プログラミング言語周りノート

-

2023 年 12 月 7 日

目次

第 1 章

Preliminaries

1.1　基本的な表記

量化子 (*quantifier*) の束縛をコンマ (,) で続けて書く．束縛の終わりをピリオド (.) で示す．例えば，

$$\forall x_1 \in X_1, x_2 \in X_2.\, \exists y_1 \in Y_1, y_2 \in Y_2.\, x_1 = y_1 \wedge x_2 = y_2$$

は，

$$\forall x_1 \in X_1.\, \forall x_2 \in X_2.\, \exists y_1 \in Y_1.\, \exists y_2 \in Y_2.\, x_1 = y_1 \wedge x_2 = y_2$$

と等しい．また，量化子の束縛において，*such that* を省略し，コンマ (,) で繋げて書く．例えば，

$$\forall x \in \{0, 1\}, x \neq 0.\, x = 1$$

は，

$$\forall x \in \{0, 1\}.\, x \neq 0 \implies x = 1$$

と等しい．また，$\implies$，$\iff$ が他の記号と混同する場合，それぞれ implies，iff を使用する．

集合 (*set*) について，以下の表記を用いる．

- 集合 A について，その**濃度** (*cardinality*) を $|A|$ と表記する．なお，A が**有限集合** (*finite set*) の時，濃度とは要素の個数のことである．
- 集合 A について，$a \in A$ を $a : A$ と表記する．
- **自然数** (*natural number*) の集合を $\mathbb{N} = \{0, 1, \dots\}$ と表記する．また，n 以上の自然数の集合を $\mathbb{N}_{\geq n} = \{n, n+1, \dots\}$ と表記する．
- 自然数 $n \in \mathbb{N}$ について，$\{1, \dots, n\}$ を $[n]$ と表記する．
- 集合 A の**冪集合** (*power set*) を $\mathcal{P}(A) = \{X \mid X \subseteq A\}$，有限冪集合を $\mathcal{P}_{fin}(A) = \{X \in \mathcal{P}(X) \mid X \text{は有限集合}\}$ と表記する．
- 集合 $A_1, \dots, A_n$ の**直積** (*cartesian product*) を $A_1 \times \cdots \times A_n = \{(a_1, \dots, a_n) \mid a_1 \in A_1, \dots, a_n \in A_n\}$ と表記する．集合 A の n 直積を $A^n = \underbrace{A \times \cdots \times A}_{n\text{ 項}}$ と表記する．特に，$A^0 = \{\epsilon\}$ である．
- 集合 $A_1, \dots, A_n$ の**直和** (*disjoin union*) を $A_1 \uplus \cdots \uplus A_n = (A_1 \times \{1\}) \cup \cdots (A_n \times \{n\})$ と表記する．なお，文脈から明らかな場合，直和の添字を省略し，$a \in A_i$ に対して，$a \in A_1 \uplus \cdots \uplus A_n$ と表記する．
- 集合 A の B との**差集合** (*relative complement*) を $A \setminus B = \{a \in A \mid a \notin B\}$ と表記する．

集合 Σ について，$\bigcup_{n \in \mathbb{N}} \Sigma^n$ を Σ^* と表記する．この時，$\alpha \in \Sigma^*$ を Σ による**列** (*sequence*) と呼ぶ．列について，以下の表記を用いる．

- $(\sigma_1, \dots, \sigma_n) \in \Sigma^n$ について，$(\sigma_1, \dots, \sigma_n)$ を $\sigma_1 \cdots \sigma_n$ と表記する．
- 列 $\alpha = \sigma_1 \cdots \sigma_n \in \Sigma^*$ について，その長さを $|\alpha| = n$ と表記する．

集合 A, B について，$R \subseteq A \times B$ を**関係** (*relation*) と呼ぶ．また，

$$A \rightharpoonup B \stackrel{\text{def}}{=} \{R \in \mathcal{P}(A \times B) \mid \forall x \in A, (x, y_1), (x, y_2) \in R.\, y_1 = y_2\}$$

という表記を導入し，関係 $f : A \rightharpoonup B$ を A から B への**部分関数** (*partial function*) と呼ぶ．さらに，

$$A \to B \stackrel{\text{def}}{=} \{f : A \rightharpoonup B \mid \forall x \in A.\, \exists y \in B.\, (x, y) \in f\}$$

という表記を導入し，部分関数 $f : A \to B$ を **(全) 関数** (*function*) と呼ぶ．関係について，以下の表記を用いる．

- 関係 $R \subseteq A \times B$ について，$(a, b) \in R$ を $a\ R\ b$ と表記する．
- 関係 $R \subseteq A \times B$ について，**定義域** (*domain*) を $\mathrm{dom}(R) = \{a \mid \exists b.\, (a, b) \in R\}$，**値域** (*range*) を $\mathrm{cod}(R) = \{b \mid \exists a.\, (a, b) \in R\}$ と表記する．

