ and $y :: s$
- From $x\,(y + 1)$ we know $x :: \mathbb{N} \to u$ and $y :: \mathbb{N}$
- Since a variable can only have one type we must unify the type information
  - Type $s \to t$ must be unifiable with $\mathbb{N} \to u$ since both refer to x
  - Type s must be unifiable with $\mathbb{N}$ since both refer to y

Unification

- Is type $s \to t$ compatible or unifiable with $\mathbb{N} \to u$? Yes
  - It suffices to take substitution $S \stackrel{\text{def}}{=} \{\mathbb{N}/s, u/t\}$
  - And observer that $S(s \to t) = \mathbb{N} \to u = S(\mathbb{N} \to u)$
- Is type s compatible or unifiable with $\mathbb{N}$? Yes
  - The aforementioned substitutions is s.t. $Ss = S\mathbb{N}$

Unification: Process of determining whether there exists a substitution S s.t. two given type expressions σ, τ are unifiable (ie. $S\sigma = S\tau$)

- We will take a closer look at unification

Substitution Composition

Composition of S and T, denoted $S \circ T$, is the substitution that behaves as follows:

$$(S \circ T)(\sigma) = S(T\sigma)$$

Example

Let $S = \{u \to \mathbb{B}/t, \mathbb{N}/s\}$ and $T = \{v \times \mathbb{N}/u, \mathbb{N}/s\}$, then $T \circ S = \{(v \times \mathbb{N}) \to \mathbb{B}/t, v \times \mathbb{N}/u, \mathbb{N}/s\}$

- We say $S = T$ if they have the same support set and $St = Tt$ for all t in the support set of S
- $S \circ id = id \circ S = S$
- $S \circ (T \circ U) = (S \circ T) \circ U$

Preorder on Substitutions

A substitution S is more general than T if there exists U s.t. $T = U \circ S$.

- S is more general than T because T is obtained by instantiation of S

Unifier

A substitution S is a unifier of the set of terms $\{\sigma_1, \ldots, \sigma_n\}$ if $S\sigma_1 = \ldots = S\sigma_n$

Examples

- Subst. $\{\mathbb{B}/v, \mathbb{B} \times \mathbb{N}/u\}$ unifies $\{v \times \mathbb{N} \to \mathbb{N}, u \to \mathbb{N}\}$
- $\{\mathbb{B} \times \mathbb{B}/v, (\mathbb{B} \times \mathbb{B}) \times \mathbb{N}/u\}$ too!
- $\{v \times \mathbb{N}/u\}$ too!
- $\{\mathbb{N} \to s, t \times u\}$ are not unifiable
- $\{s \to \mathbb{N}, s\}$ are not unifiable

Most General Unifier (MGU)

Substitution S is a MGU of $\{\sigma_1, \ldots, \sigma_n\}$ if

1. S is a unifier of $\{\sigma_1, \ldots, \sigma_n\}$
2. S is more general than any other unifier of $\{\sigma_1, \ldots, \sigma_n\}$

Examples

- Subst. $\{\mathbb{B}/v, \mathbb{B} \times \mathbb{N}/u\}$ unifies $\{v \times \mathbb{N} \to \mathbb{N}, u \to \mathbb{N}\}$ but is not a MGU since it is an instance of the unifier $\{v \times \mathbb{N}/u\}$
- $\{v \times \mathbb{N}/u\}$ is a MGU of the this set

Theorem

If $\{\sigma_1, \ldots, \sigma_n\}$ is unifiable, then there exists a MGU and, moreover, it is unique up to renaming of variables.

Unification Algorithm

Unification Algorithm for Pairs of Types

- Input: Ordered pair of types $\sigma_1 \doteq \sigma_2$
- Output:
  1. MGU S of $\sigma_1 \doteq \sigma_2$, if $\sigma_1 \doteq \sigma_2$ is unifiable
  2. `fail`, otherwise

Unification algorithm for sets of types

In order to unify $\{\sigma_1, \ldots, \sigma_n\}$ with $n > 2$,

1. obtain MGU S of $\sigma_1 \doteq \sigma_2$
2. then recursively compute MGU T of $\{S\sigma_2, \ldots, S\sigma_n\}$
3. The MGU of $\{\sigma_1, \ldots, \sigma_n\}$ is $T \circ S$

