From CBV to LLVM via CBPV

ANONYMOUS AUTHOR(S)

Additional Key Words and Phrases: keyword1, keyword2, keyword3

ACM Reference Format:
Anonymous Author(s). 2018. From CBV to LLVM via CBPV. *Proc. ACM Program. Lang.* 1, CONF, Article 1 (January 2018), 25 pages.

1 STLC

1.1 Syntax

$$
\begin{array}{llll}
\text{Types} & \tau & ::= & \mathrm{Int} \mid \mathrm{Bool} \mid \tau \times \tau \mid \tau + \tau \mid \tau \to \tau \\
\\
\text{Module Types} & \Psi & ::= & \overline{\ell \mapsto \tau} \\
\\
\text{Values} & v & ::= & x \mid \ell \mid n \mid \mathrm{true} \mid \mathrm{false} \mid \mathrm{inl}\ v \mid \mathrm{inr}\ v \mid (v, v) \mid \lambda x.\ e \\
\\
\text{Expressions} & e & ::= & \mathrm{let}\ (x, x) = e\ \mathrm{in}\ e \\
 & & \mid & \mathrm{if}\ e\ \mathrm{then}\ e\ \mathrm{else}\ e \mid \mathrm{case}\ e\ \{\mathrm{inl}\ x \Rightarrow e;\ \mathrm{inr}\ x \Rightarrow e\} \\
 & & \mid & v \mid e\ e \mid \mathrm{inl}\ e \mid \mathrm{inr}\ e \mid (e, e) \\
\\
\text{Modules} & \mathit{mod} & ::= & \overline{\ell \mapsto e}
\end{array}
$$

1.2 Typing rules

(AA: Separate the value, computation, and module rules and put an fbox above each to show shape of judgment. After modules, add a whole program typing rule.)

$$\boxed{\Psi; \Gamma \vdash e : \tau}$$

2018. 2475-1421/2018/1-ART1 $15.00
https://doi.org/

$$\frac{x : \tau \in \Gamma}{\Psi; \Gamma \vdash x : \tau} \qquad \frac{\ell \mapsto \tau \in \Psi}{\Psi; \Gamma \vdash \ell : \tau} \qquad \frac{}{\Psi; \Gamma \vdash n : \mathrm{Int}} \qquad \frac{}{\Psi; \Gamma \vdash \mathrm{true} : \mathrm{Bool}} \qquad \frac{}{\Psi; \Gamma \vdash \mathrm{false} : \mathrm{Bool}}$$

$$\frac{\Psi; \Gamma \vdash e : \tau}{\Psi; \Gamma \vdash \mathrm{inl}\ v : \tau + \tau'} \qquad \frac{\Psi; \Gamma \vdash e : \tau}{\Psi; \Gamma \vdash \mathrm{inr}\ v : \tau' + \tau} \qquad \frac{\Psi; \Gamma \vdash e_1 : \tau_1 \qquad \Psi; \Gamma \vdash e_2 : \tau_2}{\Psi; \Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2}$$

$$\frac{\Psi; \Gamma, x : \tau \vdash e : \tau'}{\Psi; \Gamma \vdash \lambda x.\ e : \tau \to \tau'} \qquad \frac{\Psi; \Gamma \vdash e_1 : \tau_1 \times \tau_2 \qquad \Psi; \Gamma, x : \tau_1, y : \tau_2 \vdash e_2 : \tau}{\Psi; \Gamma \vdash \mathrm{let}\ (x, y) = e_1\ \mathrm{in}\ e_2 : \tau}$$

$$\frac{\Psi; \Gamma \vdash e_c : \mathrm{Bool} \qquad \Psi; \Gamma \vdash e_t : \tau \qquad \Psi; \Gamma \vdash e_f : \tau}{\Psi; \Gamma \vdash \mathrm{if}\ e_c\ \mathrm{then}\ e_t\ \mathrm{else}\ e_f : \tau}$$

$$\frac{\Psi; \Gamma \vdash e_s : \tau_l \times \tau_r \qquad \Psi; \Gamma, x : \tau_l \vdash e_l : \tau \qquad \Psi; \Gamma, y : \tau_r \vdash e_r : \tau}{\Psi; \Gamma \vdash \mathrm{case}\ e_s\ \{\mathrm{inl}\ x \Rightarrow e_l;\ \mathrm{inr}\ y \Rightarrow e_r\} : \tau}$$

$$\frac{\Psi; \Gamma \vdash e_1 : \tau \to \tau' \qquad \Psi; \Gamma \vdash e_2 : \tau}{\Psi; \Gamma \vdash e_1\ e_2 : \tau'}$$

$$\boxed{\vdash mod : \Psi}$$

$$\frac{\forall i, \ell_i \mapsto \tau_i \in \Psi \wedge \Psi; \bullet \vdash e_i : \tau_i}{\vdash \ell_1 \mapsto e_1, \ldots, \ell_n \mapsto e_n : \Psi}$$

2 HIGH-LEVEL CBPV

2.1 HL-CBPV: Syntax

$$\begin{array}{llcl}
\text{Value Types} & A & ::= & \mathrm{Int} \mid \mathrm{Bool} \mid A \times A \mid A + A \mid U\underline{B} \\
\text{Computation Types} & \underline{B} & ::= & A \to \underline{B} \mid FA \\
\text{Module Types} & \Psi & ::= & \overline{\ell \mapsto \underline{B}} \\
\text{Values} & V & ::= & x \mid n \mid \mathrm{true} \mid \mathrm{false} \mid \mathrm{inl}\ v \mid \mathrm{inr}\ v \mid (v, v) \mid \mathrm{thunk}\ M \mid \mathrm{thunk}\ \ell \\
\text{Computations} & M & ::= & \mathrm{force}\ V \mid \mathrm{ret}\ V \mid x \leftarrow M;\ N \mid M\ V \mid \lambda(x : A).\ M \\
 & & \mid & \mathrm{let}\ (x, x) = V\ \mathrm{in}\ M \\
 & & \mid & \mathrm{if}\ V\ \mathrm{then}\ M\ \mathrm{else}\ M \\
 & & \mid & \mathrm{case}\ V\ \{\mathrm{inl}\ x \Rightarrow M;\ \mathrm{inr}\ x \Rightarrow M\} \\
\text{Modules} & mod & ::= & \overline{\ell \mapsto M}
\end{array}$$

2.2 HL-CBPV: Typing rules

$$\boxed{\Psi; \Gamma \vdash^{\mathsf{v}} V : A}$$

$$\frac{(x:A)\in\Gamma}{\Psi;\Gamma\vdash^{\mathsf{v}} x:A}\qquad\frac{}{\Psi;\Gamma\vdash^{\mathsf{v}} n:\mathrm{Int}}\qquad\frac{}{\Psi;\Gamma\vdash^{\mathsf{v}} \mathrm{true}:\mathrm{Bool}}\qquad\frac{}{\Psi;\Gamma\vdash^{\mathsf{v}} \mathrm{false}:\mathrm{Bool}}$$

$$\frac{\Psi;\Gamma\vdash^{\mathsf{v}} V:A}{\Psi;\Gamma\vdash^{\mathsf{v}} \mathrm{inl}\ V:A+A'}\qquad\frac{\Psi;\Gamma\vdash^{\mathsf{v}} V:A}{\Psi;\Gamma\vdash^{\mathsf{v}} \mathrm{inr}\ V:A'+A}\qquad\frac{\Psi;\Gamma\vdash^{\mathsf{v}} V_l:A_l\qquad\Psi;\Gamma\vdash^{\mathsf{v}} V_r:A_r}{\Psi;\Gamma\vdash^{\mathsf{v}} (V_l,V_r):A_l\times A_r}$$

$$\frac{\Psi;\Gamma\vdash^{\mathsf{m}} M:\underline{B}}{\Psi;\Gamma\vdash^{\mathsf{v}} \mathrm{thunk}\ M:U\underline{B}}\qquad\frac{\ell\mapsto\underline{B}\in\Psi}{\Psi;\Gamma\vdash^{\mathsf{v}} \mathrm{thunk}\ \ell:U\underline{B}}$$

$$\boxed{\Psi;\Gamma\vdash^{\mathsf{v}} M:\underline{B}}$$

$$\frac{\Psi;\Gamma\vdash^{\mathsf{v}} V:U\underline{B}}{\Psi;\Gamma\vdash^{\mathsf{m}} \mathrm{force}\ V:\underline{B}}\qquad\frac{\Psi;\Gamma\vdash^{\mathsf{v}} V:A}{\Psi;\Gamma\vdash^{\mathsf{m}} \mathrm{ret}\ V:FA}\qquad\frac{\Psi;\Gamma\vdash^{\mathsf{m}} M:FA\qquad\Psi;\Gamma,x:A\vdash^{\mathsf{m}} N:\underline{B}}{\Psi;\Gamma\vdash^{\mathsf{m}} x\leftarrow M;\ N:\underline{B}}$$

$$\frac{\Psi;\Gamma\vdash^{\mathsf{v}} V:A\qquad\Psi;\Gamma\vdash^{\mathsf{v}} M:A\to\underline{B}}{\Psi;\Gamma\vdash^{\mathsf{m}} M\ V:\underline{B}}\qquad\frac{\Psi;\Gamma,x:A\vdash^{\mathsf{m}} M:\underline{B}}{\Psi;\Gamma\vdash^{\mathsf{m}} \lambda(x:A).\,M:A\to\underline{B}}$$

$$\frac{\Psi;\Gamma\vdash^{\mathsf{m}} V:A_l\times A_r\qquad\Psi;\Gamma,x:A_l,y:A_r\vdash^{\mathsf{m}} M:\underline{B}}{\Psi;\Gamma\vdash^{\mathsf{m}} \mathrm{let}\ (x,y)=V\ \mathrm{in}\ M:\underline{B}}$$

$$\frac{\Psi;\Gamma\vdash^{\mathsf{v}} V:\mathrm{Bool}\qquad\Psi;\Gamma\vdash^{\mathsf{m}} M_t:\underline{B}\qquad\Psi;\Gamma\vdash^{\mathsf{m}} M_f:\underline{B}}{\Psi;\Gamma\vdash^{\mathsf{m}} \mathrm{if}\ V\ \mathrm{then}\ M_t\ \mathrm{else}\ M_f:\underline{B}}$$

$$\frac{\Psi;\Gamma\vdash^{\mathsf{v}} V:A_l\times A_r\qquad\Psi;\Gamma,x:A_l\vdash^{\mathsf{m}} M_l:\underline{B}\qquad\Psi;\Gamma,y:A_r\vdash^{\mathsf{m}} M_r:\underline{B}}{\Psi;\Gamma\vdash^{\mathsf{m}} \mathrm{case}\ V\ \{\mathrm{inl}\ x\Rightarrow M_l;\ \mathrm{inr}\ y\Rightarrow M_r\}:\underline{B}}$$

$$\boxed{\vdash mod:\Psi}$$

$$\frac{\forall i,\ell_i\mapsto\underline{B}_i\in\Psi\wedge\Psi;\bullet\vdash^{\mathsf{m}} M_i:\underline{B}_i}{\vdash\ell_1\mapsto M_1,\ldots,\ell_n\mapsto M_n:\Psi}$$

2.3 STLC to HL-CBPV: Type translation

$$\mathrm{Int}^+=\mathrm{Int}\qquad\mathrm{Bool}^+=\mathrm{Bool}\qquad(\tau_1\times\tau_2)^+={\tau_1}^+\times{\tau_2}^+\qquad(\tau_1+\tau_2)^+={\tau_1}^++{\tau_2}^+$$

$$(\tau_1\to\tau_2)^+=U({\tau_1}^+\to F{\tau_2}^+)$$

2.4 STLC to HL-CBPV: Term translation

$\boxed{\Psi; \Gamma \vdash e : \tau \rightsquigarrow M}$

$$\frac{(x : \tau) \in \Gamma}{\Psi; \Gamma \vdash x : \tau \rightsquigarrow \mathrm{ret}\ x} \qquad \frac{}{\Psi; \Gamma \vdash n : \mathrm{Int} \rightsquigarrow \mathrm{ret}\ n} \qquad \frac{}{\Psi; \Gamma \vdash \mathrm{true} : \mathrm{Bool} \rightsquigarrow \mathrm{ret\ true}}$$

$$\frac{}{\Psi; \Gamma \vdash \mathrm{false} : \mathrm{Bool} \rightsquigarrow \mathrm{ret\ false}} \qquad \frac{\mathrm{fresh}(x) \qquad \Psi; \Gamma \vdash e : \tau \rightsquigarrow M}{\Psi; \Gamma \vdash \mathrm{inl}\ e : \tau + \tau' \rightsquigarrow x \leftarrow M;\ \mathrm{ret\ inl}\ x}$$

$$\frac{\mathrm{fresh}(x) \qquad \Psi; \Gamma \vdash e : \tau \rightsquigarrow M}{\Psi; \Gamma \vdash \mathrm{inr}\ e : \tau + \tau' \rightsquigarrow x \leftarrow M;\ \mathrm{ret\ inr}\ x}$$

$$\frac{\mathrm{fresh}(x_l) \qquad \mathrm{fresh}(x_r) \qquad \Psi; \Gamma \vdash e_l : \tau_l \rightsquigarrow M_l \qquad \Psi; \Gamma \vdash e_r : \tau_r \rightsquigarrow M_r}{\Psi; \Gamma \vdash (e_l, e_r) : \tau_l \times \tau_r \rightsquigarrow x_l \leftarrow M_l;\ x_r \leftarrow M_r;\ \mathrm{ret}\ (x_l, x_r)}$$

$$\frac{\mathrm{fresh}(f) \qquad \mathrm{fresh}(x) \qquad \Psi; \Gamma \vdash e_1 : \tau \to \tau' \rightsquigarrow M_1 \qquad \Psi; \Gamma \vdash e_2 : \tau \rightsquigarrow M_2}{\Psi; \Gamma \vdash e_1\ e_2 : \tau' \rightsquigarrow f \leftarrow M_1;\ x \leftarrow M_2;\ (\mathrm{force}\ f)\ x}$$

$$\frac{\Psi; \Gamma, x : \tau' \vdash e : \tau \rightsquigarrow M}{\Psi; \Gamma \vdash \lambda x.\, e : \tau' \to \tau \rightsquigarrow \mathrm{ret\ thunk}\ \lambda x.\, M} \qquad \frac{\ell \mapsto \tau \in \Psi}{\Psi; \Gamma \vdash \ell : \tau \rightsquigarrow \mathrm{ret\ thunk}\ \ell}$$

$$\frac{\mathrm{fresh}(z) \qquad \Psi; \Gamma \vdash e_1 : \tau_l \times \tau_r \rightsquigarrow M_1 \qquad \Psi; \Gamma, x : \tau_l, y : \tau_r \vdash e_2 : \tau \rightsquigarrow M_2}{\Psi; \Gamma \vdash \mathrm{let}\ (x, y) = e_1\ \mathrm{in}\ e_2 : \tau \rightsquigarrow z \leftarrow M_1;\ \mathrm{let}\ (x, y) = z\ \mathrm{in}\ M_2}$$

$$\frac{\mathrm{fresh}(x) \qquad \Psi; \Gamma \vdash e_c : \mathrm{Bool} \rightsquigarrow M_c \qquad \Psi; \Gamma \vdash e_t : \tau \rightsquigarrow M_t \qquad \Psi; \Gamma \vdash e_t : \tau \rightsquigarrow M_f}{\Psi; \Gamma \vdash \mathrm{if}\ e_c\ \mathrm{then}\ e_t\ \mathrm{else}\ e_f : \tau \rightsquigarrow x \leftarrow M_c;\ \mathrm{if}\ x\ \mathrm{then}\ M_t\ \mathrm{else}\ M_f}$$

$$\frac{\mathrm{fresh}(z) \qquad \Psi; \Gamma \vdash e_s : \tau_l \times \tau_r \rightsquigarrow M_s \qquad \Psi; \Gamma, x : \tau_l \vdash e_l : \tau \rightsquigarrow M_l \qquad \Psi; \Gamma, y : \tau_r \vdash e_r : \tau \rightsquigarrow M_r}{\Psi; \Gamma \vdash \mathrm{case}\ e_s\ \{\mathrm{inl}\ x \Rightarrow e_l;\ \mathrm{inr}\ y \Rightarrow e_r\} : \tau \rightsquigarrow z \leftarrow M_s;\ \mathrm{case}\ z\ \{\mathrm{inl}\ x \Rightarrow y;\ \mathrm{inr}\ M_l \Rightarrow M_r\}}$$

Then if $\Psi; \Gamma \vdash e : \tau \rightsquigarrow M$, $\Psi; \Gamma \vdash e : \tau$ in STLC, and $\Psi^+; \Gamma^+ \vdash^{\mathrm{m}} M : F\tau^+$.

