The system λC – The Calculus of Constructions

"$\lambda C = \lambda 2 + \lambda \underline{\omega} + \lambda P$": allow all combinations of types/terms dependent on types/terms

One set of derivation rules for all systems:

(sort) $\emptyset \vdash * : \Box$

(var) $$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \quad \text{if } x \notin \Gamma$$

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

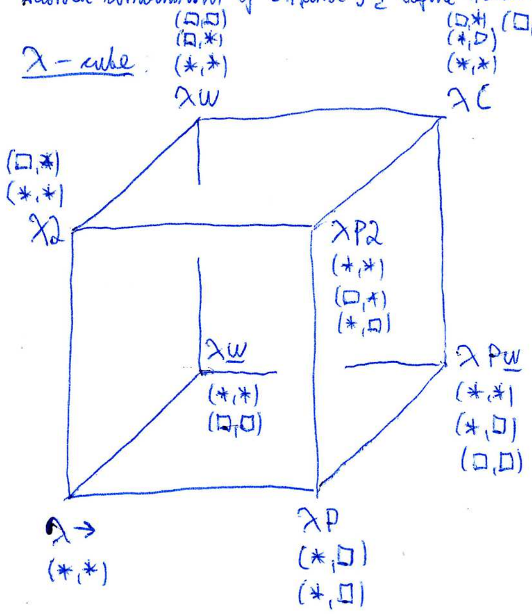
(form) $$\frac{\Gamma \vdash A : s_1, \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A . B : s_2}$$

(appl) $$\frac{\Gamma \vdash M : \Pi x : A . B, \quad \Gamma \vdash N : A}{\Gamma \vdash M N : B[x := N]}$$

(abst) $$\frac{\Gamma, x : A \vdash M : B, \quad \Gamma \vdash \Pi x : A . B : s}{\Gamma \vdash \lambda x : A . M : \Pi x : A . B}$$

(conv) $$\frac{\Gamma \vdash A : B, \quad \Gamma \vdash B' : s}{\Gamma \vdash A : B'} \quad \text{if } B =_\beta B'$$

Allowed combinations of s_1 and s_2 define which system we are in



$(*,*)$: term dep. on term

$() \vdash \lambda x : \mathcal{L} . x : \mathcal{L} \to \mathcal{L}$

$(*,\Box)$: type dep. on term

$A : *, P : A \to * \vdash \Pi x : A . P x : *$

$(\Box,\Box)$: type dep. on type

$() \vdash \lambda \mathcal{L} : * . \mathcal{L} \to \mathcal{L} : * \to *$

$(\Box,*)$: term dep. on type

$() \vdash \lambda \mathcal{L} : * . \lambda f : \mathcal{L} \to \mathcal{L} . \lambda x : \mathcal{L} . f x : \Pi \mathcal{L} : * . (\mathcal{L} \to \mathcal{L}) \to \mathcal{L} \to \mathcal{L}$

λC still has all "nice" properties:

- free variables lemma
- thinning, permutation, condensing lemma
- generation lemma
- subexpression lemma
- uniqueness of types up to conversion
- substitution lemma
- Church-Rosser theorem, confluence
- subject reduction lemma
- strong normalisation theorem
- decidability of well-typedness and type checking

Set of λC-expressions:

$$\mathcal{E} = V \mid * \mid \Box \mid (\mathcal{E}\mathcal{E}) \mid (\lambda V : \mathcal{E} . \mathcal{E}) \mid (\Pi V : \mathcal{E} . \mathcal{E})$$

Notation:

Usual conventions about variables, parentheses, successive abstractions, sorts ($s \in \{*, \Box\}$), write $A \to B$ for $\Pi x : A . B$ if $x \notin FV(B)$

We may extend λC by introducing universes or "super-kinds":

We set $\Box_0 := \Box$ and assume $\Box_i \subset \Box_{i+1}$ for all $i \in \mathbb{N}_0$

Add rules: $$\frac{\Gamma \vdash A : *}{\Gamma \vdash A : \Box_0} \qquad \frac{\Gamma \vdash A : \Box_i}{\Gamma \vdash A : \Box_{i+1}}$$

Replace (form): $$\frac{\Gamma \vdash A : \Box_i, \quad \Gamma, x : A \vdash B : \Box_j}{\Gamma \vdash \Pi x : A . B : \Box_{\max\{i,j\}}}$$