- 部分関数 $f : A \rightharpoonup B$ について，$(a,b) \in f$ を $f(a) = b$ と表記する．
- 関係 $R_1 \subseteq A \times B$，$R_2 \subseteq B \times C$ について，その**合成** (*composition*) を $R_1 ; R_2 = R_2 \circ R_1 = \{(x,z) \in A \times C \mid \exists y \in B. (x,y) \in R_1, (y,z) \in R_2\}$ と表記する．
- 関係 $R \subseteq A \times B$，集合 $X \subseteq A$ について，R の X による**制限** (*restriction*) を $R \restriction_X = \{(a,b) \in R \mid a \in X\}$ と表記する．特に関数 $f : A \to B$ の $X \subseteq A$ による制限は，関数 $f \restriction_X : X \to B$ になる．
- $a \in A$，$b \in B$ について，その組を $a \mapsto b = (a,b)$，関数 $f : A \to B$ を $f = x \mapsto f(x)$ と表記する．
- 2 項関係 $R \subseteq A^2$ について，その**推移閉包** (*transitive closure*)，つまり以下を満たす最小の 2 項関係を $R^+ \subseteq A^2$ と表記する．
  - 任意の $(a,b) \in R$ について，$(a,b) \in R^+$．
  - 任意の $(a,b) \in R^+$，$(b,c) \in R^+$ について，$(a,c) \in R^+$．
- 2 項関係 $R \subseteq A^2$ について，その**反射推移閉包** (*reflexive transitive closure*) を $R^* = R^+ \cup \{(a,a) \mid a \in A\}$ と表記する．

集合 I について，その要素で添字付けられた対象の列 $\{a_i\}_{i \in I}$ を I で添字づけられた**族** (*indexed family*) と呼ぶ．族について，以下の表記を用いる．

- 族の集合を $\prod_{i \in I} A_i = \{\{a_i\}_{i \in I} \mid \forall i \in I, a_i \in A_i\}$ と表記する．
- 集合の族 $A = \{A_i\}_{i \in I}$ について，次の条件を満たす時，A は**互いに素** (*pairwise disjoint*) であるという．

$$\forall i_1, i_2 \in I, i_1 \neq i_2. A_{i_1} \cap A_{i_2} = \varnothing$$

1.2　基本的な定義

定義 1 (ランク付きアルファベット (ranked alphabet)). ランク付きアルファベットとは，以下の組 (Σ, rank) のこと．

- 集合 Σ．
- 関数 $\mathrm{rank} : \Sigma \to \mathbb{N}$．

rank が文脈から明らかな時，単に Σ をランク付きアルファベットと呼ぶ．$f \in \Sigma$ について，$\mathrm{rank}(f) = n$ の時，f は n-変数であるという．これを明示して，$f^{(n)}$ と表記することもある．□

定義 2 (項代数 (term algebra)). 項代数 $\mathcal{T}$ とは，以下の組 (Σ, X) のこと．

- ランク付きアルファベット Σ．
- 変数の集合 X．

この時，$[\![\mathcal{T}]\!]$ を以下を満たす最小の集合として定義する．

- $X \subseteq [\![\mathcal{T}]\!]$．
- $\tau_1, \dots, \tau_n \in [\![\mathcal{T}]\!]$，$f^{(n)} \in \Sigma$ について，$f(\tau_1, \dots, \tau_n) \in [\![\mathcal{T}]\!]$．

この時，$\tau \in [\![\mathcal{T}]\!]$ を $\mathcal{T}$ の項と呼ぶ．□

定義 3 (パス (path)). 項代数 $\mathcal{T} = (\Sigma, X)$ について，$\mathrm{paths} : [\![\mathcal{T}]\!] \to \mathcal{P}(\mathrm{N}^*)$ を以下のように定義する．

- $x \in X$ について，$\mathrm{paths}(x) = \{\epsilon\}$．
- $f^{(n)} \in \Sigma$，$f^{(n)}(\tau_1, \dots, \tau_n) \in [\![\mathcal{T}]\!]$ について，$\mathrm{paths}(f^{(n)}(\tau_1, \dots, \tau_n)) = \{\epsilon\} \cup \bigcup_{i \in [n]} \{i\pi \in \mathrm{paths}(\tau_i)\}$．

この時，$\pi \in \mathrm{paths}(\tau)$ を τ のパスと呼ぶ．□

定義 4 (部分項 (subterm)). 項代数 $\mathcal{T} = (\Sigma, X)$，項 $\tau \in [\![\mathcal{T}]\!]$ について，$\mathrm{subterm}_\tau : \mathrm{paths}(\tau) \to [\![\mathcal{T}]\!]$ を以下のように定義する．

- $\mathrm{subterm}_\tau(\epsilon) = \tau$．

- $\mathrm{subterm}_{f^{(n)}(\tau_1,\ldots,\tau_i,\ldots,\tau_n)}(i\pi) = \mathrm{subterm}_{\tau_i}(\pi)$.

