Types dependent on types: the system $\lambda\underline{w}$

identity function on σ ↙ ↙ generalised identity function

Previously: abstracting a term from a type variable: $\lambda x:\sigma.x \longrightarrow \lambda\alpha:*.\lambda x:\alpha.x$

Now: abstracting a type from a type variable: $\beta\to\beta$, $\gamma\to\gamma$, $(\gamma\to\beta)\to(\gamma\to\beta)$ are all of the form $\Diamond\to\Diamond$

Introduce generalised expressions of the form $\lambda\alpha:*.\alpha\to\alpha$ with a type as a value (type constructors)

Applying a type constructor to a type yields a type:

$(\lambda\alpha:*.\alpha\to\alpha)\beta \to_\beta \beta\to\beta$

$(\lambda\alpha:*.\alpha\to\alpha)\gamma \to_\beta \gamma\to\gamma$

$(\lambda\alpha:*.\alpha\to\alpha)(\gamma\to\beta) \to_\beta (\gamma\to\beta)\to(\gamma\to\beta)$

We obtain the type constructor $\lambda\alpha:*.\alpha\to\alpha$ by abstracting the type $\alpha\to\alpha$ from the type variable α

More complex example: $\lambda\alpha:*.\lambda\beta:*:\alpha\to\beta$

What are the types of these type constructors?

$\lambda\alpha:*.\alpha\to\alpha$ is a function that takes a type and yields a type.

Educated guess for the types: $\lambda\alpha:*.\alpha\to\alpha : * \to *$ ← needs "super types" ↓

$\lambda\alpha:*.\lambda\beta:*:\alpha\to\beta : * \to (*\to *)$

We introduce the new set $\mathbb{K}$ of kinds by abstract syntax: $\mathbb{K} = *\mid(\mathbb{K}\to\mathbb{K})$

Use usual conventions for parentheses: outermost parentheses may be dropped and $\to$ is right associative:

$* \to (*\to *)\to * \;=\; (* \to((*\to *)\to *))$

Examples of kinds: $*$, $*\to *$, $(*\to *)\to *$, $*\to(*\to *)\to *$, ...

We use greek letters $\alpha,\beta,\gamma,\ldots$ for inhabitants of kinds (i.e. type constructors) too

What is the type of a kind?

We introduce a new "super super type" $\Box$ of all kinds: $*\to *:\Box$, $(*\to *)\to *:\Box$, $*:\Box$, ...

if $\mathcal{L}:\Box$ and $M:\mathcal{L}$, then M is called (type) constructor

if $\mathcal{L}\neq *$, then M is called proper (type) constructor

Summary and formal definition

$\mathbb{V} = \{a,b,c,\ldots\}$ ... set of term variables

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

$\mathbb{T} = \mathbb{V} \mid \mathbb{T}\to\mathbb{T}$ ... set of simple types

$\mathbb{K} = * \mid * \to *$ ... set of kinds

$\Lambda_\mathbb{T} = \mathbb{V} \mid (\Lambda_\mathbb{T}\ \Lambda_\mathbb{T}) \mid (\lambda\mathbb{V}:\mathbb{T}.\Lambda_\mathbb{T})$ ... set of terms

$\Lambda_{\underline{w}} = \mathbb{V} \mid (\Lambda_{\underline{w}}\ \Lambda_{\underline{w}}) \mid (\lambda\mathbb{V}:*.\Lambda_{\underline{w}})$ ... set of (type) constructors

$*$, $\Box$ ... sorts

Four levels of statements and judgements

Level 1 (terms): $x:\alpha$, $\lambda x:\alpha.x:\alpha\to\alpha$

Level 2 (constructors): $\alpha:*$, $\lambda\alpha:*.\alpha\to\alpha : * \to *$, $\alpha:*\to *, \gamma:* \vdash \alpha\gamma:*$

Level 3 (kinds): $* \to (*\to *)\to *:\Box$, $*:\Box$

Level 4 (super super type): $\Box$

Judgement chain: $x:\sigma:*\to *:\Box$

Derivation rules in $\lambda\underline{w}$

(sort) $\dfrac{}{\phi \vdash * : \Box}$

(var) $\dfrac{\Gamma \vdash A : s}{\Gamma, x:A \vdash x:A}$ if $x \notin \Gamma$ ← convention: $s \in \{*, \Box\}$

short for (var-type) $\dfrac{\Gamma \vdash A : *}{\Gamma, x:A \vdash x:A}$ if $x \notin \Gamma$

(var-kind) $\dfrac{\Gamma \vdash A : \Box}{\Gamma, x:A \vdash x:A}$ if $x \notin \Gamma$

The (var) rule allows for two things: 1) extending a context
2) derive a declaration in a context (the last one) as a statement

By this method we avoid the need to define valid contexts as we did in $\lambda 2$.