(AA: The above doesn't make sense. In the second condition should the x be e? Also, we need to translate types in Ψ and Γ.)

2.5 STLC to HL-CBPV: Module translation

$$\frac{\forall i, e_i \mapsto \tau_i \in \Psi \wedge \Psi; \bullet \vdash e_i : \tau_i \rightsquigarrow M_i}{\vdash \ell_1 \mapsto e_1, \ldots, \ell_n \mapsto e_n : \Psi \rightsquigarrow \ell_1 \mapsto M_1, \ldots, \ell_n \mapsto M_n}$$

3 CLOSURE CONVERSION

3.1 CC-CBPV: Syntax

Value Types	A	$::=$	$\mathrm{Int} \mid \mathrm{Bool} \mid A \times A \mid A + A \mid U\underline{B}$
		$\mid$	$\exists x.\, A$

$$
\begin{array}{llcl}
\text{Computation Types} & \underline{B} & ::= & A \to \underline{B} \mid FA \\
\text{Module Types} & \Psi & ::= & \overline{\ell \mapsto \underline{B}} \\
\text{Values} & V & ::= & x \mid n \mid \text{true} \mid \text{false} \mid \text{inl } v \mid \text{inr } v \mid (v, v) \mid \text{thunk } M \mid \text{thunk } \ell \\
 & & \mid & \text{pack } (A, V) \text{ as } \exists x.\, A \\
\text{Computations} & M & ::= & \text{force } V \mid \text{ret } V \mid x \leftarrow M;\ N \mid M\ V \mid \lambda(x : A).\, M \\
 & & \mid & \text{let } (x, x) = V \text{ in } M \\
 & & \mid & \text{if } V \text{ then } M \text{ else } M \\
 & & \mid & \text{case } V\ \{\text{inl } x \Rightarrow M;\ \text{inr } x \Rightarrow M\} \\
 & & \mid & \text{unpack } (\alpha, x) = V \text{ in } M \\
\text{Modules} & mod & ::= & \overline{\ell \mapsto M}
\end{array}
$$

3.2 CC-CBPV: Typing rules

$$\boxed{\Psi; \Gamma \vdash^{\mathsf{v}} V : A}$$

$$
\frac{(x : A) \in \Gamma}{\Psi; \Gamma \vdash^{\mathsf{v}} x : A}
\qquad
\frac{}{\Psi; \Gamma \vdash^{\mathsf{v}} n : \text{Int}}
\qquad
\frac{}{\Psi; \Gamma \vdash^{\mathsf{v}} \text{true} : \text{Bool}}
\qquad
\frac{}{\Psi; \Gamma \vdash^{\mathsf{v}} \text{false} : \text{Bool}}
$$

$$
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V : A}{\Psi; \Gamma \vdash^{\mathsf{v}} \text{inl } V : A + A'}
\qquad
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V : A}{\Psi; \Gamma \vdash^{\mathsf{v}} \text{inr } V : A' + A}
\qquad
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V_l : A_r \qquad \Psi; \Gamma \vdash^{\mathsf{v}} V_r : A_r}{\Psi; \Gamma \vdash^{\mathsf{v}} (V_l, V_r) : A_l \times A_r}
$$

$$
\frac{\Psi; \bullet \vdash^{\mathsf{m}} M : \underline{B}}{\Psi; \Gamma \vdash^{\mathsf{v}} \text{thunk } M : U\underline{B}}
\qquad
\frac{\ell \mapsto \underline{B} \in \Psi}{\Psi; \Gamma \vdash^{\mathsf{v}} \text{thunk } \ell : U\underline{B}}
\qquad
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V : A'[A/x]}{\Psi; \Gamma \vdash^{\mathsf{v}} \text{pack } (A, V) \text{ as } \exists x.\, A' : \exists x.\, A'}
$$

$$\boxed{\Psi; \Gamma \vdash^{\mathsf{m}} M : \underline{B}}$$

$$
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V : U\underline{B}}{\Psi; \Gamma \vdash^{\mathsf{m}} \text{force } V : \underline{B}}
\qquad
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V : A}{\Psi; \Gamma \vdash^{\mathsf{m}} \text{ret } V : FA}
$$

$$
\frac{\Psi; \Gamma \vdash^{\mathsf{m}} M : FA \qquad \Psi; \Gamma, (x : A) \vdash^{\mathsf{m}} N : \underline{B}}{\Psi; \Gamma \vdash^{\mathsf{m}} x \leftarrow M;\ N : \underline{B}}
\qquad
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V : A \qquad \Psi; \Gamma \vdash^{\mathsf{m}} M : A \to \underline{B}}{\Psi; \Gamma \vdash^{\mathsf{m}} M\ V : \underline{B}}
$$

$$
\frac{\Psi; \Gamma, (x : A) \vdash^{\mathsf{m}} M : \underline{B}}{\Psi; \Gamma \vdash^{\mathsf{m}} \lambda(x : A).\, M : A \to \underline{B}}
\qquad
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V : A_l \times A_r \qquad \Psi; \Gamma, (x : A_l), (y : A_r) \vdash^{\mathsf{m}} M : \underline{B}}{\Psi; \Gamma \vdash^{\mathsf{m}} \text{let } (x, y) = V \text{ in } M : \underline{B}}
$$

$$
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V : \text{Bool} \qquad \Psi; \Gamma \vdash^{\mathsf{m}} M_t : \underline{B} \qquad \Psi; \Gamma \vdash^{\mathsf{m}} M_f : \underline{B}}{\Psi; \Gamma \vdash^{\mathsf{m}} \text{if } V \text{ then } M_t \text{ else } M_f : \underline{B}}
$$

$$
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V : A_l + A_r \qquad \Psi; \Gamma, (x : A_l) \vdash^{\mathsf{m}} M_l : \underline{B} \qquad \Psi; \Gamma, (y : A_r) \vdash^{\mathsf{m}} M_r : \underline{B}}{\Psi; \Gamma \vdash^{\mathsf{m}} \text{case } V\ \{\text{inl } x \Rightarrow M_l;\ \text{inr } y \Rightarrow M_r\} : \underline{B}}
$$

$$
\frac{\Psi; \Gamma \vdash^{\mathsf{v}} V : \exists \beta.\, A \qquad \Psi; \Gamma, (x : A[\alpha/\beta]) \vdash^{\mathsf{m}} M : \underline{B}}{\Psi; \Gamma \vdash^{\mathsf{m}} \text{unpack } (\alpha, x) = V \text{ in } M : \underline{B}}
$$

$$\boxed{\vdash mod : \Psi}$$

$$\frac{\forall i, \ell_i \mapsto \underline{B}_i \in \Psi \land \Psi; \bullet \vdash^{\mathrm{m}} M_i : \underline{B}_i}{\vdash \ell_1 \mapsto M_1, \ldots, \ell_n \mapsto M_n : \Psi}$$

3.3 HL-CBPV to CC-CBPV: Type translation

$$\mathrm{Int}^+ = \mathrm{Int} \qquad \mathrm{Bool}^+ = \mathrm{Bool} \qquad (A \times A)^+ = A^+ \times A^+ \qquad (A + A)^+ = A^+ + A^+$$

$$(U\underline{B})^+ = \exists x.\, (U(x \to \underline{B}^+)) \times x \qquad (A \to \underline{B})^+ = A^+ \to \underline{B}^+ \qquad (FA)^+ = FA^+$$

3.4 HL-CBPV to CC-CBPV: Value translation

$$\frac{(x : A) \in \Gamma}{\Psi; \Gamma \vdash x : A \rightsquigarrow x} \qquad \frac{}{\Psi; \Gamma \vdash n : \mathrm{Int} \rightsquigarrow n} \qquad \frac{}{\Psi; \Gamma \vdash \mathrm{true} : \mathrm{Bool} \rightsquigarrow \mathrm{true}}$$

$$\frac{}{\Psi; \Gamma \vdash \mathrm{false} : \mathrm{Bool} \rightsquigarrow \mathrm{false}} \qquad \frac{\Psi; \Gamma \vdash V : A \rightsquigarrow V'}{\Psi; \Gamma \vdash \mathrm{inl}\ V : A + A' \rightsquigarrow \mathrm{inl}\ V'}$$

$$\frac{\Psi; \Gamma \vdash V : A \rightsquigarrow V'}{\Psi; \Gamma \vdash \mathrm{inr}\ V : A' + A \rightsquigarrow \mathrm{inr}\ V'} \qquad \frac{\Psi; \Gamma \vdash V_l : A_l \rightsquigarrow V_l \qquad \Psi; \Gamma \vdash V_r : A_r \rightsquigarrow V_r'}{\Psi; \Gamma \vdash (V_l, V_r) : A_l \times A_r \rightsquigarrow (V_l', V_r')}$$

$$\frac{\Psi; \Gamma \vdash M : \underline{B} \rightsquigarrow M' \qquad \overline{y} = \mathsf{FV}(M)}{\Psi; \Gamma \vdash \mathrm{thunk}\ M : U\underline{B} \rightsquigarrow \begin{array}{l} \mathrm{pack}\ (\Pi\,\overline{\Gamma(y)}^+, (\mathrm{thunk}\ \begin{array}{l}\lambda(z : \Pi\,\overline{\Gamma(y)^+}).\\ \ \mathrm{let}\ (y_1, \ldots, y_n) = z\ \mathrm{in}\ M'\end{array}, (y_1, \ldots, y_n))) \\ \quad \mathrm{as}\ \exists x.\, (U(x \to \underline{B})) \times x \end{array}}$$

$$\frac{\ell \mapsto \underline{B} \in \Psi}{\Psi; \Gamma \vdash \mathrm{thunk}\ \ell : U\underline{B} \rightsquigarrow \mathrm{thunk}\ \ell}$$

3.5 HL-CBPV to CC-CBPV: Computation translation

$$\frac{\Psi; \Gamma \vdash V : U\underline{B} \rightsquigarrow V'}{\Psi; \Gamma \vdash \mathrm{force}\ V : \underline{B} \rightsquigarrow \begin{array}{l} \mathrm{unpack}\ (\alpha, V') = V'\ \mathrm{in} \\ \mathrm{let}\ (f, ys) = V'\ \mathrm{in} \\ \mathrm{force}\ f\ ys \end{array}} \qquad \frac{\Psi; \Gamma \vdash V : A \rightsquigarrow V'}{\Psi; \Gamma \vdash \mathrm{ret}\ V : FA \rightsquigarrow \mathrm{ret}\ V'}$$

$$\frac{\Psi; \Gamma \vdash M : FA \rightsquigarrow M' \qquad \Psi; \Gamma, (x : A) \vdash N : \underline{B} \rightsquigarrow N'}{\Psi; \Gamma \vdash x \leftarrow M;\ N : \underline{B} \rightsquigarrow x \leftarrow M';\ N'}$$

$$\frac{\Psi; \Gamma \vdash V : A \rightsquigarrow V' \Psi; \Gamma \vdash M : A \to \underline{B} \rightsquigarrow M'}{\Psi; \Gamma \vdash M\ V : \underline{B} \rightsquigarrow M'\ V'} \qquad \frac{\Psi; \Gamma, (x : A) \vdash M : \underline{B} \rightsquigarrow M'}{\Psi; \Gamma \vdash \lambda(x : A).\ M : A \to \underline{B} \rightsquigarrow \lambda(x : A^{+}).\ M'}$$

$$\frac{\Psi; \Gamma \vdash V : A_l \times A_r \rightsquigarrow V' \qquad \Psi; \Gamma, (x : A_l), (y : A_r) \vdash M : \underline{B} \rightsquigarrow M'}{\Psi; \Gamma \vdash \mathrm{let}\ (x, y) = V\ \mathrm{in}\ M : \underline{B} \rightsquigarrow \mathrm{let}\ (x, y) = V'\ \mathrm{in}\ M'}$$

$$\frac{\Psi; \Gamma \vdash V : \mathrm{Bool} \rightsquigarrow V' \qquad \Psi; \Gamma \vdash M_t : \underline{B} \rightsquigarrow M_t' \qquad \Psi; \Gamma \vdash M_f : \underline{B} \rightsquigarrow M_f'}{\Psi; \Gamma \vdash \mathrm{if}\ V\ \mathrm{then}\ M_t\ \mathrm{else}\ M_f : \underline{B} \rightsquigarrow \mathrm{if}\ V'\ \mathrm{then}\ M_t'\ \mathrm{else}\ M_f'}$$

$$\frac{\Psi; \Gamma \vdash V : A_l \times A_r \rightsquigarrow V' \qquad \Psi; \Gamma, (x : A_l) \vdash M_l : \underline{B} \rightsquigarrow M_l' \qquad \Psi; \Gamma, (x : A_r) \vdash M_r : \underline{B} \rightsquigarrow M_r'}{\Psi; \Gamma \vdash \mathrm{case}\ V\ \{\mathrm{inl}\ x \Rightarrow M_l;\ \mathrm{inr}\ y \Rightarrow M_r\} : \underline{B} \rightsquigarrow \mathrm{case}\ V'\ \{\mathrm{inl}\ x \Rightarrow M_l';\ \mathrm{inr}\ y \Rightarrow M_r'\}}$$

$$\frac{\Psi; \Gamma \vdash V : \exists \beta.\ A \rightsquigarrow V' \qquad \Psi; \Gamma, (x : A\alpha/\beta) \vdash M : \underline{B} \rightsquigarrow M'}{\Psi; \Gamma \vdash \mathrm{unpack}\ (\alpha, x) = V\ \mathrm{in}\ M : \underline{B} \rightsquigarrow \mathrm{unpack}\ (\alpha, x) = V'\ \mathrm{in}\ M'}$$