この時，$\pi \in \mathrm{paths}(\tau)$ について，$\mathrm{subterm}_\tau(\pi)$ を τ の π での部分木と呼ぶ. □

定義 5 (置換 (substitution)). 項代数 $\mathcal{T} = (\Sigma, X)$ について，$\sigma \subseteq [\![\mathcal{T}]\!] \times [\![\mathcal{T}]\!]$ が置換とは，以下を満たすことを言う.

- 任意の $x \in \mathrm{dom}(\sigma)$ について，$(x, y_1), (x, y_2) \in \sigma$ ならば $y_1 = y_2$.
- 任意の $x_1, x_2 \in \mathrm{dom}(\sigma)$ について，$\mathrm{subterm}_{x_1}(\pi) = x_2$ となる $\pi \in \mathrm{paths}(x_1)$ は存在しない.

□

定義 6 (出現 (occurence)). 項代数 $\mathcal{T} = (\Sigma, X)$，項 $\tau \in [\![\mathcal{T}]\!]$ について，$\mathrm{occ}_\tau : [\![\mathcal{T}]\!] \to \mathcal{P}(\mathrm{paths}(\tau))$ を以下のように定義する.

$$\mathrm{occ}_\tau(\eta) = \{\pi \in \mathrm{paths}(\tau) \mid \mathrm{subterm}_\tau(\pi) = \eta\}$$

この時，$\pi \in \mathrm{occ}_\tau(\eta)$ を，η の τ での出現と呼ぶ. □

定義 7 (コンテキスト (context)). 項代数 $\mathcal{T} = (\Sigma, X)$ について，コンテキストとは，$T[] \in [\![(\Sigma, X \uplus \{[]\})]\!]$ で $[]$ の出現が一意であるもののことを言う．この時，$\mathcal{T}$ のコンテキストの集合を $\mathcal{C}(\mathcal{T})$ と書く.

この時，$\tau \in [\![\mathcal{T}]\!]$ について，$T[\tau] \in [\![\mathcal{T}]\!]$ を $T[\tau] = (T[])[[] \leftarrow \tau]$ で定義する. □

第 2 章

Basic Calculus

2.1 WIP: (Untyped) λ-Calculus

2.2 Simply Typed λ-Calculus

Alias: STLC, $\lambda^{\rightarrow}$ [GTL89]

2.2.1 Syntax

$$
\begin{array}{rcll}
e & ::= & x & \text{(variable)} \\
 & | & e\ e & \text{(application)} \\
 & | & \lambda x : \tau .\, e & \text{(abstraction)} \\
 & | & c_A & \text{(constant)} \\
\tau & ::= & A & \text{(atomic type)} \\
 & | & \tau \rightarrow \tau & \text{(function type)} \\
\Gamma & ::= & \cdot & \text{(empty)} \\
 & | & \Gamma, x : \tau & \text{(cons)}
\end{array}
$$

Convention:

$$\tau_1 \rightarrow \tau_2 \rightarrow \cdots \rightarrow \tau_n \stackrel{\text{def}}{=} \tau_1 \rightarrow (\tau_2 \rightarrow (\cdots \rightarrow \tau_n)\cdots)$$
$$e_1\ e_2\ \cdots\ e_n \stackrel{\text{def}}{=} (\cdots (e_1\ e_2)\cdots)\ e_n)$$

Environment Reference:
$\boxed{\Gamma(x) = \tau}$

$$\frac{x = x'}{(\Gamma, x' : \tau)(x) = \tau} \qquad \frac{x \neq x' \quad \Gamma(x) = \tau}{(\Gamma, x' : \tau')(x) = \tau}$$

Free Variable:
$\boxed{fv(e) = \{\overline{x'}\}}$

$$\frac{}{fv(x) = \{x\}} \qquad \frac{fv(e_1) = X_1 \quad fv(e_2) = X_2}{fv(e_1\ e_2) = X_1 \cup X_2} \qquad \frac{fv(e) = X}{fv(\lambda x : \tau .\, e) = X \setminus \{x\}} \qquad \frac{}{fv(c_A) = \varnothing}$$

Substitution:
部分関数 $\{x_1 \mapsto e_1, \dots, x_n \mapsto e_n\}$ を，$[x_1 \leftarrow e_1, \dots, x_n \leftarrow e_n]$ または $[x_1, \dots, x_n \leftarrow e_1, \dots, e_n]$ と表記する．
$\boxed{e[\overline{x'} \leftarrow \overline{e'}] = e''}$

$$\frac{[\overline{x'} \leftarrow \overline{e'}](x) = e}{x[\overline{x'} \leftarrow \overline{e'}] = e} \qquad \frac{x \notin \mathrm{dom}([\overline{x'} \leftarrow \overline{e'}])}{x[\overline{x'} \leftarrow \overline{e'}] = x}$$

$$\frac{e_1[\overline{x'} \leftarrow \overline{e'}] = e_1'' \quad e_2[\overline{x'} \leftarrow \overline{e'}] = e_2''}{(e_1\ e_2)[\overline{x'} \leftarrow \overline{e'}] = e_1''\ e_2''} \qquad \frac{e([\overline{x'} \leftarrow \overline{e'}] \restriction_{\mathrm{dom}([\overline{x'} \leftarrow \overline{e'}]) \setminus \{x\}}) = e''}{(\lambda x : \tau .\, e)[\overline{x'} \leftarrow \overline{e'}] = \lambda x : \tau .\, e''} \qquad \frac{}{c_A[\overline{x'} \leftarrow \overline{e'}] = c_A}$$

