$\boxed{A \vee B :\equiv \Pi C:*.(A\to C)\to(B\to C)\to C}$ (first order: $A \vee B :\equiv \neg A \to B$)

(∨-intro-left) $\dfrac{A}{A\vee B}$ (∨-intro-left-sec) $\dfrac{A}{\Pi C:*.(A\to C)\to(B\to C)\to C}$

(∨-intro-right) $\dfrac{B}{A\vee B}$ (∨-intro-right-sec) $\dfrac{B}{\Pi C:*.(A\to C)\to(B\to C)\to C}$

(∨-elim) $\dfrac{A\vee B \quad A\Rightarrow C \quad B\Rightarrow C}{C}$ (∨-elim-sec) $\dfrac{\Pi D:*.(A\to D)\to(B\to D)\to D \quad A\to C \quad B\to C}{C}$

Example of formal derivation to justify encoding:

(a) $\boxed{A:*}$

(b) $\boxed{B:*}$

(c) $\boxed{C:*}$

(d) $\boxed{x:(\Pi D:*.(A\to D)\to(B\to D)\to D)}$

(e) $\boxed{y:A\to C}$

(f) $\boxed{z:B\to C}$

(1) $x\,C:(A\to C)\to(B\to C)\to C$ (appl) (d),(c)

(2) $x\,C\,y:(B\to C)\to C$ (appl) (1),(e)

(3) $x\,C\,y\,z:C$ (appl) (2),(f)

Remark: We defined $\neg A$, $A\wedge B$, $A\vee B$ with free variables A, B

Could have defined $\neg :\equiv \lambda\alpha:*.(\alpha\to\bot)$

$\wedge :\equiv \lambda\alpha:*.\lambda\beta:*.\Pi\gamma:*.(\alpha\to\beta\to\gamma)\to\gamma$

$\vee :\equiv \lambda\alpha:*.\lambda\beta:*.\Pi\gamma:*.(\alpha\to\cancel{\beta}\gamma)\to(\beta\to\gamma)\to\gamma$

Needs $\lambda\omega = \lambda 2 + \lambda\underline{\omega}$

An example of propositional logic in λC

Summary: $A\Rightarrow B :\equiv A\to B$ ($\equiv \Pi x:A.B$ where $x\notin FV(B)$)

$\bot :\equiv \Pi\alpha:*\alpha$

$\neg A :\equiv A\to\bot$

$A\wedge B :\equiv \Pi C:*:(A\to B\to C)\to C$

$A\vee B :\equiv \Pi C:*:(A\to C)\to(B\to C)\to C$

$A\Leftrightarrow B :\equiv (A\to B)\wedge(B\to A)$

We are now able to formulate every statement from propositional logic

Example: $(A \vee B) \Rightarrow (\neg A \Rightarrow B)$

In λC: $\underbrace{(\Pi C:*.((A\to C)\to(B\to C)\to C))}_{A\vee B} \to \underbrace{(A\to\bot)}_{\neg A} \to B$

(a)	$\boxed{A:*}$	
(b)	$\boxed{B:*}$	
(c)	$\boxed{x : (\Pi C:*.((A\to C)\to(B\to C)\to C))}$	
(d)	$\boxed{y : A\to\bot}$	
(1)	$x B : (A\to B)\to(B\to B)\to B$	(appl) (c) (b)
(e)	$\boxed{u:A}$	
(2)	$y u : \bot$	(appl) (d) (e)
(3)	$y u B : B$	(appl) (2) (b)
(4)	$\lambda u:A.\, y u B : A\to B$	(abst) (3)
(5)	$x B(\lambda u:A.\, y u B) : (B\to B)\to B$	(appl) (1) (4)
(f)	$\boxed{v:B}$	
(6)	$v : B$	(var) (f)
(7)	$\lambda v:B.\, v : B\to B$	(abst) (6)
(8)	$x B(\lambda u:A.\, y u B)(\lambda v:B.\, v) : B$	(appl) (5) (7)
(9)	$\lambda y:A\to\bot.\, x B(\lambda u:A.\, y u B)(\lambda v:B.\, v) : (A\to\bot)\to B$	(abst) (8)
(10)	$\lambda x:(\Pi C:*.((A\to C)\to(B\to C)\to C)).\ \lambda y:A\to\bot.\, x B(\lambda u:A.\, y u B)(\lambda v:B.\, v) : (\Pi C:*.((A\to C)\to(B\to C)\to C))\to(A\to\bot)\to B$	(abst) (9)

Alternative for (c), (d) for higher order encodings of $\neg$ and $\vee$:

(c')	$\boxed{x : (\lambda\alpha:*.\ \lambda\beta:*.\ \Pi\gamma:*.((\alpha\to\gamma)\to(\beta\to\gamma)\to\gamma))AB}$	
(d')	$\boxed{y : (\lambda\alpha:*.\ (\alpha\to\bot))A}$	
(i)	$x : (\Pi\gamma:*.((A\to\gamma)\to(B\to\gamma)\to\gamma))$	(conv) (c')
(ii)	$y : A\to\bot$	(conv) (d')

