Simply typed lambda calculus : $\lambda\!\rightarrow$

$\mathbb{V} = \{\alpha, \beta, \gamma, \ldots\}$ infinite set of **type variables**

$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \rightarrow \mathbb{T}$ ... set of all **simple types**

(1) type variable: if $\alpha \in \mathbb{V}$, then $\alpha \in \mathbb{T}$
τ meta variable that represents a type variable

(2) arrow type: if $\sigma, \tau \in \mathbb{T}$, then $(\sigma \rightarrow \tau) \in \mathbb{T}$

Examples : γ, $(\beta \rightarrow \gamma)$, $((\gamma \rightarrow \alpha) \rightarrow (\alpha \rightarrow (\beta \rightarrow \gamma)))$

Notation : •) drop outermost parentheses : $\beta \rightarrow \gamma \overset{\text{stands for}}{=} (\beta \rightarrow \gamma)$

•) parentheses of arrow types are right-associative : $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Syntactical identity on $\mathbb{T}$: $\alpha \equiv \alpha$, $\alpha \not\equiv \beta$, $(\beta \rightarrow \gamma) \equiv (\beta \rightarrow \gamma)$, $\alpha \rightarrow \beta \rightarrow \gamma \not\equiv \beta \rightarrow \gamma$, ...

We have to be careful about different types of variables: •) variables from λ-calculus: $a, b, c, \ldots$
•) type variables : $\alpha, \beta, \gamma, \ldots$
•) meta variables : $A, B, C, \ldots$
may also use $a, b, c, \ldots, \alpha, \beta, \gamma, \ldots$ as meta-variables for variables and type variables respectively

Interpretation : •) type variables : basic types
•) arrow types : function types

Definition : A **statement** or **typing statement** $M : \sigma$
where $M \in \Lambda$ and $\sigma \in \mathbb{T}$ means "M is of type σ"

Every variable has a unique type : $x : \sigma, x : \tau \Rightarrow \sigma \equiv \tau$

Outlook: if $M : \sigma \rightarrow \tau$ and $N : \sigma$, then $M\,N : \tau$ (application)
if $x : \sigma$ and $M : \tau$, then $\lambda x.M : \sigma \rightarrow \tau$ (abstraction)

Typable term : $M \in \Lambda$ is called **typable** if there is a $\sigma \in \mathbb{T}$ such that $M : \sigma$

Note: if $f : (\alpha \rightarrow \beta \rightarrow \gamma)$, $x : \alpha$, and $y : \beta$, then $(f\,x\,y) : \gamma$,
notations fit together

How to type a term ? Church-typing and Curry-typing

Church typing : prescribes a unique type to every variable upon its introduction,
also called explicit typing

Curry typing : find type by search process such that rules of typing are satisfied,
also called implicit typing

Examples: •) explicit
if $x : \alpha \rightarrow \alpha$, $y : (\alpha \rightarrow \alpha) \rightarrow \beta$, then $y\,x : \beta$
if $z : \beta$, $u : \gamma$, then $\lambda z\,u.z : \beta \rightarrow \gamma \rightarrow \beta$ and hence $(\lambda z\,u.z)(y\,x)$ is
permitted and $(\lambda z\,u.z)(y\,x) : \gamma \rightarrow \beta$ - In particular $(\lambda z\,u.z)(y\,x)$ is typable

a) simplified

Try to type $M \equiv (\lambda z\, u.\, z)(y\, x)$, i.e. find types for x, y, z, u such that M has a valid type

M is an application $\Rightarrow \lambda z\, u.\, z : A \to B$, $y\, x : A$, $M : B$

$\lambda z\, u.\, z : A \to B \Rightarrow z : A$ & $\lambda u.\, z : B$

$\lambda u.\, z$ must have a function type, i.e. $B \equiv C \to D \Rightarrow u : C$, $z : D \Rightarrow A \equiv D$

$y\, x$ is an application $\Rightarrow y : E \to F$, $x : E$, $y\, x : F \Rightarrow A \equiv F \equiv D$

Summary: $x : E$, $y : E \to A$, $z : A$, $u : C$, $M : C \to A$

Choose: $E \equiv \alpha$, $A \equiv \beta$, $C \equiv \gamma$ $\to$ $x : \alpha$, $y : \alpha \to \beta$, $z : \beta$, $u : \gamma$

Then: $y\, x : \beta$, $\lambda u.\, z : \gamma \to \beta$, $\lambda z\, u.\, z : \beta \to \gamma \to \beta$, $M \equiv (\lambda z\, u.\, z)(y\, x) : \gamma \to \beta$

From now on: use explicit typing

Definition: $\boxed{\Lambda_\Pi} = V \mid (\Lambda_\Pi \Lambda_\Pi) \mid (\lambda V : \Pi.\, \Lambda_\Pi)$ ... set of pre-typed λ-terms

1) **Statement**: $M : \sigma$ (subject ← M, type ← σ) where $M \in \Lambda_\Pi$, $\sigma \in \Pi$
2) **Declaration**: statement with a variable as subject: $x : \alpha$
3) **Context**: list of declarations with different subjects: $x : \alpha, y : \beta, z : \gamma$
4) **Judgement**: $\Gamma \vdash M : \sigma$ where Γ context, $M : \sigma$ statement
   read: "in the context Γ, M has type σ"

Derivation rules for $\lambda \to$ (premiss-less conclusion ← $\frac{}{\Gamma \vdash x : \sigma}$)

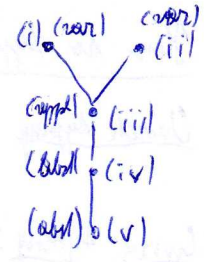
(var) $\Gamma \vdash x : \sigma$ if $x : \sigma \in \Gamma$, may write as $\frac{}{\Gamma \vdash x : \sigma}$

(appl) $\dfrac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$ (premises → conclusion)

(abst) $\dfrac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma.\, M : \sigma \to \tau}$ (extended context)

Example:

$$\dfrac{\dfrac{\dfrac{\dfrac{(i)\ y : \alpha \to \beta, z : \alpha \vdash y : \alpha \to \beta \quad (ii)\ y : \alpha \to \beta, z : \alpha \vdash z : \alpha}{(iii)\ y : \alpha \to \beta, z : \alpha \vdash y z : \beta}(appl)}{(iv)\ y : \alpha \to \beta \vdash \lambda z : \alpha.\, y z : \alpha \to \beta}(abst)}{(v)\ \phi \vdash \lambda y : \alpha \to \beta.\, \lambda z : \alpha.\, y z : (\alpha \to \beta) \to \alpha \to \beta}(abst)}$$



They formed: $y : \alpha \to \beta$ ← flags represent context

(i) $y : \alpha \to \beta$
(ii) $z : \alpha$
(iii) $y z : \beta$
(iv) $\lambda z : \alpha.\, y z : \alpha \to \beta$
(v) $\lambda y : \alpha \to \beta.\, \lambda z : \alpha.\, y z : (\alpha \to \beta) \to \alpha \to \beta$

Compare logic

first third for Curry-Howard isomorphism

$\dfrac{A \Rightarrow B \quad A}{B}$ ($\Rightarrow$-elim) (modus ponens)

Assume $: A$ ... B / $A \Rightarrow B$ ($\Rightarrow$-intro)