α-Equality:
$\boxed{e_1 \equiv_\alpha e_2}$

$$\frac{x_1 = x_2}{x_1 \equiv_\alpha x_2} \qquad \frac{x' \notin fv(e_1) \cup fv(e_2) \quad e_1[x_1 \leftarrow x'] \equiv_\alpha e_2[x_2 \leftarrow x']}{\lambda x_1 : \tau .\, e_1 \equiv_\alpha \lambda x_2 : \tau .\, e_2} \qquad \frac{e_1 \equiv_\alpha e_2 \quad e_1' \equiv_\alpha e_2'}{e_1\ e_1' \equiv_\alpha e_2\ e_2'} \qquad \frac{}{c_A \equiv_\alpha c_A}$$

定理 8 (Correctness of Substitution). 式 e，置換 $[\overline{x'} \leftarrow \overline{e'}]$ について，$X = \mathrm{dom}([\overline{x'} \leftarrow \overline{e'}])$ とした時，

$$fv(e[\overline{x'} \leftarrow \overline{e'}]) = (fv(e) \setminus X) \cup \bigcup_{x \in fv(e) \cap X} fv([\overline{x'} \leftarrow \overline{e'}](x)).$$

□

定理 9 (α-Equality Does Not Touch Free Variables). $e_1 \equiv_\alpha e_2$ ならば，$fv(e_1) = fv(e_2)$. □

2.2.2 Typing Semantics

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

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ \text{T-Var}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1 .\, e : \tau_1 \to \tau_2}\ \text{T-Abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau}\ \text{T-App}$$

$$\frac{}{\Gamma \vdash c_A : A}\ \text{T-Const}$$

特に，$\cdot \vdash e : \tau$ の時，$e : \tau$ と表記.

2.2.3 Evaluation Semantics (Call-By-Value)

$$\begin{array}{rcl} v & ::= & \lambda x : \tau .\, e \\ & \mid & c_A \\ C & ::= & [] \\ & \mid & C\ e \\ & \mid & v\ C \end{array}$$

Small Step:

$\boxed{e \Rightarrow e'}$

$$\frac{}{(\lambda x : \tau .\, e)\ v \Rightarrow e[x \leftarrow v]}$$

$$\frac{e \Rightarrow e'}{C[e] \Rightarrow C[e']}$$

Big Step:

$\boxed{e \Downarrow v}$

$$\frac{e_1 \Downarrow \lambda x : \tau .\, e_1' \quad e_2 \Downarrow v_2 \quad e_1'[x \leftarrow v_2] \Downarrow v}{e_1\ e_2 \Downarrow v}$$

定理 10 (Adequacy of Small Step and Big Step). $e \Rightarrow^* v$ iff $e \Downarrow v$. □

定理 11 (Type Soundness). $e : \tau$ の時，$e \Rightarrow^* v$，$e \Downarrow v$ となる $v = \mathrm{nf}(\Rightarrow, e)$ が存在し，

- $\tau = \tau_1 \to \tau_2$ の時，$v \equiv_\alpha \lambda x' : \tau_1 .\, e'$ となる $\lambda x' : \tau' .\, e'$ が存在する．
- $\tau = A$ の時，$v \equiv_\alpha c_A$ となる c_A が存在する．

□

2.2.4 Equational Reasoning

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau}$

$$\frac{\Gamma, x : \tau \vdash e_1 : \tau_2 \to \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\lambda x : \tau.\, e_1)\; e_2 \equiv e_1[x \leftarrow e_2] : \tau}\ \text{Eq-}\beta\text{-Lam} \qquad \frac{x \notin fv(e) \quad \Gamma \vdash e : \tau_1 \to \tau_2}{\Gamma \vdash (\lambda x : \tau_1.\, e\; x) \equiv e : \tau_1 \to \tau_2}\ \text{Eq-}\eta\text{-Lam}$$

$$\frac{e_1 \equiv_\alpha e_2 \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau}\ \text{Eq-}\alpha\text{-Refl}$$

$$\frac{\Gamma \vdash e_2 \equiv e_1 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau}\ \text{Eq-Sym} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau \quad \Gamma \vdash e_2 \equiv e_3 : \tau}{\Gamma \vdash e_1 \equiv e_3 : \tau}\ \text{Eq-Trans}$$

$$\frac{\Gamma, x : \tau \vdash e_1 \equiv e_2 : \tau'}{\Gamma \vdash \lambda x : \tau.\, e_1 \equiv \lambda x : \tau.\, e_2 : \tau \to \tau'}\ \text{Eq-Cong-Abs} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau' \to \tau \quad \Gamma \vdash e_1' \equiv e_2' : \tau'}{\Gamma \vdash e_1\; e_1' \equiv e_2\; e_2' : \tau}\ \text{Eq-Cong-App}$$

特に，$\cdot \vdash e_1 \equiv e_2 : \tau$ の時，$e_1 \equiv e_2 : \tau$ と表記.