3.6 HL-CBPV to CC-CBPV: Module translation

$$\frac{\forall i, \ell_i \mapsto \underline{B}_i \in \Psi \wedge \Psi; \bullet \vdash M_i : \underline{B}_i \rightsquigarrow M_i'}{\vdash \ell_1 \mapsto M_1, \ldots, \ell_n \mapsto M_n : \Psi \rightsquigarrow \ell_1 \mapsto M_1', \ldots, \ell_n \mapsto M_n'}$$

4 HEAP ALLOCATION

4.1 HA-CBPV: Syntax

Value Types	A	$::=$	$\mathrm{Int} \mid \mathrm{Bool} \mid \mathrm{ptr}_V(\overline{(n, A)}) \mid \mathrm{ptr}_M(\underline{B})$
		$\mid$	$\exists x.\ A$
Computation Types	$\underline{B}$	$::=$	$A \to \underline{B} \mid FA$
Module Types	Ψ	$::=$	$\overline{\ell \mapsto \underline{B}}$
Values	V	$::=$	$x \mid n \mid \mathrm{true} \mid \mathrm{false} \mid \mathrm{thunk}\ \ell$
		$\mid$	$\mathrm{pack}\ (A, V)\ \mathrm{as}\ \exists x.\ A$
Computations	M	$::=$	$\mathrm{force}\ V \mid \mathrm{ret}\ V \mid x \leftarrow M;\ N \mid M\ V \mid \lambda(x : A).\ M$
		$\mid$	$\mathrm{let}\ x = \mathrm{alloc}_n(V_1, \ldots, V_n)\ \mathrm{in}\ M \mid \mathrm{let}\ x = \mathrm{prj}_n(V)\ \mathrm{in}\ M$
		$\mid$	$\mathrm{if}\ V\ \mathrm{then}\ M\ \mathrm{else}\ M$

$$
\begin{array}{llll}
 & & | & \text{case } V \ \{\overline{n(x) \Rightarrow M;}\} \\
 & & | & \text{unpack } (\alpha, x) = V \text{ in } M \\
\text{Modules} & mod & ::= & \overline{\ell \mapsto M} \\
\text{Heap allocations} & \mathcal{A}_\mathsf{h} & ::= & \bullet \mid (x \mapsto_n \overline{V}), \mathcal{A}_\mathsf{h} \\
\text{Code allocations} & \mathcal{A}_\mathsf{c} & ::= & \bullet \mid (\ell \mapsto M), \mathcal{A}_\mathsf{c}
\end{array}
$$

The index of the $\mathrm{ptr}_V()$ type deserves some explanation. It is intended that at the moment (in order to delay the problem of sum types), the heap contain indexed sums of indexed products. The types therefore capture a set of integer tags (the possible cases), each of which is associated with a list of types (the contents of that case). The projection operators only work on a pointer value whose type contains only one possible case; the case operator converts a pointer value whose type has many cases to one which has only one case.

Note todo: using sum-of-products for things in the heap at the moment, since we kind of want to avoid at least put off figuring out how to do appropriate structuring of sums. (a) that needs to be done at some point, and (b) there may be some semantic implications of using SOP?

Note todo: does some kind of switch to the projection product need to happen here? I think not, because indexing a product by a selector is perhaps effect-free at the appropriate level, since only values are stored within. (that is, it's not possible as with the usual pattern match product for an index operation to run arbitrary computation; and it's similarly not possible to use infinitary things here.) Storing values rather than computations feels like the distinction between the pattern-match and projection products, even though semantically now that we're moving the product to explicit memory cells, we do need to project one at a time.

4.2 HA-CBPV: Typing rules

$$\boxed{\Psi; \Gamma \vdash V : A}$$

$$\frac{(x, A) \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{}{\Psi; \Gamma \vdash n : \mathrm{Int}} \qquad \frac{}{\Psi; \Gamma \vdash \mathrm{true} : \mathrm{Bool}} \qquad \frac{}{\Psi; \Gamma \vdash \mathrm{false} : \mathrm{Bool}}$$

$$\frac{\Psi; \Gamma \vdash \Psi[\ell] : \underline{B}}{\Psi; \Gamma \vdash \mathrm{thunk}\ \ell : \mathrm{ptr}_M(\underline{B})} \qquad \frac{\Psi; \Gamma \vdash V : A'[A/x]}{\Psi; \Gamma \vdash \mathrm{pack}\ (A, V)\ \mathrm{as}\ \exists x.\, A' : \exists x.\, A'}$$

$$\boxed{\Psi; \Gamma \vdash M : \underline{B}}$$

$$\frac{\Psi; \Gamma \vdash V : \mathrm{ptr}_M(\underline{B})}{\Psi; \Gamma \vdash \mathrm{force}\ V : \underline{B}} \qquad \frac{\Psi; \Gamma \vdash V : A}{\Psi; \Gamma \vdash \mathrm{ret}\ V : FA} \qquad \frac{\Psi; \Gamma \vdash M : FA \qquad \Psi; \Gamma, (x : A) \vdash N : \underline{B}}{\Psi; \Gamma \vdash x \leftarrow M;\ N : \underline{B}}$$

$$\frac{\Psi; \Gamma \vdash M : A \to \underline{B} \qquad \Psi; \Gamma \vdash V : A}{\Psi; \Gamma \vdash M\ V : \underline{B}} \qquad \frac{\Psi; \Gamma, (x : A) \vdash M : \underline{B}}{\Psi; \Gamma \vdash \lambda(x : A).\ M : A \to \underline{B}}$$

$$\frac{\Psi; \Gamma \vdash V_i : A_i \qquad \Psi; \Gamma, (x : \mathrm{ptr}_V((n, A_1, \ldots, A_n), \ldots)) \vdash M : \underline{B}}{\Psi; \Gamma \vdash \mathrm{let}\ x = \mathrm{alloc}_n(V_1, \ldots, V_n)\ \mathrm{in}\ M : \underline{B}}$$

$$\frac{\Psi; \Gamma \vdash V : \mathrm{ptr}_V((n, A_1, \ldots, A_n)) \qquad \Psi; \Gamma, (x : A_i) \vdash M : \underline{B}}{\Psi; \Gamma \vdash \mathrm{let}\ x = \mathrm{prj}_i(V)\ \mathrm{in}\ M : \underline{B}}$$

$$\frac{\Psi; \Gamma \vdash V : \mathrm{Bool} \qquad \Psi; \Gamma \vdash M_1 : \underline{B} \qquad \Psi; \Gamma \vdash M_2 : \underline{B}}{\Psi; \Gamma \vdash \mathrm{if}\ V\ \mathrm{then}\ M_1\ \mathrm{else}\ M_2 : \underline{B}}$$

$$\frac{\Psi; \Gamma \vdash V : \mathrm{ptr}_V((i_1, \overline{A_1}), \ldots, (i_n, \overline{A_n})) \qquad \Psi; \Gamma, (x_j : \mathrm{ptr}_V((i_j, \overline{A_j}))) \vdash M_n : \underline{B}}{\Psi; \Gamma \vdash \mathrm{case}\ V\ \{i_1(x_1) \Rightarrow M_1;\ , \ldots, i_n(x_n) \Rightarrow M_n;\ \} : \underline{B}}$$

$$\frac{\Psi; \Gamma \vdash V : \exists \beta.\ A \qquad \Psi; \Gamma, (x : A[\alpha/\beta]) \vdash M : \underline{B}}{\Psi; \Gamma \vdash \mathrm{unpack}\ (\alpha, x) = V\ \mathrm{in}\ M : \underline{B}}$$

4.3 CC-CBPV to HA-CBPV: Type translation

$$A_1 \times A_2^+ = \mathrm{ptr}_V((0, A_1^+, A_2^+)) \qquad A_1 + A_2^+ = \mathrm{ptr}_V((1, A_1^+), (1, A_2^+)) \qquad U\underline{B}^+ = \mathrm{ptr}_M(\underline{B}^+)$$

4.4 CC-CBPV to HA-CBPV: Value translation

The value translation judgment has the form

$$\boxed{\Psi; \Gamma \vdash V : A \rightsquigarrow V^+; \mathcal{A}_{\mathrm{h}}; \mathcal{A}_{\mathrm{c}}}$$

$$\frac{}{\Psi; \Gamma \vdash x : \Gamma(x) \rightsquigarrow x; \bullet; \bullet} \qquad \frac{}{\Psi; \Gamma \vdash n : \mathrm{Int} \rightsquigarrow n; \bullet; \bullet} \qquad \frac{}{\Psi; \Gamma \vdash \mathrm{true} : \mathrm{Bool} \rightsquigarrow \mathrm{true}; \bullet; \bullet}$$

$$\frac{}{\Psi; \Gamma \vdash \mathrm{false} : \mathrm{Bool} \rightsquigarrow \mathrm{false}; \bullet; \bullet}$$

$$\frac{\Psi; \Gamma \vdash V : A_2[A_1/x] \rightsquigarrow V'; \mathcal{A}_\mathsf{h}; \mathcal{A}_\mathsf{c}}{\Psi; \Gamma \vdash \mathrm{pack}\ (A_1, V)\ \mathrm{as}\ \exists x.\ A_2 : \exists x.\ A_2 \rightsquigarrow \mathrm{pack}\ ({A_1}^+, V')\ \mathrm{as}\ \exists x.\ {A_2}^+; \mathcal{A}_\mathsf{h}; \mathcal{A}_\mathsf{c}}$$

$$\frac{\Psi; \Gamma \vdash V : A_1 \rightsquigarrow V'; \mathcal{A}_\mathsf{h}; \mathcal{A}_\mathsf{c}}{\Psi; \Gamma \vdash \mathrm{inl}\ V' : A_1 + A_2 \rightsquigarrow x; (x \mapsto_1 V), \mathcal{A}_\mathsf{h}; \mathcal{A}_\mathsf{c}}$$

$$\frac{\Psi; \Gamma \vdash V : A_2 \rightsquigarrow V'; \mathcal{A}_\mathsf{h}; \mathcal{A}_\mathsf{c}}{\Psi; \Gamma \vdash \mathrm{inr}\ V' : A_1 + A_2 \rightsquigarrow x; (x \mapsto_2 V), \mathcal{A}_\mathsf{h}; \mathcal{A}_\mathsf{c}}$$

$$\frac{\Psi; \Gamma \vdash V_1 : A_1 \rightsquigarrow V_1'; \mathcal{A}_{\mathsf{h}1}; \mathcal{A}_{\mathsf{c}1} \qquad \Psi; \Gamma \vdash V_2 : A_2 \rightsquigarrow V_2'; \mathcal{A}_{\mathsf{h}2}; \mathcal{A}_{\mathsf{c}2}}{\Psi; \Gamma \vdash (V_1', V_2') : A_1 \times A_2 \rightsquigarrow x; (x \mapsto_0 V_1, V_2), \mathcal{A}_{\mathsf{h}1}, \mathcal{A}_{\mathsf{h}2}; \mathcal{A}_{\mathsf{c}1}, \mathcal{A}_{\mathsf{c}2}}$$

$$\frac{\Psi; \Gamma \vdash M : \Psi[\ell] \rightsquigarrow M'; \mathcal{A}_\mathsf{c}}{\Psi; \Gamma \vdash \mathrm{thunk}\ M : U(\Psi[\ell]) \rightsquigarrow \mathrm{thunk}\ \ell; \bullet; (\ell \mapsto M'), \mathcal{A}_\mathsf{c}}$$

4.5 CC-CBPV to HA-CBPV: Computation translation

The computation translation judgment has the form

$$\boxed{\Psi; \Gamma \vdash M : \underline{B} \rightsquigarrow M^+; \mathcal{A}_\mathsf{c}}$$

$$\frac{}{[\![\bullet]\!](M) = M} \qquad \frac{}{[\![(x \mapsto_n \overline{V}), \mathcal{A}_{\mathrm{h}}]\!](M) = \text{let } x = \text{alloc}_n(\overline{V}) \text{ in } [\![\mathcal{A}_{\mathrm{h}}]\!](M)}$$

$$\frac{\Psi; \Gamma \vdash U\underline{B} : V \rightsquigarrow V'; \mathcal{A}_{\mathrm{h}}; \mathcal{A}_{\mathrm{c}}}{\Psi; \Gamma \vdash \text{force } V : \underline{B} \rightsquigarrow [\![\mathcal{A}_{\mathrm{h}}]\!](\text{force } V'); \mathcal{A}_{\mathrm{c}}} \qquad \frac{\Psi; \Gamma \vdash V : A \rightsquigarrow V'; \mathcal{A}_{\mathrm{h}}; \mathcal{A}_{\mathrm{c}}}{\Psi; \Gamma \vdash \text{ret } V : FA \rightsquigarrow [\![\mathcal{A}_{\mathrm{h}}]\!](\text{ret } V'); \mathcal{A}_{\mathrm{c}}}$$

$$\frac{\Psi; \Gamma \vdash M : FA \rightsquigarrow M'; \mathcal{A}_{\mathrm{c}M} \qquad \Psi; \Gamma, x : A \vdash N : \underline{B} \rightsquigarrow N'; \mathcal{A}_{\mathrm{c}N}}{\Psi; \Gamma \vdash x \leftarrow M;\ N : \underline{B} \rightsquigarrow x \leftarrow M';\ N'; \mathcal{A}_{\mathrm{c}M}; \mathcal{A}_{\mathrm{c}N}}$$

$$\frac{\Psi; \Gamma \vdash V : A \rightsquigarrow V'; \mathcal{A}_{\mathrm{h}V}; \mathcal{A}_{\mathrm{c}V} \Psi; \Gamma \vdash M : A \to \underline{B} \rightsquigarrow M'; \mathcal{A}_{\mathrm{c}M}}{\Psi; \Gamma \vdash M\ V : \underline{B} \rightsquigarrow [\![\mathcal{A}_{\mathrm{h}}]\!](M'\ V'); \mathcal{A}_{\mathrm{c}M}; \mathcal{A}_{\mathrm{c}V}}$$

$$\frac{\Psi; \Gamma, x : A \vdash M : \underline{B} \rightsquigarrow M'; \mathcal{A}_{\mathrm{c}}}{\Psi; \Gamma \vdash \lambda(x : A).\ M : A \to \underline{B} \rightsquigarrow \lambda(x : A^{+}).\ M'; \mathcal{A}_{\mathrm{c}}}$$

