2.1 (a) $x\,x\,y$: $x : \alpha \to \beta$, $x : \alpha$, $\alpha = \alpha \to \beta$ ↯

(b) $x\,y\,y$: $x : \alpha \to \beta$, $y : \alpha$, $xy : \beta$, $xy : \alpha \to \gamma$, $\beta = \alpha \to \gamma$, $xyy : \gamma$

(c) $x\,y\,x$: $x : \alpha \to \beta$, $y : \alpha$, $xy : \beta$, $xy : (\alpha \to \beta) \to \gamma$, $\beta = (\alpha \to \beta) \to \gamma$ ↯

(d) $x\,(x\,y)$: $x : \alpha \to \beta$, $y : \alpha$, $xy : \beta$, $xy : \alpha$, $\alpha = \beta$, $x\,(xy) : \alpha$

(e) $x\,(y\,x)$: $x : \alpha \to \beta$, $y : \gamma \to \delta$, $x : \gamma$, $yx : \delta$, $\alpha = \delta$, $\gamma = \alpha \to \beta$

$x : \alpha \to \beta$, $y : (\alpha \to \beta) \to \alpha$, $yx : \alpha$, $x\,(yx) : \beta$

(b) $x : \alpha \to \alpha \to \gamma$, $y : \alpha$, $xy : \alpha \to \gamma$, $xyy : \gamma$

(d) $x : \alpha \to \alpha$, $y : \alpha$, $xy : \alpha$, $x\,(xy) : \alpha$

(e) $x : \alpha \to \beta$, $y : (\alpha \to \beta) \to \alpha$, $yx : \alpha$, $x\,(yx) : \beta$

2.2) Find types for zero, one, and two.

zero $= \lambda f x.\, x$, one $= \lambda f x.\, fx$, two $= \lambda f x.\, f(fx)$

$\lambda f : \alpha .\, \lambda x : \beta .\, x$, $: \alpha \to \beta \to \beta$

$\lambda f : \alpha \to \beta .\, \lambda x : \alpha .\, fx$, $: (\alpha \to \beta) \to \alpha \to \beta$

$\lambda f : \alpha \to \alpha .\, \lambda x : \alpha .\, f(fx)$, $: (\alpha \to \alpha) \to \alpha \to \alpha$

2.3) Find types for $K := \lambda x y.\, x$, $S := \lambda x y z.\, xz(yz)$

$(\lambda x : \alpha .\, \lambda y : \beta .\, x) : \alpha$

$\left(\lambda x : \alpha \to \alpha \to \alpha .\, \lambda y : \alpha \to \alpha .\, \lambda z : \alpha .\, xz(yz)\right) :$

$(\alpha \to \alpha \to \alpha) \to (\alpha \to \alpha) \to \alpha \to \alpha$

to bound variables

2.4) Add types ≠ to create legal pre-typed λ-terms, give types

(a) $\lambda x y z.\, x(yz)$

$\vdash \left(\lambda x : \alpha \to \alpha .\, \lambda y : \alpha \to \alpha .\, \lambda z : \alpha .\, x(yz)\right) : (\alpha \to \alpha) \to (\alpha \to \alpha) \to \alpha \to \alpha$

(b) $\lambda x y z.\, y\,(xz)x$

$\vdash \left(\lambda x : \alpha \to \alpha .\, \lambda y : \alpha \to (\alpha \to \alpha) \to \alpha .\, \lambda z : \alpha .\, y\,(xz)x\right) : (\alpha \to \alpha) \to (\alpha \to (\alpha \to \alpha) \to \alpha) \to \alpha \to \alpha$

2.5) Find pre-typed version to make term typable or show it's impossible

(a) $\lambda x y.\, x\,(\lambda z.\, y)\,y$

$\lambda x : \alpha , \lambda y : \beta .\, x\,(\lambda z : \gamma .\, y)\,y$

$\alpha = \alpha_1 \to \alpha_2$, $(\lambda z : \gamma .\, y) : \alpha_1$, $\alpha_1 = \gamma \to \beta$, $\alpha_2 = \alpha_3 \to \alpha_4$, $\alpha_3 = \beta$