定理 12 (Respect Typing). $\Gamma \vdash e_1 \equiv e_2 : \tau$ ならば，$\Gamma \vdash e_1 : \tau$ かつ $\Gamma \vdash e_2 : \tau$. □

定理 13 (Respect Evaluation). $e_1 \equiv e_2 : \tau$ の時，$e_1' \Rightarrow^* e_1$，$e_2 \Rightarrow^* e_2'$ ならば $e_1' \equiv e_2' : \tau$. □

系 14. $e_1 \equiv e_2 : \tau$ の時，$e_1 \Rightarrow^* e_1'$，$e_2 \Rightarrow^* e_2'$ ならば $e_1' \equiv e_2' : \tau$. □

証明. $e_1 \Rightarrow^* e_1$ より，定理 13 から $e_1 \equiv e_2' : \tau$. よって，T-Sym から $e_2' \equiv e_1 : \tau$ であり，$e_2' \Rightarrow^* e_2'$ より定理 13 から $e_2' \equiv e_1' : \tau$. 故に，T-Sym から $e_1' \equiv e_2' : \tau$. ∎

2.3 WIP: System-T

2.4 WIP: PCF

2.5 System-F

Alias: F，Second Order Typed Lambda Calculus，$\lambda 2$ [GTL89]

2.5.1 Syntax

$$
\begin{array}{rcll}
e & ::= & x & \text{(variable)} \\
 & | & \lambda x : \tau.\, e & \text{(abstraction)} \\
 & | & e\ e & \text{(application)} \\
 & | & \Lambda t.\, e & \text{(universal abstraction)} \\
 & | & e\ \tau & \text{(universal application)} \\
\tau & ::= & t & \text{(type variable)} \\
 & | & \tau \rightarrow \tau & \text{(function type)} \\
 & | & \forall t.\, \tau & \text{(polymorphic type)} \\
\Gamma & ::= & \cdot & \text{(empty)} \\
 & | & \Gamma, x : \tau & \text{(variable cons)} \\
 & | & \Gamma, t : \Omega & \text{(type variable cons)}
\end{array}
$$

Convention:

$$\tau_1 \rightarrow \tau_2 \rightarrow \cdots \rightarrow \tau_n \overset{\text{def}}{=} \tau_1 \rightarrow (\tau_2 \rightarrow (\cdots \rightarrow \tau_n) \cdots)$$

$$e_1\ e_2\ \cdots\ e_n \overset{\text{def}}{=} (\cdots (e_1\ e_2) \cdots)\ e_n)$$

Environment Reference:

$\boxed{\Gamma(x) = \tau}$

$$\frac{x = x'}{(\Gamma, x' : \tau)(x) = \tau} \qquad \frac{x \neq x' \quad \Gamma(x) = \tau}{(\Gamma, x' : \tau')(x) = \tau} \qquad \frac{\Gamma(x) = \tau}{(\Gamma, t : \Omega)(x) = \tau}$$

$$\frac{t = t'}{(\Gamma, t' : \Omega)(t) = \Omega} \qquad \frac{t \neq t' \quad \Gamma(t) = \Omega}{(\Gamma, t' : \Omega')(t) = \Omega} \qquad \frac{\Gamma(t) = \Omega}{(\Gamma, x : \tau)(t) = \Omega}$$

Free Variable:

$\boxed{fv(e) = \{\overline{x}\}}$

$$\frac{}{fv(x) = \{x\}} \qquad \frac{fv(e_1) = X_1 \quad fv(e_2) = X_2}{fv(e_1\ e_2) = X_1 \cup X_2} \qquad \frac{fv(e) = X}{fv(\lambda x : \tau.\, e) = X \setminus \{x\}} \qquad \frac{fv(e) = X}{fv(e\ \tau) = X} \qquad \frac{fv(e) = X}{fv(\Lambda t.\, e) = X}$$

Substitution:

部分関数 $\{x_1 \mapsto e_1, \ldots, x_n \mapsto e_n\}$ を，$[x_1 \leftarrow e_1, \ldots, x_n \leftarrow e_n]$ または $[x_1, \ldots, x_n \leftarrow e_1, \ldots, e_n]$ と表記する．