$$\frac{\begin{array}{c}\forall i, \Psi; \Gamma \vdash V_i : A_i \rightsquigarrow V_i'; \mathcal{A}_{\mathrm{h}i}; \mathcal{A}_{\mathrm{c}i} \\ \mathcal{A}_{\mathrm{h}} = \cup_i \mathcal{A}_{\mathrm{h}i} \qquad \mathcal{A}_{\mathrm{c}} = \cup_i \mathcal{A}_{\mathrm{c}i} \qquad \Psi; \Gamma, x : \text{ptr}_V((n, (A_1, \ldots, A_n))) \vdash M : \underline{B} \rightsquigarrow M'; \mathcal{A}_{\mathrm{c}M}\end{array}}{\Psi; \Gamma \vdash \text{let } x = \text{alloc}_n(V_1, \ldots, V_n) \text{ in } M : \underline{B} \rightsquigarrow [\![\mathcal{A}_{\mathrm{h}}]\!](\underline{B})\text{let } x = \text{alloc}_n(V_1', \ldots, V_n') \text{ in } M'; \mathcal{A}_{\mathrm{c}M}, \mathcal{A}_{\mathrm{c}}}$$

$$\frac{\Psi; \Gamma \vdash V : \text{ptr}_V((m, A_1, \ldots, A_n)) \rightsquigarrow V'; \mathcal{A}_{\mathrm{h}V}; \mathcal{A}_{\mathrm{c}V} \qquad \Psi; \Gamma, x : A_i \vdash M : \underline{B} \rightsquigarrow M'; \mathcal{A}_{\mathrm{c}M}}{\Psi; \Gamma \vdash \text{let } x = \text{prj}_i(V) \text{ in } M : \underline{B} \rightsquigarrow [\![\mathcal{A}_{\mathrm{h}V}]\!](\text{let } x = \text{prj}_i(V') \text{ in } M'); \mathcal{A}_{\mathrm{c}M}, \mathcal{A}_{\mathrm{c}V}}$$

$$\frac{\Psi; \Gamma \vdash V : \text{Bool} \rightsquigarrow V'; \mathcal{A}_{\mathrm{h}}; \mathcal{A}_{\mathrm{c}} \qquad \Psi; \Gamma \vdash M_1 : \underline{B} \rightsquigarrow M_1'; \mathcal{A}_{\mathrm{c}1} \qquad \Psi; \Gamma \vdash M_2 : \underline{B} \rightsquigarrow M_2'; \mathcal{A}_{\mathrm{c}2}}{\Psi; \Gamma \vdash \text{if } V \text{ then } M_1 \text{ else } M_2 : \underline{B} \rightsquigarrow [\![\mathcal{A}_{\mathrm{h}}]\!](\text{if } V' \text{ then } M_1' \text{ else } M_2'); \mathcal{A}_{\mathrm{c}}, \mathcal{A}_{\mathrm{c}1}, \mathcal{A}_{\mathrm{c}2}}$$

$$\frac{\begin{array}{c}\Psi; \Gamma \vdash V : \text{ptr}_V((n_1, \overline{A_1}), \ldots, (n_m, \overline{A_m})) \rightsquigarrow V'; \mathcal{A}_{\mathrm{h}V}; \mathcal{A}_{\mathrm{c}V} \\ \forall i, \Psi; \Gamma, x_i : \text{ptr}_V((n_i, \overline{A_i})) \vdash M_i : \underline{B} \rightsquigarrow M_i'; \mathcal{A}_{\mathrm{c}i} \qquad \mathcal{A}_{\mathrm{c}} = \cup_i \mathcal{A}_{\mathrm{c}i}\end{array}}{\begin{array}{l}\Psi; \Gamma \vdash \text{case } V\ \{n_1(x_1) \Rightarrow M_1;\ , \ldots, n_m(x_m) \Rightarrow M_m;\ \} : \underline{B} \\ \qquad \rightsquigarrow [\![\mathcal{A}_{\mathrm{h}V}]\!](\text{case } V'\ \{n_1(x_1) \Rightarrow M_1';\ , \ldots, n_m(x_m) \Rightarrow M_m';\ \}); \mathcal{A}_{\mathrm{c}V}, \mathcal{A}_{\mathrm{c}}\end{array}}$$

$$\frac{\Psi; \Gamma \vdash V : \exists \beta. A \rightsquigarrow V'; \mathcal{A}_{\mathrm{h}V}; \mathcal{A}_{\mathrm{c}V} \qquad \Psi; \Gamma, (x : A[\alpha/\beta]) \vdash M : \underline{B} \rightsquigarrow M'; \mathcal{A}_{\mathrm{c}M}}{\Psi; \Gamma \vdash \text{unpack } (\alpha, x) = V \text{ in } M : \underline{B} \rightsquigarrow [\![\mathcal{A}_{\mathrm{h}}]\!](\text{ret } V'); \mathcal{A}_{\mathrm{c}V}, \mathcal{A}_{\mathrm{c}M}}$$

4.6 CC-CBPV to HA-CBPV: Module translation

$$\frac{\Psi^{+} <: \Psi' \qquad \Psi'; \bullet \vdash M_i : \Psi'[\ell_i] \rightsquigarrow M_i'; \mathcal{A}_{\mathrm{c}i}}{\vdash \ell_1 \mapsto M_1, \ldots \ell_n \mapsto M_n : \Psi \rightsquigarrow \ell_1 \mapsto M_1', \ldots, \ell_n \mapsto \ell_n', \mathcal{A}_{\mathrm{c}1}, \ldots, \mathcal{A}_{\mathrm{c}n} : \Psi'}$$

5 LL-CBPV

Is there a need for anything much here?

Also, we were going to switch to using that ArXiV draft by Steve,right? Does anything significant change between the heap allocated IR above and that? It's not super clear.

(AA: I think LL-CBPV is Vellvm)

(LM: I'll use LL-CBPV for now for the Oxide target, which is similar too HA-CBPV but a bit different (for instance, including pairs as value types in certain places—perhaps subject to a size constraint? but I'm not sure if we need that, actually—on closer inspection, I think LLVM will actually handle arbitrarily sized values, just allocating lots of registers/stack space for them as needed. (Which isn't to say that the HA pass isn't needed; it or something similar may be appropriate for the semantics of certain source languages).))

(LM: TODO: Think about garbage collection [important for unrestricted pointers])

(LM: TODO: For unrestricted pointers, do we actually need ω for ε, or can we use Φ?. Brief answer: Yes, we probably need ω because getting a copy of a truly unrestricted pointer out permanently out of a another capability isn't a strong update, while getting a fraction out is. Rather than omega, we should probably use $1/\omega$ (or some form of ϵ, as that is usually notated), however.)

(LM: TODO: Look at fractional permissions literature, including Boyland 2013. Our `own` primitive may be fundamentally connected to their nesting/focusing setup! As well for terminology/etc.)

5.1 LL-CBPV: Syntax

$$
\begin{array}{llcl}
\text{Locations} & \eta & ::= & \varrho \mid \eta + \eta \mid \eta + n \\[1ex]
\text{Initialization Markers} & I & ::= & \circ \mid \bullet \\[1ex]
\text{Frame contexts} & \mathcal{F} & ::= & \bullet \mid \mathcal{F}, x : A \mid \Gamma, \varrho \\[1ex]
\text{Contexts} & \Gamma & ::= & \bullet \mid \Gamma :: \mathcal{F} \mid \Gamma \circledast \mathcal{F} \\[1ex]
\text{Value Types} & A & ::= & \mathrm{Int}_n \mid \mathrm{Bool} \mid \mathrm{Unit} \mid A \times A \mid \mathrm{ptr}_V\ \eta \mid \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ A \mid \eta\varepsilon = \eta \mid \\
 & & & \mathrm{ptr}_M(\underline{B}) \mid \alpha \\
 & & \mid & \exists x.\, A \\[1ex]
\text{Computation Types} & \underline{B} & ::= & A \rightarrow \underline{B} \mid FA \\
 & & \mid & \forall \varrho.\, \underline{B} \\
 & & \mid & \forall \alpha.\, \underline{B} \\[1ex]
\text{Module Types} & \Psi & ::= & \overline{\ell \mapsto \underline{B}} \\[1ex]
\text{Values} & V & ::= & x \mid n \mid \mathsf{true} \mid \mathsf{false} \mid \mathsf{unit} \mid \mathsf{thunk}\ \ell \\
 & & \mid & (V, V) \mid \mathsf{pack}\ (A, V)\ \mathsf{as}\ \exists x.\, A \\[1ex]
\text{Computations} & M & ::= & \mathsf{force}\ V \mid \mathsf{ret}\ V \mid x \leftarrow M;\ N \mid M\ V \mid \lambda(x : A).\, M \\
 & & \mid & \mathsf{let}\ (x, x) = V\ \mathsf{in}\ M \\
 & & \mid & \mathsf{if}\ V\ \mathsf{then}\ M\ \mathsf{else}\ M \\
 & & \mid & \mathsf{unpack}\ (\alpha, x) = V\ \mathsf{in}\ M \\
 & & \mid & \mathsf{read}\ x\ \mathsf{from}\ V\ \mathsf{by}\ V\ \mathsf{rcap}\ x.\, M \mid \mathsf{write}\ V\ \mathsf{into}\ V\ \mathsf{by}\ V\ \mathsf{rcap}\ x.\, M \\
 & & \mid & \mathsf{take}\ x\ \mathsf{from}\ V\ \mathsf{by}\ V\ \mathsf{rcap}\ x.\, M \mid \mathsf{give}\ V\ \mathsf{to}\ V\ \mathsf{by}\ V\ \mathsf{rcap}\ x.\, M \\
 & & \mid & \mathsf{break}\ V\ \mathsf{into}\ x\ \mathsf{and}\ x.\, M \mid \mathsf{mend}\ V\ \mathsf{and}\ V\ \mathsf{into}\ x.\, M \\
 & & \mid & \mathsf{split}\ V\ \mathsf{into}\ x\ \mathsf{and}\ x.\, M \mid \mathsf{join}\ V\ \mathsf{and}\ V\ \mathsf{into}\ x.\, M \\
 & & \mid & \mathsf{own}\ V\ \mathsf{as}\ x\ \mathsf{coercion}\ x.\, M \mid \mathsf{coerce}\ V\ \mathsf{by}\ V\ \mathsf{as}\ x.\, M \mid \mathsf{relinquish}\ V\ \mathsf{by}\ V\ \mathsf{as}\ x.\, M \\[1ex]
\text{Definitions} & d & ::= & (M, \Phi, \overline{A}) \\[1ex]
\text{Modules} & mod & ::= & \overline{\ell \mapsto d}
\end{array}
$$

5.2 LL-CBPV: Statics

$$\boxed{\textsc{lin}(A) = \text{Bool}} \quad \text{Read: Type } A \text{ is/isn't linear}$$

$$\frac{}{\textsc{lin}(\text{Int}_n) = \bot} \qquad \frac{}{\textsc{lin}(\text{Bool}) = \bot} \qquad \frac{}{\textsc{lin}(A_1 \times A_2) = \textsc{lin}(A_1) \lor \textsc{lin}(A_2)} \qquad \frac{}{\textsc{lin}(\text{ptr}_V\ \eta) = \bot}$$

$$\frac{}{\textsc{lin}(\text{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ A) = \top} \qquad \frac{}{\textsc{lin}(\eta\varepsilon = \eta) = \bot} \qquad \frac{}{\textsc{lin}(\text{ptr}_M(\underline{B})) = \bot} \qquad \frac{}{\textsc{lin}(\exists x.\ A) = \textsc{lin}(A)}$$

$$\boxed{\textsc{size}(A) = n} \quad \text{Read: Type } A \text{ has size } n$$

$$\frac{}{\textsc{size}(\text{Int}_n) = n} \qquad \frac{}{\textsc{size}(\text{Bool}) = \textsc{bool-size}} \qquad \frac{}{\textsc{size}(A_1 \times A_2) = \textsc{size}(A_1) + \textsc{size}(A_2)}$$

$$\frac{}{\textsc{size}(\text{ptr}_V\ \eta) = \textsc{pointer-size}} \qquad \frac{}{\textsc{size}(\text{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ A) = 0} \qquad \frac{}{\textsc{size}(\eta\varepsilon = \eta) = 0}$$

$$\frac{}{\textsc{size}(\text{ptr}_M(\underline{B})) = \textsc{pointer-size}} \qquad \frac{}{\textsc{size}(\exists x.\ A) = \textsc{size}(A)}$$

$$\boxed{(x, A) \in \Gamma} \quad \text{Read: Variable } x \text{ has type } A \text{ in context } \Gamma$$

$$\frac{(x : A) \in \mathcal{F}}{(x : A) \in \Gamma :: \mathcal{F}} \qquad \frac{(x : A) \in \mathcal{F}}{(x : A) \in \Gamma \circledast \mathcal{F}} \qquad \frac{(x : A) \notin \mathcal{F} \qquad (x : A) \in \Gamma}{(x : A) \in \Gamma \circledast \mathcal{F}}$$

$$\boxed{\Gamma \setminus x = \Gamma'} \quad \text{Read: } \Gamma' \text{ is } \Gamma \text{ without } x$$

$$\frac{\exists A.\ (x : A) \in \mathcal{F}}{(\Gamma :: \mathcal{F}) \setminus x = \Gamma :: (\mathcal{F} \setminus x)} \qquad \frac{\exists A.\ (x : A) \in \mathcal{F}}{(\Gamma \circledast \mathcal{F}) \setminus x = \Gamma \circledast (\mathcal{F} \setminus x)} \qquad \frac{\neg \exists A.\ (x : A) \in \mathcal{F}}{(\Gamma \circledast \mathcal{F}) \setminus x = (\Gamma \setminus x) \circledast \mathcal{F}}$$

$$\boxed{\Psi; \Gamma \vdash V : A \dashv \Gamma'} \quad \text{Read: In function context } \Psi \text{ and context } \Gamma,\ V \text{ has type } A \text{ with output context } \Gamma'$$

$$\frac{(x, A) \in \Gamma \qquad \neg\, \textsc{lin}(A)}{\Psi; \Gamma \vdash x : A \dashv \Gamma} \qquad \frac{(x, A) \in \Gamma \qquad \textsc{lin}(A)}{\Psi; \Gamma \vdash x : A \dashv \Gamma \setminus x} \qquad \frac{}{\Psi; \Gamma \vdash n : \text{Int}_m \dashv \Gamma}$$

$$\frac{}{\Psi; \Gamma \vdash \text{true} : \text{Bool} \dashv \Gamma} \qquad \frac{}{\Psi; \Gamma \vdash \text{false} : \text{Bool} \dashv \Gamma} \qquad \frac{}{\Psi; \Gamma \vdash \text{unit} : \text{Unit} \dashv \Gamma}$$