Martelli-Montanari Algorithm

- Non-deterministic algorithm
- Consists of simplification rules that simplify sets of pairs of types that must be unified (*goals*)

$$G_0 \mapsto G_1 \mapsto \ldots \mapsto G_n$$

- Sequences that terminate in an empty goal are successful; those that terminate in `fail` fail
- Some simplification steps carry a substitution that represents a partial solution to the problem

$$G_0 \mapsto G_1 \mapsto_{S_1} G_2 \mapsto \ldots \mapsto_{S_k} G_n$$

- If the sequence is successful the MGU is $S_k \circ \ldots \circ S_1$

Rules of the Martelli-Montanari Algorithm

1. **Decomposition**
   $\{\sigma_1 \to \sigma_2 \doteq \tau_1 \to \tau_2\} \cup G \mapsto \{\sigma_1 \doteq \tau_1, \sigma_2 \doteq \tau_2\} \cup G$
   $\{\mathbb{N} \doteq \mathbb{N}\} \cup G \mapsto G$
   $\{\mathbb{B} \doteq \mathbb{B}\} \cup G \mapsto G$
2. **Trivial Pair Elimination**
   $\{s \doteq s\} \cup G \mapsto G$
3. **Swap**: if σ is not a variable
   $\{\sigma \doteq s\} \cup G \mapsto \{s \doteq \sigma\} \cup G$
4. **Variable Elimination**: if $s \notin FV(\sigma)$
   $\{s \doteq \sigma\} \cup G \mapsto_{\sigma/s} G[\sigma/s]$
5. **Fail**
   $\{\sigma \doteq \tau\} \cup G \mapsto \texttt{fail}$, con $(\sigma, \tau) \in T \cup T^{-1}$ y
   $T = \{(\mathbb{B}, \mathbb{N}), (\mathbb{N}, \sigma_1 \to \sigma_2), (\mathbb{B}, \sigma_1 \to \sigma_1)\}$
6. **Occur check**: si $s \neq \sigma$ y $s \in FV(\sigma)$
   $\{s \doteq \sigma\} \cup G \mapsto \texttt{fail}$

Example – Successful Sequence

$$
\begin{array}{ll}
 & \{(\mathbb{N} \to x) \to (x \to u) \doteq z \to (y \to y) \to z\} \\
\mapsto^{1} & \{\mathbb{N} \to x \doteq z, x \to u \doteq (y \to y) \to z\} \\
\mapsto^{3} & \{z \doteq \mathbb{N} \to x, x \to u \doteq (y \to y) \to z\} \\
\mapsto^{4}_{\mathbb{N} \to x/z} & \{x \to u \doteq (y \to y) \to (\mathbb{N} \to x)\} \\
\mapsto^{1} & \{x \doteq y \to y, u \doteq \mathbb{N} \to x\} \\
\mapsto^{4}_{y \to y/x} & \{u \doteq \mathbb{N} \to (y \to y)\} \\
\mapsto^{4}_{\mathbb{N} \to (y \to y)/u} & \emptyset
\end{array}
$$

- The MGU is $\{\mathbb{N} \to (y \to y)/u\} \circ \{y \to y/x\} \circ \{\mathbb{N} \to x/z\} = \{\mathbb{N} \to (y \to y)/z, y \to y/x, \mathbb{N} \to (y \to y)/u\}$

Example – Failed Sequence

$$
\begin{array}{ll}
 & \{x \to (y \to x) \doteq y \to ((x \to \mathbb{N}) \to x)\} \\
\mapsto^1 & \{x \doteq y, y \to x \doteq (x \to \mathbb{N}) \to x\} \\
\mapsto^4_{y/x} & \{y \to y \doteq (y \to \mathbb{N}) \to y\} \\
\mapsto^1 & \{y \doteq y \to \mathbb{N}, y \doteq y\} \\
\mapsto^6 & \texttt{fail}
\end{array}
$$

Properties of the Algorithm

Theorem

- The Martelli-Montanari algorithm always terminates
- Let G be the set of of pairs of types
  - if G has a unifier, the algorithm will terminate successfully and return an MGU
  - if G has no unifier, the algorithm terminates with a `fail`

Inference Algorithm

- We'll present the type inference algorithm for $\lambda_V^{\mathbb{B},\mathbb{N},\rightarrow}$
- Aim: define $\mathbb{W}(U)$ by recursion on the structure of U
- Frist we address the case for constants and variables, then we address the others
- We will make use of the unification algorithm

Inference Algorithm (constants and variables)

$$
\begin{aligned}
\mathbb{W}(\mathit{0}) &\stackrel{\text{def}}{=} \emptyset \rhd \mathit{0} : \mathbb{N} \\
\mathbb{W}(\mathit{true}) &\stackrel{\text{def}}{=} \emptyset \rhd \mathit{true} : \mathbb{B} \\
\mathbb{W}(\mathit{false}) &\stackrel{\text{def}}{=} \emptyset \rhd \mathit{false} : \mathbb{B} \\
\mathbb{W}(x) &\stackrel{\text{def}}{=} \{x : s\} \rhd x : s, \quad s \text{ fresh variable}
\end{aligned}
$$

