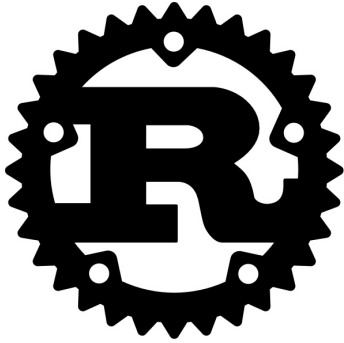


LINEAR (AND SUBSTRUCTURAL) TYPES!



CYCLONE

HOME BLOG WIKI DOWNLOAD

Cyclone is a safe dialect of C.

Cyclone is like C: it has pointers and pointer arithmetic, structs, arrays, goto, manual memory management, and C's preprocessor and syntax.

Cyclone adds features such as pattern matching, algebraic datatypes, exceptions, region-based memory management, and optional garbage collection.

Cyclone is safe: pure Cyclone programs are not vulnerable to a wide class of bugs that plague C

$\Psi \vdash \phi$

INTU.

$A \rightarrow B, A \rightarrow C, A + B \wedge C$

LINEAR

$A \multimap B, A \multimap C, A \not\wedge B \otimes C$

$A \multimap B, A \multimap C, A, A \vdash B \otimes C$

$H, H, O \multimap H_2O$

$$A, A \multimap B \otimes B \vdash B \otimes B$$

$$A, A, A \multimap B \otimes B \vdash B \otimes B \otimes A$$

$$\phi ::= A \mid \phi \multimap \psi \mid \phi \otimes \psi \mid$$

$$\phi \oplus \psi$$

$$\frac{}{\phi \vdash \phi} \text{AXIOM}$$

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \multimap \psi} \multimap I$$

$$\frac{\Gamma \vdash \phi \multimap \psi \quad \Gamma' \vdash \phi}{\Gamma, \Gamma' \vdash \psi} \multimap E$$

$$\frac{\Gamma \vdash \phi \quad \Gamma' \vdash \psi}{\Gamma, \Gamma' \vdash \phi \otimes \psi} \otimes I$$

$$\frac{\Gamma \vdash \phi \otimes \psi \quad \Gamma', \phi, \psi \vdash \chi}{\Gamma, \Gamma' \vdash \chi} \text{ } \otimes E$$

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \oplus \psi} \oplus I_1 \quad \frac{\Gamma \vdash \psi}{\Gamma \vdash \phi \oplus \psi} \oplus I_2$$

$$\frac{\Gamma \vdash \phi \oplus \psi \quad \Gamma', \phi \vdash \chi \quad \Gamma', \psi \vdash \chi}{\Gamma, \Gamma' \vdash \chi}$$

STRUCTURAL (INTU.)

 $\frac{\Gamma \vdash \phi}{\Gamma, \psi \vdash \phi}$ <p>WEAKENING</p>	$\frac{\Gamma, \Delta \vdash \phi}{\Delta, \Gamma \vdash \phi} \text{ EXCHANGE}$
 $\frac{\Gamma, \psi, \psi \vdash \phi}{\Gamma, \psi \vdash \phi}$ <p>CONTRACTION</p>	 $\frac{\Gamma, \psi, \psi \vdash \phi}{\Gamma, \psi \vdash \phi}$ <p>CONTRACTION</p>

LINEAR

LINEAR

$$e ::= x \mid \lambda x: \tau. e \mid e_1 e_2$$

$$\tau ::= b \mid \tau_1 \rightarrow \tau_2$$

\downarrow
 int/unit

$$\underbrace{x: \tau \vdash x: \tau}_{\{(x, \tau)\}}$$

sys.stdin.read()

let p = malloc 4 in
 store p ((load p) + 38);
 free p