$$\frac{\Psi[\ell] = \underline{B}}{\Psi; \Gamma \vdash \text{thunk}\ \ell : \text{ptr}_M(\underline{B}) \dashv \Gamma} \qquad \frac{\Psi; \Gamma \vdash V : A'[A/x] \dashv \Gamma'}{\Psi; \Gamma \vdash \text{pack}\ (A, V)\ \text{as}\ \exists x.\ A' : \exists x.\ A' \dashv \Gamma'}$$

$$\boxed{\Psi; \Gamma \vdash M : \underline{B} \dashv \Gamma'} \quad \text{Read: In function context } \Psi \text{ and context } \Gamma,\ M \text{ has type } \underline{B} \text{ with output context } \Gamma'$$

$$\frac{\Psi;\Gamma \vdash V : \mathrm{ptr}_M(\underline{B}) \dashv \Gamma'}{\Psi;\Gamma \vdash \mathrm{force}\ V : \underline{B} \dashv \Gamma'} \qquad \frac{\Psi;\Gamma \vdash V : A \dashv \Gamma'}{\Psi;\Gamma \vdash \mathrm{ret}\ V : FA \dashv \Gamma'}$$

$$\frac{\Psi;\Gamma \vdash M : FA \dashv \Gamma' \qquad \Psi;\Gamma', (x : A) \vdash N : \underline{B} \dashv \Gamma''}{\Psi;\Gamma \vdash x \leftarrow M;\ N : \underline{B} \dashv \Gamma''}$$

$$\frac{\Psi;\Gamma \vdash M : A \rightarrow \underline{B} \dashv \Gamma' \qquad \Psi;\Gamma' \vdash V : A \dashv \Gamma''}{\Psi;\Gamma \vdash M\ V : \underline{B} \dashv \Gamma''}$$

$$\frac{\Psi;\Gamma{::}\bullet, x : A \vdash M : \underline{B} \dashv \Gamma'{::}\mathcal{F} \qquad \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x))}{\Psi;\Gamma \vdash \lambda(x : A).\ M : A \rightarrow \underline{B} \dashv \Gamma'}$$

$$\frac{\Psi;\Gamma \vdash V : A_l \times A_r \dashv \Gamma' \qquad \Psi;\Gamma'\circledast\bullet, x : A_l, y : A_r \vdash M : \underline{B} \dashv \Gamma''\circledast\mathcal{F} \qquad \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x))}{\Psi;\Gamma \vdash \mathrm{let}\ (x, y) = V\ \mathrm{in}\ M : \underline{B} \dashv \Gamma''}$$

$$\frac{\Psi;\Gamma \vdash V : \mathrm{Bool} \dashv \Gamma' \qquad \Psi;\Gamma' \vdash M_1 : \underline{B} \dashv \Gamma'' \qquad \Psi;\Gamma' \vdash M_2 : \underline{B} \dashv \Gamma''}{\Psi;\Gamma \vdash \mathrm{if}\ V\ \mathrm{then}\ M_1\ \mathrm{else}\ M_2 : \underline{B} \dashv \Gamma''}$$

$$\frac{\Psi;\Gamma \vdash V : M \dashv \exists\beta.\ A\Gamma' \qquad \Psi;\Gamma\circledast\bullet, x : A[\alpha/\beta] \vdash M : \underline{B} \dashv \Gamma''\circledast\mathcal{F} \qquad \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x))}{\Psi;\Gamma \vdash \mathrm{unpack}\ (\alpha, x) = V\ \mathrm{in}\ M : \underline{B} \dashv \Gamma''}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V_p : \mathrm{ptr}_V\ \eta \dashv \Gamma_1 \qquad \Psi;\Gamma_1 \vdash V_c : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon \bullet A \dashv \Gamma_2 \\ \Psi;\Gamma_2\circledast\bullet, x_v : A, x_c : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon \bullet A \vdash M : \underline{B} \dashv \Gamma_3\circledast\mathcal{F} \\ \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x)) \qquad \Phi(\varepsilon) \preccurlyeq \mathsf{R} \qquad \neg\,\textsc{lin}(A)\end{array}}{\Psi;\Gamma \vdash \mathrm{read}\ x_v\ \mathrm{from}\ V_p\ \mathrm{by}\ V_c\ \mathrm{rcap}\ x_c.\ M : \underline{B} \dashv \Gamma_3}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V_p : \mathrm{ptr}_V\ \eta \dashv \Gamma_1 \qquad \Psi;\Gamma_1 \vdash V_c : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ A \dashv \Gamma_2 \qquad \Psi;\Gamma_2 \vdash V_v : A' \dashv \Gamma_3 \\ \Psi;\Gamma_3\circledast\bullet, x_c : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon \bullet A' \vdash M : \underline{B} \dashv \Gamma_4\circledast\mathcal{F} \qquad \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x)) \\ \textsc{size}(A) = \textsc{size}(A') \qquad \Phi(\varepsilon) \preccurlyeq \mathsf{S} \vee (A = A' \wedge \Phi(\varepsilon) \preccurlyeq \mathsf{W}) \qquad \neg\,\textsc{lin}(A)\end{array}}{\Psi;\Gamma \vdash \mathrm{write}\ V_v\ \mathrm{into}\ V_p\ \mathrm{by}\ V_c\ \mathrm{rcap}\ x_c.\ M : \underline{B} \dashv \Gamma_4}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V_p : \mathrm{ptr}_V\ \eta \dashv \Gamma_1 \qquad \Psi;\Gamma_1 \vdash V_c : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon \bullet A \dashv \Gamma_2 \\ \Psi;\Gamma_2\circledast\bullet, x_v : A, x_c : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon \circ A \vdash M : \underline{B} \dashv \Gamma_3\circledast\mathcal{F} \\ \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x)) \qquad \Phi(\varepsilon) \preccurlyeq \mathsf{S} \qquad \textsc{lin}(A)\end{array}}{\Psi;\Gamma \vdash \mathrm{take}\ x_v\ \mathrm{from}\ V_p\ \mathrm{by}\ V_c\ \mathrm{rcap}\ x_c.\ M : \underline{B} \dashv \Gamma_3}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V_p : \mathrm{ptr}_V\ \eta \dashv \Gamma_1 \qquad \Psi;\Gamma_1 \vdash V_c : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon \circ A \dashv \Gamma_2 \\ \Psi;\Gamma_2 \vdash V_v : A' \dashv \Gamma_3 \qquad \Psi;\Gamma_3\circledast\bullet, x_c : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon \bullet A' \vdash M : \underline{B} \dashv \Gamma_4\circledast\mathcal{F} \\ \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x)) \qquad \textsc{size}(A) = \textsc{size}(A') \qquad \Phi(\varepsilon) \preccurlyeq \mathsf{S} \qquad \textsc{lin}(A)\end{array}}{\Psi;\Gamma \vdash \mathrm{give}\ V_v\ \mathrm{to}\ V_p\ \mathrm{by}\ V_c\ \mathrm{rcap}\ x_c.\ M : \underline{B} \dashv \Gamma_4}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ (A_l \times A_r) \dashv \Gamma' \\ \Psi;\Gamma'\circledast\bullet, x : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ A_l, y : \mathrm{cap}_V\ (\eta + \textsc{size}(A_l))\ \Phi\ \varepsilon\ I\ A_r \vdash M : \underline{B} \dashv \Gamma''\circledast\mathcal{F} \\ \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x))\end{array}}{\Psi;\Gamma \vdash \text{break } V \text{ into } x \text{ and } y.\ M : \underline{B} \dashv \Gamma''}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V_l : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ A_l \dashv \Gamma_1 \qquad \Psi;\Gamma_1 \vdash V_r : \mathrm{cap}_V\ (\eta + \textsc{size}(A_l))\ \Phi\ \varepsilon\ I\ A_r \dashv \Gamma_2 \\ \Psi;\Gamma_2\circledast\bullet, x : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ (A_l \times A_r) \vdash M : \underline{B} \dashv \Gamma_3\circledast\mathcal{F} \qquad \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x))\end{array}}{\Psi;\Gamma \vdash \text{mend } V_l \text{ and } V_r \text{ into } x.\ M : \underline{B} \dashv \Gamma_3}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon_1 + \varepsilon_2\ I\ A \dashv \Gamma' \\ \Psi;\Gamma'\circledast\bullet, x : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon_1\ I\ A, y : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon_2\ I\ A \vdash M : \underline{B} \dashv \Gamma''\circledast\mathcal{F} \qquad \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x))\end{array}}{\Psi;\Gamma \vdash \text{split } V \text{ into } x \text{ and } y.\ M : \underline{B} \dashv \Gamma''}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V_1 : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon_1\ I\ A \dashv \Gamma_1 \qquad \Psi;\Gamma_1 \vdash V_2 : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon_2\ I\ A \dashv \Gamma_2 \\ \Psi;\Gamma_2\circledast\bullet, x : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon_1 + \varepsilon_2\ I\ A \vdash M : \underline{B} \dashv \Gamma_3\circledast\mathcal{F} \qquad \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x))\end{array}}{\Psi;\Gamma \vdash \text{join } V_1 \text{ and } V_2 \text{ into } x.\ M : \underline{B} \dashv \Gamma_3}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ A \dashv \Gamma' \\ \Psi;\Gamma'\circledast\bullet, \varrho, x : \mathrm{cap}_V\ \varrho\ \Phi\ \varepsilon'\ I\ [\![A]\!]^{\varepsilon/\varepsilon'}, y : \varrho\varepsilon/\varepsilon' = \eta \vdash M : \underline{B} \dashv \Gamma''\circledast\mathcal{F} \\ \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x)) \qquad \varepsilon \leq \varepsilon' \leq 1 \qquad \textsc{size}(A) = 0\end{array}}{\Psi;\Gamma \vdash \text{own } V \text{ as } x \text{ coercion } y.\ M : \underline{B} \dashv \Gamma''}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V_p : \mathrm{ptr}_V\ \eta \dashv \Gamma_1 \\ \Psi;\Gamma_1 \vdash V_c : \eta'\varepsilon = \eta \dashv \Gamma_2 \qquad \Psi;\Gamma_2\circledast\bullet, x : \mathrm{ptr}_V\ \eta' \vdash M : \underline{B} \dashv \Gamma_3\circledast\mathcal{F} \qquad \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x))\end{array}}{\Psi;\Gamma \vdash \text{coerce } V_p \text{ by } V_e \text{ as } x.\ M : \underline{B} \dashv \Gamma_3}$$

$$\frac{\begin{array}{c}\Psi;\Gamma \vdash V_c : \mathrm{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ A \dashv \Gamma_1 \qquad \Psi;\Gamma \vdash V_e : \eta\varepsilon' = \eta' \dashv \Gamma_2 \\ \Psi;\Gamma_2\circledast\bullet, x : \mathrm{cap}_V\ \eta'\ \Phi\ (\varepsilon\varepsilon')\ I\ [\![A]\!]^{1/\varepsilon'} \vdash M : \underline{B} \dashv \Gamma_3\circledast\mathcal{F} \qquad \forall x \in \mathcal{F}.\ \neg\,\textsc{lin}(\mathcal{F}(x))\end{array}}{\Psi;\Gamma \vdash \text{relinquish } V_c \text{ by } V_e \text{ as } x.\ M : \underline{B} \dashv \Gamma_3}$$

$\boxed{[\![A]\!]^{\varepsilon} = A'}$ Read: A' is an ε-fraction of A

$$\frac{}{[\![A_l \times A_r]\!]^{\varepsilon} = [\![A_l]\!]^{\varepsilon} \times [\![A_r]\!]^{\varepsilon}} \qquad \frac{}{[\![\mathrm{cap}_V\ \eta\ \Phi\ \varepsilon'\ I\ A]\!]^{\varepsilon} = \mathrm{cap}_V\ \eta\ \Phi\ (\varepsilon'\varepsilon)\ I\ A}$$

$\boxed{[\![\underline{B}]\!]^{\Phi,\overline{\eta,A}^n} = \underline{B}'}$ Read: $\underline{B}'$ is the type of a function body which must return memory $\overline{\eta,A}^n$ and has apparent type $\underline{B}$

$$\frac{\mathrm{fresh}(\varrho)}{[\![A \to \underline{B}]\!]^{\Phi,\overline{\eta,A}^n} = \forall\varrho.\ \mathrm{ptr}_V\ \varrho \to \mathrm{cap}_V\ \varrho\ \Phi\ 1 \bullet A \to [\![\underline{B}]\!]^{\Phi,\varrho,A,\overline{\eta,A}^n}}$$

$$\frac{}{[\![FA]\!]^{\Phi,\overline{\eta,A}^n} = F(A \times \overline{\mathrm{cap}_V\ \eta_i\ \Phi\ 1\ I\ A_i}^{i\in n})}$$

$\boxed{\Psi \vdash d : \underline{B}}$ Read: In function context Ψ, definition d has type $\underline{B}$

$$\frac{\overline{\mathrm{fresh}(\varrho_i)}^{i\in n} \qquad \Psi;\bullet \vdash M : \overline{\mathrm{ptr}_V\ \varrho_i \to \mathrm{cap}_V\ \varrho_i\ \Phi\ 1 \circ A_i \to}^{i\in n} [\![\underline{B}]\!]^{\Phi,\overline{\varrho_i,A_i}^{i\in n}} \dashv \bullet}{\Psi \vdash (M, \Phi, \overline{A}^n) : \underline{B}}$$

$\boxed{mod : \Psi}$ Read: Module mod has type Ψ

$$\frac{\forall i.\ \exists \underline{B}.\ \ell_i \mapsto \underline{B} \in \Psi \wedge \Psi \vdash d_i : \underline{B}}{\ell_1 \mapsto d_1, \ldots, \ell_1 \mapsto d_n : \Psi}$$

5.3 LL-CBPV: Dynamics

5.3.1 LL-CBPV: Dynamic syntax.

Values	V	::=	$\ldots \mid \mathrm{cap}\ \Phi\ \varepsilon\ \rho\ n \mid \mathrm{ptr}\ \rho$
Heaps	$\mathcal{H}$	::=	$\bullet \mid \mathcal{H}, \rho \mapsto V \mid \mathcal{H}, \rho \mapsto_\varepsilon \rho$
Evaluation contexts	E	::=	$[]$
		$\mid$	$\mathrm{force}\ V \mid \mathrm{ret}\ V \mid x \leftarrow E;\ M \mid E\ M \mid V\ E \mid \lambda(x : A).\ M$
		$\mid$	$\mathrm{let}\ (x, x) = V\ \mathrm{in}\ M$
		$\mid$	$\mathrm{if}\ V\ \mathrm{then}\ M\ \mathrm{else}\ M$
		$\mid$	$\mathrm{unpack}\ (\alpha, x) = V\ \mathrm{in}\ M$
		$\mid$	$\mathrm{read}\ x\ \mathrm{from}\ V\ \mathrm{by}\ V\ \mathrm{rcap}\ x.\ M \mid \mathrm{write}\ V\ \mathrm{into}\ V\ \mathrm{by}\ V\ \mathrm{rcap}\ x.\ M$
		$\mid$	$\mathrm{take}\ x\ \mathrm{from}\ V\ \mathrm{by}\ V\ \mathrm{rcap}\ x.\ M \mid \mathrm{give}\ V\ \mathrm{to}\ V\ \mathrm{by}\ V\ \mathrm{rcap}\ x.\ M$
		$\mid$	$\mathrm{break}\ V\ \mathrm{into}\ x\ \mathrm{and}\ x.\ M \mid \mathrm{mend}\ V\ \mathrm{and}\ V\ \mathrm{into}\ x.\ M$
		$\mid$	$\mathrm{split}\ V\ \mathrm{into}\ x\ \mathrm{and}\ x.\ M \mid \mathrm{join}\ V\ \mathrm{and}\ V\ \mathrm{into}\ x.\ M$
		$\mid$	$\mathrm{own}\ V\ \mathrm{as}\ x\ \mathrm{coercion}\ x.\ M \mid \mathrm{coerce}\ V\ \mathrm{by}\ V\ \mathrm{as}\ x.\ M \mid \mathrm{relinquish}\ V\ \mathrm{by}\ V\ \mathrm{as}\ x.\ M$

Note that although capability values exist, they are never inspected by the below operational semantics.