$\boxed{e[\overline{x'} \leftarrow \overline{e'}] = e''}$

$$\frac{[\overline{x'} \leftarrow \overline{e'}](x) = e}{x[\overline{x'} \leftarrow \overline{e'}] = e} \qquad \frac{x \notin \mathrm{dom}([\overline{x'} \leftarrow \overline{e'}])}{x[\overline{x'} \leftarrow \overline{e'}] = x}$$

$$\frac{e_1[\overline{x'} \leftarrow \overline{e'}] = e_1'' \quad e_2[\overline{x'} \leftarrow \overline{e'}] = e_2''}{(e_1\ e_2)[\overline{x'} \leftarrow \overline{e'}] = e_1''\ e_2''} \qquad \frac{e([\overline{x'} \leftarrow \overline{e'}] \restriction_{\mathrm{dom}([\overline{x'} \leftarrow \overline{e'}]) \setminus \{x\}}) = e''}{(\lambda x : \tau.\, e)[\overline{x'} \leftarrow \overline{e'}] = \lambda x : \tau.\, e''}$$

$$\frac{e[\overline{x'} \leftarrow \overline{e'}] = e''}{(e\ \tau)[\overline{x'} \leftarrow \overline{e'}] = e''\ \tau} \qquad \frac{e[\overline{x'} \leftarrow \overline{e'}] = e''}{(\Lambda t.\, e)[\overline{x'} \leftarrow \overline{e'}] = \Lambda t.\, e''}$$

Type Free Variable:

$\boxed{tyfv(e) = \{\overline{x}\}}$

$$\frac{}{tyfv(x) = \varnothing} \quad \frac{tyfv(e_1) = T_1 \quad tyfv(e_2) = T_2}{tyfv(e_1\ e_2) = T_1 \cup T_2} \quad \frac{tyfv(\tau) = T_1 \quad tyfv(e) = T_2}{tyfv(\lambda x : \tau.\, e) = T_1 \cup T_2}$$

$$\frac{tyfv(e) = T_1 \quad tyfv(\tau) = T_2}{tyfv(e\ \tau) = T_1 \cup T_2} \quad \frac{tyfv(e) = T}{tyfv(\Lambda t.\, e) = T \setminus \{t\}}$$

$$\frac{}{tyfv(t) = \{t\}} \quad \frac{tyfv(\tau_1) = T_1 \quad tyfv(\tau_2) = T_2}{tyfv(\tau_1 \to \tau_2) = T_1 \cup T_2} \quad \frac{tyfv(\tau) = T}{tyfv(\forall t.\, \tau) = T \setminus \{t\}}$$

Type Substitution:

部分関数 $\{t_1 \mapsto \tau_1, \dots, t_n \mapsto \tau_n\}$ を，$[t_1 \leftarrow \tau_1, \dots, t_n \leftarrow \tau_n]$ または $[t_1, \dots, t_n \leftarrow t_1, \dots, t_n]$ と表記する．

$\boxed{e[\overline{t \leftarrow \tau}] = e'}$

$$\frac{}{x[\overline{t'} \leftarrow \overline{\tau'}] = x} \quad \frac{e_1[\overline{t'} \leftarrow \overline{\tau'}] = e_1'' \quad e_2[\overline{t'} \leftarrow \overline{\tau'}] = e_2''}{(e_1\ e_2)[\overline{t'} \leftarrow \overline{\tau'}] = e_1''\ e_2''} \quad \frac{\tau[\overline{t'} \leftarrow \overline{\tau'}] = \tau'' \quad e[\overline{t'} \leftarrow \overline{\tau'}] = e''}{(\lambda x : \tau.\, e)[\overline{t'} \leftarrow \overline{\tau'}] = \lambda x : \tau''.\, e''}$$

$$\frac{e[\overline{t'} \leftarrow \overline{\tau'}] = e'' \quad \tau[\overline{t'} \leftarrow \overline{\tau'}] = \tau''}{(e\ \tau)[\overline{t'} \leftarrow \overline{\tau'}] = e''\ \tau''} \quad \frac{e([\overline{t'} \leftarrow \overline{\tau'}] \restriction_{\mathrm{dom}([\overline{t'} \leftarrow \overline{\tau'}]) \setminus \{t\}}) = e''}{(\Lambda t.\, e)[\overline{t'} \leftarrow \overline{\tau'}] = \Lambda t.\, e''}$$

$\boxed{\tau[\overline{t' \leftarrow \tau'}] = \tau''}$

$$\frac{[\overline{t'} \leftarrow \overline{\tau'}](t) = \tau}{t[\overline{t'} \leftarrow \overline{\tau'}] = \tau} \quad \frac{t \notin \mathrm{dom}([\overline{t'} \leftarrow \overline{\tau'}])}{t[\overline{t'} \leftarrow \overline{\tau'}] = t} \quad \frac{\tau_1[\overline{t'} \leftarrow \overline{\tau'}] = \tau_1'' \quad \tau_2[\overline{t'} \leftarrow \overline{\tau'}] = \tau_2''}{(\tau_1 \to \tau_2)[\overline{t'} \leftarrow \overline{\tau'}] = \tau_1'' \to \tau_2''} \quad \frac{\tau([\overline{t'} \leftarrow \overline{\tau'}] \restriction_{\mathrm{dom}([\overline{t'} \leftarrow \overline{\tau'}]) \setminus \{t\}}) = \tau''}{(\forall t.\, \tau)[\overline{t'} \leftarrow \overline{\tau'}] = \forall t.\, \tau''}$$

