Dependent types

Intuition:

$$\left.\begin{array}{l} f : A \to B \\ a : A \end{array}\right\} \Rightarrow f\,a : B$$

non-dependent function

Dependent function type:

$$\prod_{x:A} B(x) \qquad \left(\begin{array}{ll} \text{or} & (x:A) \to B(x) \\ \text{or} & \forall x : A.\ B(x) \end{array}\right)$$

$$\left.\begin{array}{l} f : \prod\limits_{x:A} B(x) \\ a : A \end{array}\right\} \Rightarrow f\,a : B(a)$$

$\prod\limits_{x:A} B(x)$ is also called a dependent product type

Informal example: ($\mathbb{B}$ = booleans)

$$\lambda b : \mathbb{B}.\ \text{if } b \text{ then } 5 \text{ else } \langle 5, \text{false} \rangle$$
$$: \prod_{b:\mathbb{B}} (\text{if } b \text{ then } \mathbb{N} \text{ else } \mathbb{N} \times \mathbb{B})$$

now, terms can appear "inside" types!

Pure Type Systems

(For the full details, see Barendregt Lambda Calculi with types, 5.1.10, 5.2)

A collection of type systems, some of which include dependent types

Informal example: a PTS can allow

$5 : \mathbb{N}$	type	"5 is a natural"
$\mathbb{N} : *$	kind	"$\mathbb{N}$ is a type"
$* : \square$	---	"$*$ is a kind"

In PTS, λ-calculi are defined according to which $\prod_{x:A} B(x)$ types are allowed.

For instance, we can allow:

$\prod_{x:A} B$ where

$A : -$	$B : -$	
$*$	$*$	terms depending on terms ($\lambda^{\rightarrow}$)
$\square$	$*$	terms depending on types (ex: System F)
$*$	$\square$	types depending on terms
$\square$	$\square$	types depending on types

Examples:

$(*, *)$ – simple types $\lambda^{\rightarrow}$

$$\tau : * \vdash \lambda x:\tau . x : \prod_{x:\tau} \tau$$

(green annotations: τ in the subscript $x:\tau$ has sort $*$, the body τ has sort $*$)

Note: we can define (x does not occur in σ)

$$\tau \to \sigma :\equiv \sigma^{\tau} :\equiv \prod_{x:\tau} \sigma$$

hence $\prod_{x:\tau} \tau = \tau \to \tau$

$(\square, *)$ (beyond $(*, *)$) = System F

$$\vdash \lambda \alpha:* . \lambda x:\alpha . x : \prod_{\alpha:*} \prod_{x:\alpha} \alpha$$

$$= \prod_{\alpha:*} (\alpha \to \alpha)$$

(green annotations: $\alpha:*$ is of sort $\square$, $(\alpha \to \alpha)$ is of sort $*$)

$$= \forall \alpha . \alpha \to \alpha$$

$(*, \square)$ (beyond $(*, *)$)

$\alpha : * \vdash$

$\lambda P:\alpha \to * . \lambda x:\alpha . \lambda h:Px . h :$

$$\prod_{P:\alpha \to *} \prod_{x:\alpha} (Px \to Px)$$

$$= \forall P, x . Px \to Px$$

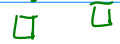
Note that $\alpha \to * = \prod_{y:\alpha} *$

(green annotations: α in $y:\alpha$ is of sort $*$, the body $*$ is of sort $\square$)

In a logic, $(*, \square)$ allows predicates

$(\Box, \Box)$

$\vdash \lambda \alpha : * . \alpha \to \alpha : \prod_{\alpha : *} * = (* \to *)$



λ-calculi which are PTSs and that involves products according to a subset of $\{(*,*), (*,\Box), (\Box,*), (\Box,\Box)\}$
($\uparrow$ this is always "in", though)
form the so-called λ-cube.
The Calculus of Constructions allows all the products we have seen in the examples.

A few selected rules (simplified)

$$\frac{\Gamma, x:A \vdash t : B \qquad \Gamma \vdash \prod_{x:A} B : s \quad (s \in \{*, \Box\})}{\Gamma \vdash \lambda x:A.t : \prod_{x:A} B} \; \Pi\text{-I}$$

$$\frac{\Gamma \vdash t : \Pi_{x:A} B \qquad \Gamma \vdash e : A}{\Gamma \vdash te : B\{e/x\}} \; \Pi\text{-E}$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x:A \vdash B : s_2 \quad s_1, s_2 \in \{*, \Box\}}{\Gamma \vdash \prod_{x:A} B : s_2} \; \Pi\text{-F}$$

type formation