Example: different realisations of $A:s$ and $x:A$ for different s:

	$s \equiv \Box$		$s \equiv *$	
$A:s$	$*:\Box$	$*\to *:\Box$	$\mathcal{L}:*$	$\mathcal{L}\to\beta:*$
$x:A$	$\mathcal{L}:*$	$\beta:*\to *$	$x:\mathcal{L}$	$y:\mathcal{L}\to\beta$

Example: combination of (sort) and (var) to give first derivation:

(1) $\phi \vdash * : \Box$ (sort)

(2) $\mathcal{L}:* \vdash \mathcal{L}:*$ (var-kind)

(3) $\mathcal{L}:*, x:\mathcal{L} \vdash x:\mathcal{L}$ (var-type)

tree format →

(1) $*:\Box$ (sort)

(2) $\boxed{\mathcal{L}:*}$ $\mathcal{L}:*$ (var) on (1)

(3) $\boxed{x:\mathcal{L}}$ $x:\mathcal{L}$ (var) on (2)

The (var) rule in $\lambda\underline{w}$ is less general than in previous systems $\lambda\to$ and $\lambda 2$.
It only allows to derive the last declaration of a context as a statement.

So for we cannot derive e.g.: $\mathcal{L}:*, x:\mathcal{L} \vdash \mathcal{L}:*$

$\mathcal{L}:*, \beta:* \vdash \mathcal{L}:*$

or even $\mathcal{L}:*, \beta:* \vdash \beta:*$ since we cannot obtain the premiss $\mathcal{L}:* \vdash *:\Box$ which is necessary to get $\mathcal{L}:*, \beta:* \vdash \beta:*$ by the (var) rule

Solution: (weak) rule (weakening)

$$\frac{\Gamma \vdash A:B \quad \Gamma \vdash C:s}{\Gamma, x:C \vdash A:B} \quad \text{if } x \notin \Gamma$$

The weakening rule allows to "weaken" the context of a judgement by adding new declarations, provided that the types of the new declarations are "well-formed".

Note: "weakening" is a special form of "thinning": thinning means adding assumptions to a context while weakening means adding assumptions to a context at the end only.

Effect of (weak) rule: Assume we can derive $\Gamma \vdash A:B$ (first premiss) Then we may weaken the context by adding an arbitrary declaration at the end of the context (conclusion) provided the type of the new declaration is well-formed itself (second premiss)

Examples:

$$\dfrac{\dfrac{(1)\ \Phi \vdash * : \square \quad (\text{sort})}{(2)\ \mathcal{L} : * \vdash \mathcal{L} : * \quad (\text{var})} \qquad \dfrac{(1)\ \Phi \vdash * : \square \quad (\text{sort})}{(2)\ \mathcal{L} : * \vdash \mathcal{L} : * \quad (\text{var})}}{(4)\ \mathcal{L} : *, x : \mathcal{L} \vdash \mathcal{L} : * \quad (\text{weak})}$$

$$\dfrac{\dfrac{(1)\ \Phi \vdash * : \square \quad (\text{sort})}{(2)\ \mathcal{L} : * \vdash \mathcal{L} : * \quad (\text{var})} \qquad \dfrac{\dfrac{(1)\ \Phi \vdash * : \square \quad (\text{sort}) \qquad (1)\ \Phi \vdash * : \square \quad (\text{sort})}{(5)\ \mathcal{L} : * \vdash * : \square \quad (\text{weak})}}{(6)\ \mathcal{L} : *, B : * \vdash \mathcal{L} : * \quad (\text{weak})}$$

$$\dfrac{\dfrac{(1)\ \Phi \vdash * : \square \quad (\text{sort}) \qquad (1)\ \Phi \vdash * : \square \quad (\text{sort})}{(5)\ \mathcal{L} : * \vdash * : \square \quad (\text{weak})}}{(7)\ \mathcal{L} : *, B : * \vdash B : * \quad (\text{var})}$$

Tree format:

(1) $*$: $\square$ (sort)

$\boxed{\mathcal{L} : *}$

(2) $\mathcal{L} : *$ (var) on (1)

$\boxed{x : \mathcal{L}}$

(3) $x : \mathcal{L}$ (var) on (2)

(4) $\mathcal{L} : *$ (weak) on (2) and (2)

(5) $* : \square$ (weak) on (1) and (1)

$\boxed{B : *}$

(6) $\mathcal{L} : *$ (weak) on (2) and (5)

(7) $B : *$ (var) on (5)

The formation rule (form)

$$(\text{form})\ \dfrac{\Gamma \vdash A : s \qquad \Gamma \vdash B : s}{\Gamma \vdash A \to B : s}$$

Example:

(8) $\mathcal{L} \to B : *$ (form) on (6) and (7)

(9) $* \to * : \square$ (form) on (5) and (5)