$\left(\lambda x : (\gamma \to \beta) \to \beta \to \alpha_4 .\, \lambda y : \beta .\, (\lambda z : \gamma .\, y)\,y\right) : ((\gamma \to \beta) \to \beta \to \alpha_4) \to \beta \to \alpha_4$

(b) $\lambda x y.\, x\,(\lambda z.\, x)\,y$

$\lambda x : \alpha .\, \lambda y : \beta .\, x\,(\lambda z : \gamma .\, x)\,y$

$\alpha = \alpha_1 \to \alpha_2$, $(\lambda z : \gamma .\, x) : \alpha_1$, $\alpha_1 = \gamma \to \alpha$, $\alpha = (\gamma \to \alpha) \to \alpha_2$ ↯

Kinds of problems to solve in type theory

Definition: A pre typed λ-term $M \in \Lambda_T$ is called **legal** if there exists a context Γ and a type σ such that $\Gamma \vdash M : \sigma$

Example: $\lambda z : \iota . yz$ is legal because $y : \iota \to \beta \vdash \lambda z : \iota . yz : \iota \to \beta$

Problems: 1) Well-typedness (typability): $? \vdash \text{term} : ?$

1a) Type assignment: $\text{context} \vdash \text{term} : ?$

2) Type checking: $\text{context} \overset{?}{\vdash} \text{term} : \text{type}$

3) Term finding (term construction, or inhabitation): $\text{context} \vdash ? : \text{type}$

checking → proof (later)

↑

the real problem: proving! (later)

In $\lambda\to$: algorithms for all 3 problems!

later: 3) undecidable

1) Well-typedness: $M \equiv \lambda y : \iota \to \beta . \lambda z : \iota . yz$

$? \vdash M : ?$, i.e. is M legal?

Example: $M \equiv \lambda y : \iota \to \beta . \lambda z : \iota . yz$

Find Γ and σ s.t. $\Gamma \vdash M : \sigma$

$\Gamma = \emptyset$ (no free variables in M)

- $\lambda y : \iota \to \beta . \lambda z : \iota . yz : ?$ ← goal
- (a) $y : \iota \to \beta$
  - ⋮
  - (m) $\lambda z : \iota . yz : ?$ ← new goal
  - (n) $\lambda y : \iota \to \beta . \lambda z : \iota . yz : \ldots$ (abst) on (m)
- (a) $y : \iota \to \beta$
  - (b) $z : \iota$
    - ⋮
    - (l) $yz : ?$ ← new goal
  - (m) $\lambda z : \iota . yz : \ldots$ (abst) on (l)
  - (n) $\lambda y : \iota \to \beta . \lambda z : \iota . yz : \ldots$ (abst) on (m)

•) (a) $y : \mathcal{L} \to \mathcal{B}$

(b) $z : \mathcal{L}$

$\vdots$

(k_1) $y : ?_1$ ← new goals

$\vdots$

(k_2) $z : ?_2$ ←

(l) $yz : \ldots$ (appl) on (k_1) and (k_2)

(m) $\lambda z : \mathcal{L} . yz : \ldots$ (abst) on (l)

(n) $\lambda y : \mathcal{L} \to \mathcal{B} . \lambda z : \mathcal{L} . yz : \ldots$ (abst) on (m)

•) (a) $y : \mathcal{L} \to \mathcal{B}$

(b) $z : \mathcal{L}$

(k_1) $y : \mathcal{L} \to \mathcal{B}$ (var) on (a)

(k_2) $z : \mathcal{L}$ (var) on (b)

(l) $yz : \mathcal{B}$ (appl) on (k_1) and (k_2)

(m) $\lambda z : \mathcal{L} . yz : \mathcal{L} \to \mathcal{B}$ (abst) on (l)

(n) $\lambda y : \mathcal{L} \to \mathcal{B} . \lambda z : \mathcal{L} . yz : (\mathcal{L} \to \mathcal{B}) \to \mathcal{L} \to \mathcal{B}$ (abst) on (m)

Hence: $\Phi \vdash M : (\mathcal{L} \to \mathcal{B}) \to \mathcal{L} \to \mathcal{B}$