α-Equality:

$\boxed{e_1 \equiv_\alpha e_2}$

$$\frac{x_1 = x_2}{x_1 \equiv_\alpha x_2} \quad \frac{e_1 \equiv_\alpha e_2 \quad e_1' \equiv_\alpha e_2'}{e_1\ e_1' \equiv_\alpha e_2\ e_2'} \quad \frac{\tau_1 \equiv_\alpha \tau_2 \quad x' \notin fv(e_1) \cup fv(e_2) \quad e_1[x_1 \leftarrow x'] \equiv_\alpha e_2[x_2 \leftarrow x']}{\lambda x_1 : \tau_1.\, e_1 \equiv_\alpha \lambda x_2 : \tau_2.\, e_2}$$

$$\frac{e_1 \equiv_\alpha e_2 \quad \tau_1 \equiv_\alpha \tau_2}{e_1\ \tau_1 \equiv_\alpha e_2\ \tau_2} \quad \frac{t' \notin tyfv(e_1) \cup tyfv(e_2) \quad e_1[t_1 \leftarrow t'] \equiv_\alpha e_2[t_2 \leftarrow t']}{\Lambda t_1.\, e_1 \equiv_\alpha \Lambda t_2.\, e_2}$$

$\boxed{\tau_1 \equiv_\alpha \tau_2}$

$$\frac{t_1 = t_2}{t_1 \equiv_\alpha t_2} \quad \frac{\tau_1 \equiv_\alpha \tau_2 \quad \tau_1' \equiv_\alpha \tau_2'}{\tau_1 \to \tau_1' \equiv_\alpha \tau_2 \to \tau_2'} \quad \frac{t' \notin tyfv(\tau_1) \cup tyfv(\tau_2) \quad \tau_1[t_1 \leftarrow t'] \equiv_\alpha \tau_2[t_2 \leftarrow t']}{\forall t_1.\, \tau_1 \equiv_\alpha \forall t_2.\, \tau_2}$$

定理 15 (Correctness of Substitution). 置換 $[\overline{x'} \leftarrow \overline{e'}]$ について，$X = \mathrm{dom}([\overline{x'} \leftarrow \overline{e'}])$ とした時，

$$fv(e[\overline{x'} \leftarrow \overline{e'}]) = (fv(e) \setminus X) \cup \bigcup_{x \in fv(e) \cap X} fv([\overline{x'} \leftarrow \overline{e'}](x)).$$

□

定理 16 (Correctness of Type Substitution). 式 e，型 τ，型置換 $[\overline{t'} \leftarrow \overline{\tau'}]$ について，$T = \mathrm{dom}([\overline{t'} \leftarrow \overline{\tau'}])$ とした時，

$$tyfv(e[\overline{t'} \leftarrow \overline{\tau'}]) = (tyfv(e) \setminus T) \cup \bigcup_{t \in tyfv(e) \cap T} tyfv([\overline{t'} \leftarrow \overline{\tau'}](t))$$

$$tyfv(\tau[\overline{t'} \leftarrow \overline{\tau'}]) = (tyfv(\tau) \setminus T) \cup \bigcup_{t \in tyfv(\tau) \cap T} tyfv([\overline{t'} \leftarrow \overline{\tau'}](t)).$$

□

定理 17 (α-Equality Does Not Touch Free Variables).

- $\tau_1 \equiv_\alpha \tau_2$ ならば $tyfv(\tau_1) = tyfv(\tau_2)$.
- $e_1 \equiv_\alpha e_2$ ならば，$fv(e_1) = fv(e_2)$，$tyfv(e_1) = tyfv(e_2)$.

□

2.5.2 Typing Semantics

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

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ \text{T-Var}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1 .\, e : \tau_1 \to \tau_2}\ \text{T-Abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau}\ \text{T-App}$$

$$\frac{\Gamma, t : \Omega \vdash e : \tau}{\Gamma \vdash \Lambda t .\, e : \forall t .\, \tau}\ \text{T-UnivAbs}$$

$$\frac{\Gamma \vdash e : \forall t .\, \tau_1}{\Gamma \vdash e\ \tau_2 : \tau_1[t \leftarrow \tau_2]}\ \text{T-UnivApp}$$

$$\frac{\Gamma \vdash \tau \equiv_\alpha \tau' : \Omega \quad \Gamma \vdash e : \tau'}{\Gamma \vdash e : \tau}\ \text{T-}\alpha\text{-Equiv}$$

特に，$\cdot \vdash e : \tau$ の時，$e : \tau$ と表記.