$\boxed{mod; \mathcal{H} \vdash M \rightsquigarrow \mathcal{H}'; M'}$ Read: Computation M in module mod reduces under heap $\mathcal{H}$ to M' and heap $\mathcal{H}'$

$$\frac{mod[\ell] = (M, \Phi, \overline{A_i}^{i \in n}) \qquad \overline{\mathrm{fresh}(\rho_i)}^{i \in n} \qquad \mathrm{fresh}(x) \qquad \mathrm{fresh}(y) \qquad \mathrm{fresh}(z)}{mod; \mathcal{H} \vdash \mathrm{force}\ \mathrm{thunk}\ \ell \rightsquigarrow \mathcal{H}; x \leftarrow M\ \overline{(\mathrm{ptr}\ \rho_i)\ (\mathrm{cap}\ \Phi\ 1\ \rho_i\ \textsc{size}(A_i))}^{i \in n};\ \mathrm{let}\ (y, z) = x\ \mathrm{in}\ y}$$

$$\frac{}{mod; \mathcal{H} \vdash x \leftarrow \mathrm{ret}\ V;\ M \rightsquigarrow \mathcal{H}; M[V/x]} \qquad \frac{}{mod; \mathcal{H} \vdash (\lambda(x : A).\ M)\ V \rightsquigarrow \mathcal{H}; M[V/x]}$$

$$\frac{}{mod; \mathcal{H} \vdash \mathrm{let}\ (x_1, x_2) = (V_1, V_2)\ \mathrm{in}\ M \rightsquigarrow \mathcal{H}; M[V_1/x_1][V_2/x_2]}$$

$$\frac{}{mod; \mathcal{H} \vdash \mathrm{if}\ \mathrm{true}\ \mathrm{then}\ M_t\ \mathrm{else}\ M_f \rightsquigarrow \mathcal{H}; M_t} \qquad \frac{}{mod; \mathcal{H} \vdash \mathrm{if}\ \mathrm{false}\ \mathrm{then}\ M_t\ \mathrm{else}\ M_f \rightsquigarrow \mathcal{H}; M_f}$$

$$\frac{mod; \mathcal{H} \vdash M \rightsquigarrow \mathcal{H}'; M'}{mod; \mathcal{H} \vdash E[M] \rightsquigarrow \mathcal{H}'; E[M']}$$

$$\frac{}{mod; \mathcal{H} \vdash \mathrm{unpack}\ (\alpha, x) = \mathrm{pack}\ (A, V)\ \mathrm{as}\ \exists x'.\ A'\ \mathrm{in}\ M \rightsquigarrow \mathcal{H}; M[V/x]}$$

$$\frac{\mathcal{H}[\rho] = V_v}{mod; \mathcal{H} \vdash \mathrm{read}\ x_v\ \mathrm{from}\ \mathrm{ptr}\ \rho\ \mathrm{by}\ V_c\ \mathrm{rcap}\ x_c.\ M \rightsquigarrow \mathcal{H}; M[V_c/x_c][V_v/x_v]}$$

$$\frac{}{mod; \mathcal{H} \vdash \mathrm{write}\ V_v\ \mathrm{into}\ \mathrm{ptr}\ \rho\ \mathrm{by}\ V_c\ \mathrm{rcap}\ x_c.\ M \rightsquigarrow \mathcal{H}[\rho \mapsto V_v]; M[V_c/x_c]}$$

$$\frac{\mathcal{H}[\rho] = V_v}{mod; \mathcal{H} \vdash \text{take } x_v \text{ from ptr } \rho \text{ by } V_c \text{ rcap } x_c.\ M \rightsquigarrow \mathcal{H}; M[V_c/x_c][V_v/x_v]}$$

$$\frac{}{mod; \mathcal{H} \vdash \text{give } V_v \text{ to ptr } \rho \text{ by } V_c \text{ rcap } x_c.\ M \rightsquigarrow \mathcal{H}[\rho \mapsto V_v]; M[V_c/x_c]}$$

$$\frac{}{mod; \mathcal{H} \vdash \text{break cap } \Phi\ \varepsilon\ \rho\ (n_1 + n_2) \text{ into } x_1 \text{ and } x_2.\ M \rightsquigarrow \mathcal{H}; M[\text{cap } \Phi\ \varepsilon\ \rho\ n_1/x_1][\text{cap } \Phi\ \varepsilon\ (\rho + n_1)\ n_2/x_2]}$$

$$\frac{}{mod; \mathcal{H} \vdash \text{mend } (\text{cap } \Phi\ \varepsilon\ \rho\ n_1) \text{ and } (\text{cap } \Phi\ \varepsilon\ (\rho + n_1)\ n_2) \text{ into } x.\ M \rightsquigarrow \mathcal{H}; M[\text{cap } \Phi\ \varepsilon\ \rho\ (n_1 + n_2)/x]}$$

$$\frac{}{mod; \mathcal{H} \vdash \text{split cap } \Phi\ (\varepsilon_1 + \varepsilon_2)\ \rho\ n \text{ into } x_1 \text{ and } x_2.\ M \rightsquigarrow \mathcal{H}; M[\text{cap } \Phi\ \varepsilon_1\ \rho\ n/x_1][\text{cap } \Phi\ \varepsilon_2\ \rho\ n/x_2]}$$

$$\frac{}{mod; \mathcal{H} \vdash \text{join cap } \Phi\ \varepsilon_1\ \rho\ n \text{ and cap } \Phi\ \varepsilon_2\ \rho\ n \text{ into } x.\ M \rightsquigarrow \mathcal{H}; M[\text{cap } \Phi\ (\varepsilon_1 + \varepsilon_2)\ \rho\ n/x]}$$

Notes:
There are problems with sizing closures
We should elaborate into having explicit numbers in break/mend/split/join [these semantics are currently wrong].

5.4 LL-CBPV: Soundness

$$\text{Ownership map} \quad \mathrm{O} \quad ::= \quad \overline{\rho \mapsto \epsilon}$$

$$\text{Semantic store} \quad \mathcal{W} \quad ::= \quad \rho \mapsto \epsilon\rho \mid \rho \mapsto \overline{(\mathcal{W}, V)}$$

$\boxed{\mathcal{W} \sqsubseteq_{\mathrm{O}} \mathcal{W}'}$ Read: Semantics store $\mathcal{W}'$ is an O-extension of semantic store $\mathcal{W}$

$$\mathcal{W} \sqsubseteq_{\mathrm{O}} \mathcal{W} \triangleq \forall \rho \in \mathrm{dom}(\mathcal{W}), \mathrm{keeps}(\mathrm{O}(\rho), \rho, \mathcal{W}, \mathcal{W}')$$

$$\mathrm{keeps}(\varepsilon, \rho, \mathcal{W}, \mathcal{W}') \triangleq \begin{cases} \textbf{true} & \mathrm{S} \preccurlyeq \Phi(\varepsilon) \\ \mathcal{W}'(\rho) = \mathcal{W}\rho & \text{otherwise} \end{cases}$$

$$\mathcal{V}[\![\mathrm{Int}_n]\!]_\psi = \{ (\mathcal{W}, m) \mid m < 2^n \} \qquad \mathcal{V}[\![\mathrm{Bool}]\!]_\psi = \{ (\mathcal{W}, \mathrm{true}), (\mathcal{W}, \mathrm{false}) \}$$

$$\mathcal{V}[\![A_1 \times A_2]\!]_= \{ (\mathcal{W}, (v_1, v_2)) \mid (\mathcal{W}, v_1) \in \mathcal{V}[\![A_1]\!]_\psi \wedge (\mathcal{W}, v_2) \mid \mathcal{V}[\![A_2]\!]_\psi \}$$

$$\mathcal{V}[\![\mathrm{ptr}_V\ \eta]\!]_\psi = \{ (\mathcal{W}, \mathrm{ptr}\ \psi(\eta)) \}$$

$$\mathcal{V}[\![\mathrm{cap}_V\ \eta\ \Phi\ \varepsilon\ I\ A]\!]_\psi = \{ (\mathcal{W}, \mathrm{cap}\ \Phi\ \varepsilon\ \rho\ n) \mid n = \textsc{size}(A) \wedge \rho = \psi(\eta) \wedge ((\mathcal{W}(\rho) \subseteq \mathcal{V}[\![A]\!]_\psi) \vee$$
$$(\exists \eta', \mathcal{W}(\rho) = \varepsilon'\rho' \wedge \rho' = \psi\eta' \wedge n = 0 \wedge (\mathcal{W}, \mathrm{cap}\ \Phi\ (\varepsilon' \cdot \varepsilon)\ \rho'\ n) \in \mathcal{V}[\![\mathrm{cap}_V\ \eta'\ \Phi\ (\varepsilon' \cdot \varepsilon)\ I\ A]\!]_\psi)) \}$$

$$\mathcal{V}[\![\eta\varepsilon = \eta']\!]_\psi = \{ (\mathcal{W}, \mathrm{coerce}) \mid \mathcal{W}(\psi(\eta)) = \varepsilon\eta' \}$$

$$\mathcal{V}[\![\mathrm{ptr}_M(\underline{B})]\!]_\psi = \{ (\mathcal{W}, \mathrm{thunk}\ \ell) \mid (\mathcal{W}, \mathrm{force\ thunk}\ \ell) \in \mathcal{E}[\![\underline{B}]\!]_\psi \}$$

$$\mathcal{E}[\![A \to \underline{B}]\!]_\psi = \{ (\mathcal{W}, M) \mid \forall \mathcal{W}', V.\ (\mathcal{W}', V) \in \mathcal{V}[\![A]\!]_\psi \wedge \mathcal{W} \sqsubseteq_{1-O(M)} \mathcal{W}' \Rightarrow (\mathcal{W}', M\ V) \in \mathcal{E}[\![\underline{B}]\!]_\psi \}$$

$$\mathcal{E}[\![FA]\!]_\psi = \{ (\mathcal{W}, M) \mid \exists \mathcal{W}', H, H'.\ \mathcal{W} \sqsubseteq_{O(M)} \mathcal{W}' \wedge H \in \mathcal{W} \wedge H' \in \mathcal{W}' \wedge (H, M) \mapsto^* (H', V) \wedge (\mathcal{W}', V) \in \mathcal{V}[\![A]\!]_\psi \}$$

$$[\![\Psi; \Gamma \vdash V : A \dashv \Gamma']\!] = \{ (mod, \mathcal{W}, V) \mid \forall \gamma \in \mathcal{G}[\![\Gamma]\!], (\mathcal{W}, V/\gamma) \in \mathcal{V}[\![A]\!]_\psi \}$$

$$[\![\Psi; \Gamma \vdash V : A \dashv \Gamma']\!] = \{ (mod, \mathcal{W}, V) \mid \forall \gamma \in \mathcal{G}[\![\Gamma]\!], (\mathcal{W}, M/\gamma) \in \mathcal{E}[\![A]\!]_\psi \}$$

5.5 Oxide to LL-CBPV: Syntax

Type translation context	Ξ	::=	$\bullet \mid \Xi, \alpha \mapsto \alpha \mid \Xi, \varphi \mapsto \alpha \mid \Xi, \rho \mapsto (\eta, \varepsilon)$
Path	*path*	::=	$\cdot \mid path.n \mid *(path, path)$
Place translation	*place-trans*	::=	$(path, \overline{(\varepsilon, path)})$
Translation context	ξ	::=	$\overline{p \mapsto place\text{-}trans}$

(LM: TODO: is the oxprov to mvPermFrac right? I don't really know where else the fraction in the type translation can come from, and I do think that it's needed.)

5.6 Oxide to LL-CBPV: Frame expression translation

Frame variables are used in closure types in oxide, but we don't have any equivalent such thing at the LL-CBPV level. Instead, we translate an oxide frame expression into a product A-type, which will be used at a few specific places.

$$\frac{\Xi(\varphi) = \alpha}{\Xi \vdash \varphi \rightsquigarrow \alpha} \qquad \frac{}{\Xi \vdash \bullet \rightsquigarrow \text{Unit}} \qquad \frac{\Xi \vdash \mathcal{F} \rightsquigarrow A \qquad \Xi \vdash \tau^{\text{SX}} \rightsquigarrow A'}{\Xi \vdash \mathcal{F}, x : \tau^{\text{SX}} \rightsquigarrow A \times A'} \qquad \frac{\Xi \vdash \mathcal{F} \rightsquigarrow A}{\Xi \vdash \mathcal{F}, r \mapsto \{\overline{\ell}\} \rightsquigarrow A}$$

(LM: TODO: Is this the right thing to do with the loan information? We shouldn't actually need it here, I hope, though of course it's presumably important to have somewhere? Presumably the relevant inforamtion will end up inside ξ somewhere....)

5.7 Oxide to LL-CBPV: Type translation

Currently, dynamically sized types are not handled quite right.

In order to handle DSTs we would presumably need to add some kind of (limited?) dependency, so that we can work with `Pi_(n:Int) cap eta Phi epsilon I (tau^+)^n`, which I believe is essentially what `&[tau]` means.

We also need this dependency for a proper model of sum types. Currently, this translation pretends that native sum types exist at the LL-CBPV level, which is quite false.

