**Lem** $\nvdash_\lambda t : b$ with $b \in BT$

$\nvdash_\lambda e : b \to 0$

**Proof** Otherwise, we would get

$\vdash_\lambda t\{0/b\} : b\{0/b\} = 0$ $\leadsto$ contradiction

$\vdash_\lambda e\{1/b\} : (b \to 0)\{1/b\} = 1 \to 0$

$\vdash_\lambda e\{1/b\} \bullet : 0$ $\leadsto$ contradiction

**Coroll**: $\nvdash_{IPC} a$, $\nvdash_{IPC} \neg a$

with $a$ atomic

**Lem** (Disjunction property)

$\vdash_{IPC} p \vee q \iff (\vdash_{IPC} p)$ or $(\vdash_{IPC} q)$

**Proof** ($\Leftarrow$) is trivial ($\vee I1$, $\vee I2$)

($\Rightarrow$) assume $\exists t.\ \vdash_\lambda t : \tau + \sigma$

with $\ulcorner \tau \urcorner = p$, $\ulcorner \sigma \urcorner = q$

Wlog we assume that $t$ is a normal form

By normal form lemma,

$t = in1(e)$ with $\vdash_\lambda e : \tau \leadsto \vdash_{IPC} \ulcorner \tau \urcorner = p$

or

$t = in2(f)$ with $\vdash_\lambda f : \sigma \leadsto \vdash_{IPC} \ulcorner \sigma \urcorner = q$

**Coroll** If $a$ atomic, $\nvdash_{IPC} a \vee \neg a$

Hence, IPC is not the "classical" logic we are used to

**Def**: $\vdash_{CPC} = \vdash_{IPC}$ augmented with

LEM (law of excluded middle) $\vdash_{CPC} p \vee \neg p$

Coroll. $\vdash_{IPC} \subsetneq \vdash_{CPC}$

Th $\vdash_{CPC}$ is equivalent to $\vdash_{IPC}$ augmented with any one of:

1) LEM $\vdash_{CPC} p \vee \neg p \quad (\forall p)$
2) DNE double negation elimination
$\vdash_{CPC} \neg\neg p \to p \quad (\forall p)$
3) Peirce's law
$\vdash_{CPC} ((p \to q) \to p) \to p \quad (\forall p, q)$

Ex: prove it.
Ex: prove $\vdash_{IPC} p \to \neg\neg p$
$\vdash_{IPC} \neg\neg\neg p \Leftrightarrow \neg p$

Note: IPC does not prove the De Morgan's laws
$p \vee q \Leftrightarrow \neg(\neg p \wedge \neg q)$
$p \wedge q \Leftrightarrow \neg(\neg p \vee \neg q)$
(only $\to$ holds, $\leftarrow$ does not)

---

Th (Glivenko) $\vdash_{CPC} p \Leftrightarrow \vdash_{IPC} \neg\neg p$
(where $p$ is a propositional formula)

Proof ($\Leftarrow$) trivial
$\vdash_{IPC} \neg\neg p \Rightarrow \vdash_{CPC} \neg\neg p \Rightarrow \vdash_{CPC} p$

($\Rightarrow$) By induction on $\vdash_{CPC}$ we can prove $\Gamma \vdash_{CPC} p \Rightarrow \Gamma \vdash_{IPC} \neg\neg p$

A few cases:

$$(\mathrm{CPC})\ \frac{\Gamma \vdash p \quad \Gamma \vdash q}{\Gamma \vdash p \wedge q}\ \wedge I \quad \rightsquigarrow \quad \text{If } \Gamma \vdash_I \neg\neg p$$

$$\Gamma \vdash_I \neg\neg q$$

then $\Gamma \vdash_I \neg\neg(p \wedge q)$

We can assume (with some abuses of notation)

$\Gamma \vdash_\lambda t : \neg\neg p$

$\Gamma \vdash_\lambda e : \neg\neg q$

and define $f$ s.t. $\Gamma \vdash f : \neg\neg(p \wedge q)$

$$\begin{aligned} f = \lambda h_1 : \neg(p \wedge q).\ & \\ & t\ (\lambda h_2 : p. \\ & e\ (\lambda h_3 : q. \\ & h_1 \langle h_2, h_3 \rangle\ )) \end{aligned}$$

---

Another case

$\vdash_{CPC} p \vee \neg p \quad \rightsquigarrow \quad \vdash_{IPC} \neg\neg(p \vee \neg p)$

```
(fun H1: ~(p \/ ~p) =>
    H1 (@or_intror p (~p)
       (fun H2: p =>
        H1 (@or_introl p (~p) H2))))
```

$: \neg\neg(p \vee \neg p)$

Simple Types in real-word programming languages

C : $\times$ (no real $\to$, no currying)

C++ : $\to$, $\times$

C# : $\to$, $\times$

Java : $\to$, $\times$

Scala, Swift, Haskell, Ocaml : $\to$, $\times$, $+$