Inference Algorithm (*succ*)

- Let $\mathbb{W}(U) = \Gamma \triangleright M : \tau$
- Let $S = MGU\{\tau \doteq \mathbb{N}\}$
- Then

$$\mathbb{W}(succ(U)) \stackrel{\text{def}}{=} S\Gamma \triangleright S\, succ(M) : \mathbb{N}$$

- Note: Case *pred* is similar

Inference Algorithm (*iszero*)

- Let $\mathbb{W}(U) = \Gamma \rhd M : \tau$
- Let $S = MGU\{\tau \doteq \mathbb{N}\}$
- Then

$$\mathbb{W}(\mathit{iszero}(U)) \stackrel{\text{def}}{=} S\Gamma \rhd S\ \mathit{iszero}(M) : \mathbb{B}$$

Inference Algorithm (ifThenElse)

- Let
  - $\mathbb{W}(U) = \Gamma_1 \triangleright M : \rho$
  - $\mathbb{W}(V) = \Gamma_2 \triangleright P : \sigma$
  - $\mathbb{W}(W) = \Gamma_3 \triangleright Q : \tau$
  - All type variables in $\mathbb{W}(U), \mathbb{W}(V)$ and $\mathbb{W}(W)$ must be different; if they are not we rename them
- Let

$$S = MGU(\{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_i \wedge x : \sigma_2 \in \Gamma_j, i \neq j\} \cup \{\sigma \doteq \tau, \rho \doteq \mathbb{B}\})$$

- Then

$$\mathbb{W}(\mathit{if}\ U\ \mathit{then}\ V\ \mathit{else}\ W) \stackrel{\text{def}}{=} S\Gamma_1 \cup S\Gamma_2 \cup S\Gamma_3 \triangleright S(\mathit{if}\ M\ \mathit{then}\ P\ \mathit{else}\ Q) : S\sigma$$

Inference Algorithm (application)

- Let
  - $\mathbb{W}(U) = \Gamma_1 \rhd M : \tau$
  - $\mathbb{W}(V) = \Gamma_2 \rhd N : \rho$
  - All type variables in $\Gamma_2 \rhd N : \rho$ must be renamed to be disjoint from those in $\mathbb{W}(U)$
- Let

$$S = MGU\{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_1 \wedge x : \sigma_2 \in \Gamma_2\} \cup \{\tau \doteq \rho \to t\} \text{ with } t \text{ a fresh variable}$$

- Then

$$\mathbb{W}(U\,V) \stackrel{\text{def}}{=} S\Gamma_1 \cup S\Gamma_2 \rhd S(M\,N) : St$$

Inference Algorithm (abstraction)

- Let $\mathbb{W}(U) = \Gamma \rhd M : \rho$
- If the context has type information on x (i.e. $x : \tau \in \Gamma$ for some τ), then

$$\mathbb{W}(\lambda x.U) \stackrel{\text{def}}{=} \Gamma \setminus \{x : \tau\} \rhd \lambda x : \tau.M : \tau \to \rho$$

- If the context has no information on the type of x (i.e. $x \notin Dom(\Gamma)$) we choose a fresh type variable s and then

$$\mathbb{W}(\lambda x.U) \stackrel{\text{def}}{=} \Gamma \rhd \lambda x : s.M : s \to \rho$$

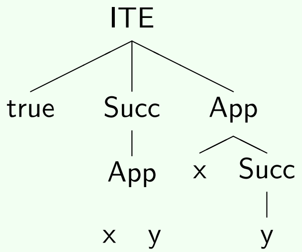
Inference Algorithm (*fix*)

- Let $\mathbb{W}(U) = \Gamma \triangleright M : \tau$
- Let $S = MGU\{\tau \doteq t \to t\}$, t fresh variable

$$\mathbb{W}(\mathit{fix}(U)) \stackrel{\mathrm{def}}{=} S\Gamma \triangleright S\,\mathit{fix}(M) : St$$

Example

- *if true then succ(x y) else x (succ(y))*
- We'll apply the algorithm, step-by-step



Example (1/4)

$$\textit{if } \textit{true} \textit{ then } \textit{succ}(x\,y) \textit{ else } x\,(\textit{succ}(y))$$

$$\mathbb{W}(\textit{true}) \quad = \quad \emptyset \triangleright \textit{true} : \mathbb{B}$$