<u>Classical logic in λC</u>

Up to now: constructive logic

Need law of excluded third (ET) : $A\vee\neg A$

double negation law (DN) : $\neg\neg A \Rightarrow A$

Sufficient to add only one rule, other one follows

How to? add assumption in front of context: $i_{ET} : \Pi\alpha:*.\ \alpha\vee\neg\alpha$ ("addition of an axiom")

Example: Prove (DN) assuming (ET)

(a) $\boxed{i_{ET} : \Pi\alpha:*. \alpha\vee\neg\alpha}$

(b) $\boxed{\beta : *}$

(c) $\boxed{x : \neg\neg\beta}$

(1) $i_{ET}\,\beta : \beta\vee\neg\beta$ — (appl) (a)(b)

(2) $i_{ET}\,\beta : \Pi\gamma:*. (\beta\to\gamma)\to(\neg\beta\to\gamma)\to\gamma$ — (appl) (a)(b)

(3) $i_{ET}\,\beta\beta : (\beta\to\beta)\to(\neg\beta\to\beta)\to\beta$ — (appl) (2)(b)

(4) $\lambda y:\beta. y : \beta\to\beta$ — (ad hoc)

(5) $i_{ET}\,\beta\beta(\lambda y:\beta.y) : (\neg\beta\to\beta)\to\beta$ — (appl) (3)(4)

(d) $\boxed{z : \neg\beta}$

(6) $xz : \bot$ — (appl) (c)(d)

(7) $xz\beta : \beta$ — (appl) (6)(b)

(8) $\lambda z:\neg\beta. xz\beta : \neg\beta\to\beta$ — (abst) (7)

(9) $i_{ET}\,\beta\beta(\lambda y:\beta.y)(\lambda z:\neg\beta. xz\beta) : \beta$ — (appl) ~~(a)~~ (5)(8)

(10) $\lambda x:\neg\neg\beta. i_{ET}\,\beta\beta(\lambda y:\beta.y)(\lambda z:\neg\beta.xz\beta) : \neg\neg\beta\to\beta$ — (abst) (9)

(11) $\lambda\beta:*.\lambda x:\neg\neg\beta. i_{ET}\,\beta\beta(\lambda y:\beta.y)(\lambda z:\neg\beta.xz\beta) : \Pi\beta:*.\neg\neg\beta\to\beta$ — (abst) (10)

Predicate logic in λC

Universal quantification: As in λP: $\forall x\in S: P(x) :\equiv \Pi x:S. Px$

Existential quantification: First order (only classical logic): $\exists x\in S: P(x) :\equiv \neg\forall x\in S:\neg P(x)$

Second order: $\exists x\in S: P(x) :\equiv \Pi\alpha:*.((\Pi x:S.(Px\to\alpha))\to\alpha)$

Comparison with ($\exists$-elim) and ($\exists$-intro) rules:

($\exists$-elim) $\dfrac{\exists x\in S: P(x) \qquad \forall x\in S:(P(x)\Rightarrow A)}{A}$

($\exists$-elim-sec) $\dfrac{\Pi\alpha:*.((\Pi x:S.(Px\to\alpha))\to\alpha) \qquad \Pi x:S.(Px\to A)}{A}$

Derivation in λC:

(a) $\boxed{S : *}$

(b) $\boxed{P : S\to *}$

(c) $\boxed{A : *}$

(d) $\boxed{y : \Pi\alpha:*.((\Pi x:S.(Px\to\alpha))\to\alpha)}$

(e) $\boxed{z : \Pi x:S.(Px\to A)}$

(1) $yA : (\Pi x:S.(Px\to A))\to A$ — (appl) (d)(c)

(2) $yAz : A$ — (appl) (1),(e)

$:\equiv \Pi x:S.(Px\to\alpha)$

$y : \Pi\alpha:*.\varphi(\alpha)\to\alpha$

$z : \varphi(A)$

$yA : \varphi(A)\to A$

$yAz : A$

($\exists$-intro) $\dfrac{a\in S \quad P(a)}{\exists x\in S: P(x)}$

($\exists$-intro-sec) $\dfrac{a:S \qquad Pa}{\Pi\alpha:*.((\Pi x:S.(Px\to\alpha))\to\alpha)}$

Derivation in λC:

(a) $\boxed{S : *}$

(b) $\boxed{P : S\to *}$

(c) $\boxed{a : S}$

(d) $\boxed{u : Pa}$

(1) $\lambda\alpha:*.\lambda v:(\Pi x:S.(Px\to\alpha))\to\alpha). vau :$ (ad hoc)

$\Pi\alpha:*.((\Pi x:S.(Px\to\alpha))\to\alpha)$

Could define: $\exists :\equiv \lambda S:*.\lambda P:S\to *.\Pi\alpha:*.((\Pi x:S.(Px\to\alpha))\to\alpha)$, need $\lambda P\underline{\omega}$