Remarks: Derivations are not unique in general.
Derivation fails iff term is not eapel (e.g. $M \equiv \lambda y : \mathcal{L} \to \mathcal{B} . \lambda z : \mathcal{B} . yz$)

2) Type checking: context $\overset{?}{\vdash}$ term : type

Example: $x : \mathcal{L} \to \mathcal{L}, y : (\mathcal{L} \to \mathcal{L}) \to \mathcal{B} \overset{?}{\vdash} (\lambda z : \mathcal{B} . \lambda u : \gamma . z)(yz) : \gamma \to \mathcal{B}$

•) (a) $x : \mathcal{L} \to \mathcal{L}$

(b) $y : (\mathcal{L} \to \mathcal{L}) \to \mathcal{B}$

$\vdots$

(m) $(\lambda z : \mathcal{B} . \lambda u : \gamma . z)(yx) : \gamma \to \mathcal{B}$

•) (a) $x : \mathcal{L} \to \mathcal{L}$

(b) $y : (\mathcal{L} \to \mathcal{L}) \to \mathcal{B}$

$\vdots$

(m_1) $\lambda z : \mathcal{B} . \lambda u : \gamma . z : ?_1$

$\vdots$

(m_2) $yx : ?_2$

$(\lambda z : \mathcal{B} . \lambda u : \gamma . z)(yx) : \gamma \to \mathcal{B}$ (appl) on (m_1) and (m_2)

•) (a) $x : \mathcal{L} \to \mathcal{L}$

(b) $y : (\mathcal{L} \to \mathcal{L}) \to \mathcal{B}$

(c) $z : \mathcal{B}$

(d) $u : \gamma$

(1) $z : \mathcal{B}$

(2) $\lambda u : \gamma . z : \gamma \to \mathcal{B}$

(m_1) $\lambda z : \mathcal{B} . \lambda u : \gamma . z : \mathcal{B} \to \gamma \to \mathcal{B}$

(m_2) $yx : \mathcal{B}$

(n) $(\lambda z : \mathcal{B} . \lambda u : \gamma . z)(yx) : \gamma \to \mathcal{B}$

Convention: suppress non-essential uses of the (var)-rule

(var) on (a)

(abst) on (1)

(abst) on (2)

(appl) on (b) and (a)

(appl) on (m_1) and (m_2)

3) Term finding: context $\vdash ? :$ type ("Find inhabitant of a given type in a given context")

Example: $\Gamma = \phi$, $\sigma \equiv A \to B \to A$

Find M s.t. $\Gamma \vdash M : \sigma$

- (m) $? : A \to B \to A$
- (a) $x : A$
  - (m) $? : B \to A$
  - (m) $\ldots : A \to B \to A$ (abst) on (m)
- (a) $x : A$
  - (b) $y : B$
    - (l) $? : A$
    - (m) $\ldots : B \to A$ (abst) on (l)
  - (m) $\ldots : A \to B \to A$ (abst) on (m)
- (a) $x : A$
  - (b) $y : B$
    - (l) $x : A$ (var) on (a)
  - (m) $\lambda y : B . x : B \to A$ (abst) on (l)
  - (m) $\lambda x : A . \lambda y : B . x : A \to B \to A$ (abst) on (m)

Interpretation as a logical statement and its proof:

Read: $\to$ as $\Rightarrow$

σ as the proposition $A \Rightarrow B \Rightarrow A$ (in the empty context, i.e. for arbitrary A, B)

(a): Assume x is a proof of A

(b): Assume y is a proof of B

(l): Then x is (still) a proof of A

(m): So the function mapping y to x sends a proof of B to a proof of A, i.e. $\lambda y : B . x$ proves $B \Rightarrow A$ in the stated sense: there is a proof of A (= A is true)

(m): Consequently, $\lambda x : A . \lambda y : B . x$ proves $A \Rightarrow B \Rightarrow A$ in the empty context

Remark: • The above observation is called Curry - Howard - isomorphism or PAT - interpretation where PAT stands for both "propositions as types" and "proofs as terms"

• The final term $\lambda x : A . \lambda y : B . x$ encodes its derivation and the statement it proves (both can be recovered by the "recall - typedness" - algorithm)