Example (2/4)

$$\textit{if true then } succ(x\,y) \textit{ else } x\,(succ(y))$$

$$\begin{array}{rcl}
\mathbb{W}(x) & = & \{x : s\} \rhd x : s \\
\mathbb{W}(y) & = & \{y : t\} \rhd y : t \\
\mathbb{W}(x\,y) & = & \{x : t \to r, y : t\} \rhd x\,y : r \\
 & & \text{where } S = MGU(\{s \doteq t \to r\}) = \{t \to r/s\} \\
\mathbb{W}(succ(x\,y)) & = & \{x : t \to \mathbb{N}, y : t\} \rhd succ(x\,y) : \mathbb{N} \\
 & & \text{where } S = MGU(\{r \doteq \mathbb{N}\}) = \{\mathbb{N}/r\}
\end{array}$$

Example (3/4)

$$\mathit{if}\ \ \mathit{true}\ \ \mathit{then}\ \mathit{succ}(x\,y)\ \mathit{else}\ x\,(\mathit{succ}(y))$$

$$\begin{array}{rcl}
\mathbb{W}(y) & = & \{y : v\} \rhd y : v \\
\mathbb{W}(\mathit{succ}(y)) & = & \{y : \mathbb{N}\} \rhd \mathit{succ}(y) : \mathbb{N} \\
 & & \text{where } S = \mathit{MGU}(\{v \doteq \mathbb{N}\}) = \{\mathbb{N}/v\} \\
\mathbb{W}(x) & = & \{x : u\} \rhd x : u \\
\mathbb{W}(x\,\mathit{succ}(y)) & = & \{x : \mathbb{N} \to w, y : \mathbb{N}\} \rhd x\,\mathit{succ}(y) : w \\
 & & \text{where } \mathit{MGU}(\{u \doteq \mathbb{N} \to w\}) = \{\mathbb{N} \to w/u\}
\end{array}$$

Example (4/4)

$$M = \textit{if } \textit{true then succ}(x\,y) \textit{ else } x\,(\textit{succ}(y))$$

- $\mathbb{W}(\textit{true}) = \emptyset \rhd \textit{true} : \mathbb{B}$
- $\mathbb{W}(\textit{succ}(x\,y)) = \{x : t \to \mathbb{N}, y : t\} \rhd \textit{succ}(x\,y) : \mathbb{N}$
- $\mathbb{W}(x\,\textit{succ}(y)) = \{x : \mathbb{N} \to w, y : \mathbb{N}\} \rhd x\,\textit{succ}(y) : w$

$$\mathbb{W}(M) \quad = \quad \{x : \mathbb{N} \to \mathbb{N}, y : \mathbb{N}\} \rhd M : \mathbb{N}$$

where $S = MGU(\{t \to \mathbb{N} \doteq \mathbb{N} \to w,\ t \doteq \mathbb{N}, \mathbb{N} \doteq w\}) =$
$= \{\mathbb{N}/t, \mathbb{N}/w\}$

An Example that Fails

$$M = \mathit{if}\ \ \mathit{true}\ \ \mathit{then}\ x\,\underline{2}\ \mathit{else}\ x\,\mathit{true}$$

$$\begin{array}{rcl}
\mathbb{W}(x) & = & \{x : s\} \rhd x : s \\
\mathbb{W}(\underline{2}) & = & \emptyset \rhd \underline{2} : \mathbb{N} \\
\mathbb{W}(x\,\underline{2}) & = & \{x : \mathbb{N} \to t\} \rhd x\,\underline{2} : t \\
\\
\mathbb{W}(x) & = & \{x : u\} \rhd x : u \\
\mathbb{W}(\mathit{true}) & = & \emptyset \rhd \mathit{true} : \mathbb{B} \\
\mathbb{W}(x\,\mathit{true}) & = & \{x : \mathbb{B} \to v\} \rhd x\,\underline{2} : v \\
\\
\mathbb{W}(M) & = & \mathit{fail}
\end{array}$$

there is no $MGU(\{\mathbb{N} \to t \doteq \mathbb{B} \to v\})$

Complexity

- Both unification and type inference for $\lambda_V^{\mathbb{B},\to}$ are linear
- The principal type associated to a term without annotations can be exponential in the size of the term

Consider $P^n\,M$ where $P : s \to s \times s$ and $M : \sigma$

- Does this contradict the statement in the first item?
- No. They may be represented as dags (in which case the size of the principal type is $\mathcal{O}(n)$)
- NB: In the presence for polymorphism type inference is exponential