Another extension: Σ-types

$$\frac{\Gamma \vdash A : * \quad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Sigma x : A . B : *} \qquad \frac{\Gamma \vdash A : \Box_i \quad \Gamma, x : A \vdash B : \Box_j}{\Gamma \vdash \Sigma x : A . B : \Box_{\max\{i,j\}}}$$

Furthermore: inductive types $\to$ Calculus of Inductive Constructions

Encoding logical notions in λC

I, II Absurdity and negation

If $\bot$ is true, then every proposition is true → If $\bot$ is inhabitated, then all propositions are inhabitated
→ If $\bot$ is inhabitated, then there is a function mapping an arbitrary proposition $\mathcal{L}$ to an inhabitant of $\mathcal{L}$, such a function has type $\Pi \mathcal{L} : *. \mathcal{L}$

Set $\boxed{\bot :\equiv \Pi \mathcal{L} : *. \mathcal{L}}$

Get ($\bot$-elim) for free: ($\bot$-elim) $\dfrac{\bot}{A}$

(a) $\boxed{f : \Pi \mathcal{L} : *. \mathcal{L}}$
(i) $f A : A$ (appl) on (a) and $A : *$

$\bot$ lives in $\lambda 2$: need $s_1 = \square$ and $s_2 = *$ in (form) rule
can derive $\bot : *$ in $\lambda 2$

Set $\boxed{\neg A :\equiv A \to \bot}$

Also get ($\bot$-intro) for free: ($\bot$-intro) $\dfrac{A \quad \neg A}{\bot}$ ($\Rightarrow$-elim) $\dfrac{A \quad A \Rightarrow \bot}{\bot}$

Same for ($\neg$-intro), ($\neg$-elim): ($\neg$-intro) $\dfrac{\boxed{\text{Assume: } A} \quad \vdots \quad \bot}{\neg A}$ ($\Rightarrow$-intro) $\dfrac{\boxed{\text{Assume: } A} \quad \vdots \quad B}{A \Rightarrow B}$

($\neg$-elim) $\dfrac{\neg A \quad A}{\bot}$ ($\Rightarrow$-elim) $\dfrac{A \Rightarrow B \quad A}{B}$

III, IV Conjunction and disjunction

$\boxed{A \wedge B :\equiv \Pi C : *. (A \to B \to C) \to C}$ ("If A and B together imply C, then C holds on its own, i.e. that A and B hold is redundant")
Because used definition of $A \wedge B$ is known as "second order encoding of $A \wedge B$"; first order: $A \wedge B :\equiv \neg(A \to \neg B)$, only works in classical logic

($\wedge$-intro) $\dfrac{A \quad B}{A \wedge B}$ ($\wedge$-intro-sec) $\dfrac{A \quad B}{\Pi C : *. (A \to B \to C) \to C}$

($\wedge$-elim-left) $\dfrac{A \wedge B}{A}$ ($\wedge$-elim-sec-left) $\dfrac{\Pi C : *. (A \to B \to C) \to C}{A}$

($\wedge$-elim-right) $\dfrac{A \wedge B}{B}$ ($\wedge$-elim-sec-right) $\dfrac{\Pi C : *. (A \to B \to C) \to C}{B}$

Need to prove that second order rules are derivable in λC:

($\wedge$-intro-sec-AA) $\dfrac{\Gamma \vdash \mathcal{L} : A \qquad \Gamma \vdash b : B}{\Gamma \vdash ?_1 : \Pi C : *. (A \to B \to C) \to C}$

($\wedge$-elim-left-sec-AA) $\dfrac{\Gamma \vdash c : \Pi C : *. (A \to B \to C) \to C}{\Gamma \vdash ?_2 : A}$

($\wedge$-elim-right-sec-AA) $\dfrac{\Gamma \vdash d : \Pi C : *. (A \to B \to C) \to C}{\Gamma \vdash ?_3 : B}$

Conclusion: all necessary rules already derivable in λC and don't need to be added!

Example:
(a) $\boxed{A : *}$
(b) $\boxed{B : *}$
(c) $\boxed{x : A}$
(d) $\boxed{y : B}$
(e) $\boxed{C : *}$
(f) $\boxed{z : A \to B \to C}$
(1) $z x : B \to C$ (appl) (f), (c)
(2) $z x y : C$ (appl) (1), (d)
(3) $\lambda z : A \to B \to C. z x y : (A \to B \to C) \to C$ (abst) (2)
(4) $\lambda C : *. \lambda z : A \to B \to C. z x y : \Pi C : *. (A \to B \to C) \to C$ (abst) (3)