The combination of the fact that our type translation doesn't handle dead types with the fact that we haven't quite dealt with unsized types yet makes the type translation quite simple, except for the function case.

$$\frac{}{\Xi \vdash \texttt{bool} \rightsquigarrow \mathrm{Bool}} \qquad \frac{}{\Xi \vdash \texttt{bool} \rightsquigarrow \mathrm{Bool}} \qquad \frac{}{\Xi \vdash \texttt{u32} \rightsquigarrow \mathrm{Int}_{32}} \qquad \frac{}{\Xi \vdash \texttt{unit} \rightsquigarrow \mathrm{Unit}}$$

$$\frac{\Xi(\alpha) = \alpha}{\Xi \vdash \alpha \rightsquigarrow \alpha} \qquad \frac{\Xi(\rho) = (\eta, \varepsilon) \qquad \Xi \vdash \tau^{\mathrm{XI}} \rightsquigarrow A}{\Xi \vdash \&\rho\ \omega\ \tau^{\mathrm{XI}} \rightsquigarrow \mathrm{ptr}_V\ \eta \times \mathrm{cap}_V\ \eta\ \Phi_{\mathrm{ox}}\ \varepsilon \bullet A} \qquad \frac{}{\Xi \vdash [\tau^{\mathrm{SI}};\ 0] \rightsquigarrow \mathrm{Unit}}$$

$$\frac{\Xi \vdash \tau^{\mathrm{SI}} \rightsquigarrow A}{\Xi \vdash [\tau^{\mathrm{SI}};\ 1] \rightsquigarrow A} \qquad \frac{\Xi \vdash [\tau^{\mathrm{SI}};\ n] \rightsquigarrow A \qquad \Xi \vdash \tau^{\mathrm{SI}} \rightsquigarrow A'}{\Xi \vdash [\tau^{\mathrm{SI}};\ n+1] \rightsquigarrow A \times A'} \qquad \frac{\forall i, \Xi \vdash \tau_i^{\mathrm{SI}} \rightsquigarrow A_i}{\Xi \vdash (\tau_1^{\mathrm{SI}}\texttt{,}\ \ldots\texttt{,}\ \tau_n^{\mathrm{SI}}) \rightsquigarrow A_1 \times \cdots \times A_n}$$

$$\frac{\forall i, \Xi \vdash \tau_i^{\mathrm{SI}} \rightsquigarrow A_i}{\Xi \vdash \texttt{Either<}\tau_1^{\mathrm{SI}}\texttt{,}\ \tau_2^{\mathrm{SI}}\texttt{>} \rightsquigarrow A_1 + A_2} \qquad \frac{\text{(LM: TODO)}}{\Xi \vdash \forall\texttt{<}\overline{\varphi}\texttt{,}\ \overline{\varrho}\texttt{,}\ \overline{\alpha}\texttt{>}(\tau_1^{\mathrm{SI}}\texttt{,}\ \ldots\texttt{,}\ \tau_n^{\mathrm{SI}}) \xrightarrow{\Phi} \tau_r^{\mathrm{SI}}\ \texttt{where}\ \overline{\varrho_1;\varrho_2} \rightsquigarrow A}$$

My original understanding was that:

An oxide concrete region is matched to a set of loans, i.e. places to which it may be pointing. This is precisely equivalent to our notion of a location, which may be one-of-many locations via $\eta + \eta$.

A region-outlives constraint morally means that when all regions are instantiated, the loan set of the longer region is a superset of the loan set of the shorter region. This corresponds to our notion that the longer region is syntactically a superset of the shorter region, i.e. it can be divided into the origins of the short region and some other arbitrary set of origins that are in fact the passed in ones.

However, consider:

```
fn foo<'a, 'b, 'c, 'd>(x: &'a mut &'c mut i32, y: &'b mut &'d mut i32) where 'a : 'd {
    *y = *x;
}
```

The above interpretation of lifetime bounds doesn't really make sense in this case.

Upsettingly, what we actually want to generate as the type for this is likely something along the lines of:

```
foo: forall eta_a, eta_b, eta_c, eta_d,
  ptr eta_a -> cap eta_a Phi_ox 1 F (ptr eta_c * cap eta_c Phi_ox 1 F i32) ->
  ptr eta_b -> cap eta_b Phi_ox 1 F (ptr eta_d * cap eta_d Phi_ox 1 F i32) ->
  cap eta_a Phi_ox 1 E (ptr eta_c * cap eta_c Phi_ox 1 F i32) *
  cap eta_b Phi_ox 1 F (ptr (eta_c + eta_d) * cap (eta_c + eta_d) Phi_ox 1 F i32)
```

Is this even plausibly something that we can get from the lifetime bounds? If not, what other type is plausibly something that we can get from lifetime bounds that makes more sense?

5.8 Oxide to LL-CBPV: Place translation

Rust places, also known as *lvalues*, (oxide *place expressions*) are translated to M-computation contexts which break up caps as necessary to get to a useful point wherein other things that use the place can be put and have it just right there in a variable.

$\boxed{\xi \vdash_\varepsilon p \rightsquigarrow M^{[]}, V, V \dashv \xi'}$ Read: Place p may be accessed in ξ inside of $M^{[]}$ which contains context ξ', via p

$$\frac{\iota \subseteq \mathbb{N} \qquad |\iota| = n \qquad \varepsilon' = \Sigma_{i\in\iota}\varepsilon_i\varepsilon \leq \varepsilon' \qquad \xi(p) = (V_p, \overline{(\varepsilon_i, V_{ci})}) \qquad \forall i, \text{fresh}(x_i) \qquad V_1 = V_{ci} \qquad \forall i > 1, V_i = x_i}{\xi \vdash_\varepsilon p \rightsquigarrow \text{join } V_{c1} \text{ and } V_{c2} \text{ into } x_2.\ \text{join } x_2 \text{ and } V_{c3} \text{ into } x_3.\ \cdots \text{join } x_{n-1} \text{ and } V_{cn} \text{ into } x_n.\ [], V_p, x_n \dashv \xi \setminus \overline{V_{ci}} \cup \{}$$

$$\frac{\forall i, \xi_i \vdash_\varepsilon p.i \rightsquigarrow M_i, V_{p_i}, V_{c_i} \dashv \xi_{i+1}}{\xi \vdash_\varepsilon x \rightsquigarrow M_1[\cdots[M_n[\text{mend } V_{c1} \text{ and } V_{c2} \text{ into } x_2.\ \cdots \text{mend } x_{n-1} \text{ and } V_{cn} \text{ into } x_n.\ []]]], V_{p_0}, x_n \dashv \xi_n \setminus \overline{V_{ci}} \cup \{x \mapsto}$$

$$\frac{\xi \vdash_\varepsilon p \rightsquigarrow M, V_p, V_c \dashv \xi'}{\xi \vdash_\varepsilon *p \rightsquigarrow M[\text{take } y \text{ from } V_p \text{ by } V_c \text{ rcap } x_c.\ \text{let } (y_p, y_c) = y \text{ in } []] \dashv \xi' \setminus V_c \cup \{p \mapsto x_c, *p \mapsto (y_p, (\varepsilon, y_c))\}}$$

$$\frac{\xi \vdash_\varepsilon p \rightsquigarrow M, V_p, V_c \dashv \xi'}{\xi \vdash_\varepsilon p.i \rightsquigarrow M[\text{break } V_c \text{ into } x_{c1} \text{ and } x_2.\ \text{break } x_2 \text{ into } x_{c2} \text{ and } x_3.\ \cdots \text{break } x_{n-1} \text{ into } x_{cn-1} \text{ and } x_{cn}.\ []], V_p + \text{o}}$$

5.9 Oxide to LL-CBPV: Expression translation

Expressions, also known as *rvalues*, are translated to M-computations which return a pair of the actual final result of the expression and any other "stuff" that we had to mess with in order to get at it.

$$\frac{\xi \vdash_\omega p \rightsquigarrow M, V_p, V_c \dashv \xi'}{\xi \vdash p \rightsquigarrow M[\text{read } x \text{ from } V_p \text{ by } V_c \text{ rcap } x_c.\ []], x \dashv \xi' \setminus V_c \cup x_c}$$

$$\frac{\xi \vdash e \rightsquigarrow M_e, V_e \dashv \xi' \qquad \xi' \vdash_1 p \rightsquigarrow M_p, V_p, V_c \dashv \xi''}{\xi \vdash p := e \rightsquigarrow M_e[M_p[\text{write } V_e \text{ into } V_p \text{ by } V_c \text{ rcap } x_c.\ []]], \text{unit} \dashv \xi'' \setminus V_c \cup x_c}$$

$$\frac{\xi \vdash p \rightsquigarrow M, V_p, V_c \dashv oxTrCtx'}{\xi \vdash \&r\,\omega\,p \rightsquigarrow M, (V_p, V_c) \dashv \xi'}$$

5.10 Oxide to LL-CBPV: Module translation

6 VELLVM

Note that the vellvm type system is the same as the llvm one, and not really sufficient for much.

6.1 Vellvm: Syntax

Wherein we steal some syntax from the vellvm paper so that we can write translations.

6.2 Vellvm: Typing

Value Types	A	$::=$	$\mathbf{in} \mid A* \mid \{\overline{A}\} \mid \forall\alpha.\, A \mid \exists\alpha.\, A \mid [\overline{val = val \Rightarrow A}] \mid U\underline{B}$
Computation Types	$\underline{B}$	$::=$	$val \times l \times A \to \underline{B} \mid FA$
Module Types	Ψ	$::=$	.

What is this monstrosity? Not call by push value, but preserving the value/computation type distinction for *reasons*.

Also does not force structures to be treated as pointer types, for LLVM Reasons.

Definition typing:

$$\boxed{\Psi \vdash \textbf{define}\ typ\ id(\overline{arg})\{\overline{b}\} : \overline{id \times A} \longrightarrow FA}$$

Block typing:

$$\boxed{\Psi; \Xi \vdash b : \overline{val \times l \times A} \longrightarrow FA}$$

$$\frac{(l : args \longrightarrow FA) \in \Xi \qquad \exists \overline{A}. \Psi; \Xi; \overline{x : A}, l \vdash \overline{c}; tmn : FA \wedge \forall i. A_i^- = typ_i \wedge \forall j. (val_{i,j}, l_{i,j}, A_i) \in args}{\Psi; \Xi; \Gamma \vdash l\ \overline{(x = \textbf{phi}\ typ\ \overline{[val\ l]})}\ \overline{c}\ tmn : args \longrightarrow FA}$$

Instruction sequence typing:

$$\boxed{\Psi; \Xi; \Gamma; l \vdash \overline{c}; tmn : FA}$$

$$\frac{\begin{array}{c}\Psi; \Xi; \Gamma \vdash val_1 : \textbf{i}n_1 \qquad \Psi; \Xi; \Gamma \vdash val_2 : \textbf{i}n_2 \\ \exists \Gamma'. x = val_1\ op\ val_2 \vdash \Gamma' <: \Gamma \wedge \Psi; \Xi; \Gamma, (x : \mathrm{Bool}); l \vdash \overline{c}; tmn : FA\end{array}}{\Psi; \Xi; \Gamma; l \vdash x = \textbf{icmp}\ cond\ typ\ val_1\ val_2, \overline{c}; tmn : FA}$$

$$\frac{\Psi; \Xi; \Gamma \vdash val : A}{\Psi; \Xi; \Gamma; l \vdash \varnothing; \textbf{ret}\ val : FA}$$

$$\frac{\begin{array}{c}\Psi; \Xi; \Gamma \vdash val_c : \mathrm{Bool} \\ \Psi; \Xi; \Gamma \vdash val_t : U(\overline{a_t} \longrightarrow FA) \\ \forall val\,A.\ (val, l, A) \in \overline{a_t} \Rightarrow \exists \Gamma'. val_c = \mathrm{true} \vdash \Gamma' <: \Gamma \wedge \Psi; \Xi; \Gamma' \vdash val : A \\ \Psi; \Xi; \Gamma \vdash val_f : U(\overline{a_f} \longrightarrow FA) \\ \forall val\,A.\ (val, l, A) \in \overline{a_f} \Rightarrow \exists \Gamma'. val_c = \mathrm{false} \vdash \Gamma' <: \Gamma \wedge \Psi; \Xi; \Gamma' \vdash val : A\end{array}}{\Psi; \Xi; \Gamma; l \vdash \varnothing; \textbf{br}\ val_c\ val_t\ val_f}$$

Value typing:

$$\boxed{\Psi; \Xi; \Gamma \vdash val : A}$$

$$\frac{(x : A) \in \Gamma}{\Psi; \Xi; \Gamma \vdash x : A} \qquad \frac{(l : \underline{B}) \in \Xi}{\Psi; \Xi; \Gamma \vdash l : U\underline{B}}$$

Instructions we currently use: **br**, **ret**, **icmp**
store, **load**, **alloca**, **getelementptr**, **call**.
How to deal with existential types>? We can't have term witness.s

6.3 HA-CBPV to Vellvm: Type translation

$$\mathrm{Int}^+ = \mathbf{i64}$$

$$\mathrm{Bool}^+ = \mathbf{i64}$$

$$\mathrm{ptr}_V((n, \overline{A}))^+ = \{\mathbf{i32}, \overline{A^+}\} *$$

$$\mathrm{ptr}_V(\overline{(n, \overline{A})})^+ = \{\mathbf{i32}, \overline{\mathbf{i8}}^{\mathrm{sz}(\overline{(n,\overline{A})})}\} *$$

$$\mathrm{ptr}_M(\underline{B})^+ = \underline{B}^+ *$$

$$(\exists \alpha.\, A)^+ = A[\mathbf{void} * / \alpha]^+$$

$$(A_1 \to \ldots \to A_n \to FA)^+ = A^+(A_1{}^+, \ldots, A_n{}^+)$$

6.4 HA-CBPV to Vellvm: Value translation

Values are, at this point, quite simple, and all register sized. (Note that pack has no runtime representation, so they're solidly just integers/booleans/text pointers; with some heap pointers to come at runtime.)

The value translation judgment takes the form

$$\boxed{\Psi; \Gamma; \rho \vdash V : A \rightsquigarrow \mathit{val}}$$

which means that under environments Γ (identifiers to types), Ψ (labels to function types), ρ (identifiers to llvm values), the value V with type A is translated to the llvm value val

$$\frac{x : A \in \Gamma}{\Psi; \Gamma; \rho \vdash x : A \rightsquigarrow \rho(x)} \qquad \frac{}{\Psi; \Gamma; \rho \vdash n : \mathrm{Int} \rightsquigarrow \mathbf{i64}\ n} \qquad \frac{}{\Psi; \Gamma; \rho \vdash \mathrm{true} : \mathrm{Bool} \rightsquigarrow \mathbf{i64}\ 1}$$

$$\frac{}{\Psi; \Gamma; \rho \vdash \mathrm{false} : \mathrm{Bool} \rightsquigarrow \mathbf{i64}\ 0} \qquad \frac{}{\Psi; \Gamma; \rho \vdash \mathrm{thunk}\ \ell : U\underline{B} \rightsquigarrow \ell}$$

$$\frac{\Psi; \Gamma; \rho \vdash V : A'[A/x] \rightsquigarrow \mathit{val}}{\Psi; \Gamma; \rho \vdash \mathrm{pack}\ (A, V)\ \mathrm{as}\ \exists x.\, A' : \exists x.\, A' \rightsquigarrow \mathbf{bitcast}\ \mathit{val}\ \mathbf{to}\ (\exists x.\, A')^+}$$

6.5 HA-CBPV to Vellvm: Computation translation

The module toplevel (see next section) invokes a judgment which converts a top-level M-term that was stored in the code heap into an LLVM definition. That judgment has the form

$$\boxed{\Psi \vdash M \rightsquigarrow_\ell \mathit{prod}}$$

and is defined by

$$\frac{\mathcal{S}; \mathit{args} = \mathrm{args}(\Psi[\ell]) \qquad \Psi, \bullet, \bullet, \mathcal{S} \vdash M \rightsquigarrow c; \mathit{tmn}; \overline{b}; \mathrm{a} \qquad \mathrm{fresh}(l)}{\Psi \vdash M \rightsquigarrow_\ell \mathbf{define}\ \Psi[\ell]^+\ \ell(\mathit{args})\ \{\mathrm{mkBlock}(l, c, \mathit{tmn}, \mathrm{a}), \overline{b}\}}$$

The toplevel that we get from above layers is the function store produced as part of the above translation into heap, containing every thunked comptutation. Currently, each such thunked computation will be translated into an LLVM function definition, containing parameter declarations inferred from its type and a body, which is a collection of labelled blocks, each of which itself contains a number a commands and ends with a terminator (control-flow instruction).

The translation uses an argument stack a la that draft by steve and co for various things.

The judgment we use in this section is of the form:

$$\boxed{\Psi; \Gamma; \rho; \mathcal{S} \vdash M \rightsquigarrow c; tmn; \overline{b}; \mathrm{a}}$$

$$\boxed{\Psi; \Gamma; \rho; \mathcal{S} \vdash M : \underline{B} \rightsquigarrow \overline{c}; tmn; \overline{b}; \mathrm{a}}$$

which has a mildly unreasonble number of positions:

- Γ the usual typing environment
- Ψ a function typing environment, mapping text segment addresses to types
- ρ a mapping from identifiers to llvm-values (either identifiers (i.e. registers) or constants)
- $\mathcal{S}$ analogous to the "argument stack" in the draft by zdancewic, rizkallah, and company. although, this one will always be full when a lambda is run into (unless something has gone deeply deeply wrong!). The argument stack indicates to a lambda where it shoud retrive its arguments from (note that it is always full, even in the case of a top-level function, since the function definition has already acquired arguments via whatever platform calling convention exists and emplaced them into registers), and also where returns should place their results (a register or returned via platform calling convention). The translation of force, which will always be a call/tail instruction, also uses it, for the obvious reasons.
- M The computation that is being translated
- c a sequence of commands, which will effect this computation, when combined with
- tmn a final terminator instruction
- $\overline{b}$ further blocks which are required in order to carry out this computation
- a allocas which are necessary in order to carry out this computation; collected while generating the other three. this is because mem2reg only does the right thing when all allocas are promoted to the entry block.

For the grammars of c, tmn, and $\overline{b}$, see the vellvm grammar. For the grammars of Γ, Ψ, and M, see previous sections.

Argument stacks $\quad \mathcal{S} \;::=\; \mathrm{ret}(\bullet) \mid \mathrm{ret}(l, x) \mid val, \mathcal{S}$

Stack allocations $\quad \mathrm{a} \;::=\; \overline{x \mapsto A}$

Todo: figure out how to point out to llvm that the second force there is a tail call.

$$\frac{\Psi;\Gamma;\rho \vdash V : U\underline{B} \rightsquigarrow val}{\begin{array}{l}\Psi;\Gamma;\rho; val_1,\ldots,val_n, \mathrm{ret}(l,z)\\ \ \vdash \mathrm{force}\ V : A_1 \to \ldots \to A_n \to FA\\ \rightsquigarrow y = \mathbf{call}\ A^+\ val\ ((A_1^+,\emptyset,val_1),\ldots,(A_n^+,\emptyset,val_n)), \mathbf{store}\ A^+\ y\ z; \mathbf{br}\ l; \varnothing; \varnothing\end{array}}$$

$$\frac{\Psi;\Gamma;\rho \vdash V : U\underline{B} \rightsquigarrow val}{\begin{array}{l}\Psi;\Gamma;\rho; val_1,\ldots,val_n, \mathrm{ret}(\bullet)\\ \ \vdash \mathrm{force}\ V : A_1 \to \ldots \to A_n \to FA\\ \rightsquigarrow y = \mathbf{call}\ A^+\ val\ ((A_1^+,\emptyset,val_1),\ldots,(A_n^+,\emptyset,val_n)); \mathbf{ret}\ y; \varnothing; \varnothing\end{array}}$$

$$\frac{\Psi;\Gamma;\rho \vdash V : A \rightsquigarrow val}{\Psi;\Gamma;\rho; \mathrm{ret}(\bullet); \mathrm{ret}\ V : FA \rightsquigarrow \varnothing; \mathbf{ret}\ val; \varnothing; \varnothing}$$

$$\frac{\Psi;\Gamma;\rho \vdash V : A \rightsquigarrow val}{\Psi;\Gamma;\rho; \mathrm{ret}(l,z) \vdash \mathrm{ret}\ V : FA \rightsquigarrow \mathbf{store}\ A^+\ val\ z; \mathbf{br}\ l; \varnothing; \varnothing}$$

$$\frac{\begin{array}{c}\mathrm{fresh}(l) \qquad \mathrm{fresh}(y) \qquad \mathrm{fresh}(z)\\ \Psi;\Gamma;\rho;\mathrm{ret}(l,y) \vdash M : FA \rightsquigarrow \overline{c_M}; tmn_M; \overline{b_M}; \mathrm{a}_M\\ \Psi;\Gamma,(x:A);\rho[x \mapsto z];\mathcal{S} \vdash N : \underline{B} \rightsquigarrow \overline{c_N}; tmn_N; \overline{b_N}; \mathrm{a}_N\end{array}}{\Psi;\Gamma;\rho;\mathcal{S} \vdash x \leftarrow M;\ N : \underline{B} \rightsquigarrow \overline{c_M}; tmn_M; \mathrm{mkBlock}(l, (z = \mathbf{load}\ A^+\ y, \overline{c_N}), tmn, \varnothing), \overline{b_M}, \overline{b_N}; y \mapsto A^+, \mathrm{a}_M, \mathrm{a}_N}$$

$$\frac{\Psi;\Gamma;\rho \vdash V : A \rightsquigarrow val \qquad \Psi;\Gamma;\rho; val, \mathcal{S} \vdash M : A \to \underline{B} \rightsquigarrow \overline{c}; tmn; \overline{b}; \mathrm{a}}{\Psi;\Gamma;\mathcal{S} \vdash M\ V : \underline{B} \rightsquigarrow \overline{c}; tmn; \overline{b}; \mathrm{a}}$$

$$\frac{\Psi;\Gamma[x \mapsto A];\rho[x \mapsto val];\mathcal{S} \vdash M : \underline{B} \rightsquigarrow \overline{c}; tmn; \overline{b}; \mathrm{a}}{\Psi;\Gamma;\rho; val, \mathcal{S} \vdash \lambda(x:A).\ M : A \to \underline{B} \rightsquigarrow \overline{c}; tmn; \overline{b}; \mathrm{a}}$$

$$\frac{\begin{array}{c}\forall i, \Psi;\Gamma;\rho \vdash V_i : A_i \rightsquigarrow val_i\\ \Psi;\Gamma,(x:\mathrm{ptr}_V((n,A_1,\ldots,A_m)));\rho[x \mapsto x];\mathcal{S} \vdash M : \underline{B} \rightsquigarrow \overline{c}; tmn; \overline{b}; \mathrm{a}\end{array}}{\begin{array}{l}\Psi;\Gamma;\rho;\mathcal{S}\\ \ \vdash \mathrm{let}\ x = \mathrm{alloc}_n(V_1,\ldots,V_m)\ \mathrm{in}\ M : \underline{B}\\ \rightsquigarrow x = \mathbf{malloc}\ (\mathrm{ptr}_V((n,A_1,\ldots,A_m)))^+\ \mathrm{sz}(\mathrm{ptr}_V((n,A_1,\ldots,A_m)))\ 0, \overline{c}; tmn; \overline{b}; \mathrm{a}\end{array}}$$

$$\frac{\begin{array}{c}\mathrm{fresh}(y) \qquad \mathrm{fresh}(z)\\ \Psi;\Gamma;\rho \vdash V : \mathrm{ptr}_V((n,A_1,\ldots,A_n)) \rightsquigarrow val \qquad \Psi;\Gamma,(x:A_i);\rho[x \mapsto z];\mathcal{S} \vdash M : \underline{B} \rightsquigarrow \overline{c}; tmn; \overline{b}; \mathrm{a}\end{array}}{\begin{array}{l}\Psi;\Gamma;\rho;\mathcal{S}\\ \ \vdash \mathrm{let}\ x = \mathrm{prj}_i(V)\ \mathrm{in}\ M : \underline{B}\\ \rightsquigarrow y = \mathbf{getelementptr}\ \mathrm{ptr}_V((n,A_1,\ldots,A_n))^+\ val\ 0\ i, z = \mathbf{load}\ ({A_i}^+) *\ y\ 0, \overline{c}; tmn; \overline{b}; \mathrm{a}\end{array}}$$

$$\frac{\begin{array}{c}\mathrm{fresh}(l_M) \qquad \mathrm{fresh}(l_N) \\ \Psi; \Gamma; \rho \vdash V : A \rightsquigarrow \mathit{val} \\ \Psi; \Gamma; \rho; \mathcal{S} \vdash M : \underline{B} \rightsquigarrow \overline{c_M}; \mathit{tmn}_M; \overline{b_M}; \mathsf{a}_M \qquad \Psi; \Gamma; \rho; \mathcal{S} \vdash N : \underline{B} \rightsquigarrow \overline{c_N}; \mathit{tmn}_N; \overline{b_N}; \mathsf{a}_N\end{array}}{\begin{array}{l}\Psi; \Gamma; \rho; \mathcal{S} \\ \;\vdash \text{if } V \text{ then } M \text{ else } N : \underline{B} \\ \rightsquigarrow \varnothing; \mathbf{br}\ \mathit{val}\ l_M\ l_N; \mathrm{mkBlock}(M, \overline{c_M}, \mathit{tmn}_M, \varnothing), \mathrm{mkBlock}(M, \overline{c_M}, \mathit{tmn}_M, \varnothing), \overline{b_M}, \overline{b_N}; \mathsf{a}_M, \mathsf{a}_N\end{array}}$$

$$\frac{\begin{array}{c}A = \mathrm{ptr}_V((n_1, A_{1,1}, \ldots, A_{1,o_1}), \ldots, (n_m, A_{m,1}, \ldots, A_{m,o_m})) \qquad \Psi; \Gamma; \rho \vdash V : A \rightsquigarrow \mathit{val} \\ \forall i, \mathrm{fresh}(\mathit{lc}_i) \wedge \mathrm{fresh}(l_i) \wedge \mathrm{fresh}(c_i) \wedge \mathit{cb}_i = \mathrm{mkBlock}(\mathit{lc}_i, c_i = \mathbf{icmp\ eq\ i32}\ z\ (\mathbf{i32}\ i), \mathbf{br}\ c_i\ l_i\ \mathit{lc}_{i+1}, \varnothing) \\ \forall i, \Psi; \Gamma, (x_i : \mathrm{ptr}_V((n_i, A_{i,1}, \ldots, A_{i,o_i}))); \rho[x \mapsto \mathit{val}]; \mathcal{S} \vdash M_i : \underline{B} \rightsquigarrow \overline{c_i}; \mathit{tmn}_i; \overline{b_i}; \mathsf{a}_i \\ \wedge \mathit{bb}_i = \mathrm{mkBlock}(l_i, \overline{c_i}, \mathit{tmn}_i, \varnothing)\end{array}}{\begin{array}{l}\Psi; \Gamma; \rho; \mathcal{S} \\ \;\vdash \text{case } V\ \{n_1(x_1) \Rightarrow M_1; , \ldots, n_m(x_m) \Rightarrow M_m; \} : \underline{B} \\ \rightsquigarrow y = \mathbf{getelementptr}\ A^+\ \mathit{val}\ 0\ i, z = \mathbf{load}\ (\mathbf{i32}) *\ y; \mathbf{br}\ \mathit{lc}_1; \overline{b_1}, \ldots, \overline{b_m}, \mathit{cb}_1, \ldots, \mathit{cb}_m, \mathit{bb}_1, \ldots, \mathit{bb}_m; \mathsf{a}_1, \ldots, \mathsf{a}_m\end{array}}$$

$$\frac{\Psi; \Gamma; \rho \vdash V : \exists \beta.\, A \rightsquigarrow \mathit{val} \qquad \Psi; \Gamma, (x : A[\alpha/\beta]); \rho[x \mapsto \mathit{val}]; \mathcal{S} \vdash M : \underline{B} \rightsquigarrow \overline{c}; \mathit{tmn}; \overline{b}; \mathsf{a}}{\Psi; \Gamma; \rho; \mathcal{S} \vdash \text{unpack } (\alpha, x) = V \text{ in } M : \underline{B} \rightsquigarrow \overline{c}; \mathit{tmn}; \overline{b}; \mathsf{a}}$$

todo: use lookup table or equivalent for case.
Typing however is hard.

6.6 HA-CBPV to Vellvm: Module translation

$$\frac{\forall i, \Psi, \bullet \vdash M_i \rightsquigarrow \mathit{prod}_i}{\vdash \ell_1 \mapsto M_1, \ldots, \ell_n \mapsto M_n : \Psi \rightsquigarrow \varnothing\ \varnothing\ (\mathit{prod}_1, \ldots, \mathit{prod}_n)}$$

The LLVM perfomance tips for frontend authors stresses that it is quite important to put in the data layout and target triple stuff, so that llvm target-specific optimizations can run. We don't do that yet, since we kind of have to pull them out of thin air/the compilation environment, and because it's not *entirely* clear how to put them into vellvm (well the layout entries do exist in vellvm, although they look awfully complex.)

Note that we currently have Int as a base type in all of the languages. This becomes a specific word size by the time that we hit the target here, and we just sort of choose one indiscriminately. This does severely affect the semantics of the high level integer type, so perhaps it should either be replaced with sized ints all the way up the stack (almost certainly preferable), or some type of arbitrary precision arithmetic introduction pass should be inserted (seems less preferable; also requires either writing our own implementation of arbitrary precision arithmetic in the compiler, which probably wouldn't be very good, or in trusting/annotating the semantics of gmp or something.)

REFERENCES

Who knows. 2022. bibtex insists on a reference entry existing. *Journal of Software Defect Workarounds* 1, 1 (2022), 1.