2.5.3 Evaluation Semantics (Call-By-Value)

$$\begin{array}{rcl} v & ::= & \lambda x : \tau .\, e \\ & | & \Lambda t .\, e \\ C & ::= & [] \\ & | & C\ e \\ & | & v\ C \\ & | & C\ \tau \end{array}$$

Small Step:

$\boxed{e \Rightarrow e'}$

$$\frac{}{(\lambda x : \tau .\, e)\ v \Rightarrow e[x \leftarrow v]}$$

$$\frac{}{(\Lambda t .\, e)\ \tau \Rightarrow e[t \leftarrow \tau]}$$

$$\frac{e \Rightarrow e'}{C[e] \Rightarrow C[e']}$$

Big Step:

$\boxed{e \Downarrow v}$

$$\frac{e_1 \Downarrow \lambda x : \tau .\, e_1' \quad e_2 \Downarrow v_2 \quad e_1'[x \leftarrow v_2] \Downarrow v}{e_1\ e_2 \Downarrow v}$$

$$\frac{e \Downarrow \Lambda t .\, e_1' \quad e_1'[t \leftarrow \tau] \Downarrow v}{e\ \tau \Downarrow v}$$

定理 18 (Adequacy of Small Step and Big Step). $e \Rightarrow^* v$ iff $e \Downarrow v$. □

定理 19 (Type Soundness). $e : \tau$ の時，$e \Rightarrow^* v$，$e \Downarrow v$ となる $v = \mathrm{nf}(\Rightarrow, e)$ が存在し，

- $\tau = \tau_1 \to \tau_2$ の時，$v \equiv_\alpha \lambda x' : \tau_1 . e'$ となる $\lambda x' : \tau_1 . e'$ が存在する．
- $\tau = \forall t. \tau_1$ の時，$v \equiv_\alpha \Lambda t. e'$ となる $\Lambda t. e'$ が存在する．

□

2.5.4 Equational Reasoning

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau}$

$$\frac{\Gamma, x : \tau_2 \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\lambda x : \tau_2 . e_1)\, e_2 \equiv e_1[x \leftarrow e_2] : \tau}\ \text{Eq-}\beta\text{-Lam} \qquad \frac{x \notin fv(e) \quad \Gamma \vdash e : \tau_1 \to \tau_2}{\Gamma \vdash (\lambda x : \tau_1 . e\ x) \equiv e : \tau_1 \to \tau_2}\ \text{Eq-}\eta\text{-Lam}$$

$$\frac{\Gamma, t : \Omega \vdash e : \tau}{\Gamma \vdash (\Lambda t. e)\, \tau_2 \equiv e[t \leftarrow \tau_2] : \tau[t \leftarrow \tau_2]}\ \text{Eq-}\beta\text{-UnivLam} \qquad \frac{t \notin tyfv(e) \quad \Gamma \vdash e : \forall t'. \tau}{\Gamma \vdash (\Lambda t. e\ t) \equiv e : \forall t'. \tau}\ \text{Eq-}\eta\text{-UnivLam}$$

$$\frac{e_1 \equiv_\alpha e_2 \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau}\ \text{Eq-}\alpha\text{-Refl} \qquad \frac{\tau \equiv_\alpha \tau' \quad \Gamma \vdash e_1 \equiv e_2 : \tau'}{\Gamma \vdash e_1 \equiv e_2 : \tau}\ \text{Eq-}\alpha\text{-Type}$$

$$\frac{\Gamma \vdash e_2 \equiv e_1 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau}\ \text{Eq-Sym} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau \quad \Gamma \vdash e_2 \equiv e_3 : \tau}{\Gamma \vdash e_1 \equiv e_3 : \tau}\ \text{Eq-Trans}$$

$$\frac{\Gamma, x : \tau \vdash e_1 \equiv e_2 : \tau'}{\Gamma \vdash \lambda x : \tau. e_1 \equiv \lambda x : \tau. e_2 : \tau \to \tau'}\ \text{Eq-Cong-Abs} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau' \to \tau \quad \Gamma \vdash e_1' \equiv e_2' : \tau'}{\Gamma \vdash e_1\, e_1' \equiv e_2\, e_2' : \tau}\ \text{Eq-Cong-App}$$

$$\frac{\Gamma, t : \Omega \vdash e_1 \equiv e_2 : \tau}{\Gamma \vdash \Lambda t. e_1 \equiv \Lambda t. e_2 : \forall (t.\tau)}\ \text{Eq-Cong-UnivAbs} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \forall t. \tau}{\Gamma \vdash e_1\, \tau' \equiv e_2\, \tau' : \tau[t \leftarrow \tau']}\ \text{Eq-Cong-UnivApp}$$

特に，$\cdot \vdash e_1 \equiv e_2 : \tau$ の時，$e_1 \equiv e_2 : \tau$ と表記．

定理 20 (Respect Typing). $\Gamma \vdash e_1 \equiv e_2 : \tau$ ならば，$\Gamma \vdash e_1 : \tau$ かつ $\Gamma \vdash e_2 : \tau$． □

定理 21 (Respect Evaluation). $e_1 \equiv e_2 : \tau$ の時，$e_1' \Rightarrow^* e_1$，$e_2 \Rightarrow^* e_2'$ ならば $e_1' \equiv e_2' : \tau$． □

系 22. $e_1 \equiv e_2 : \tau$ の時，$e_1 \Rightarrow^* e_1'$，$e_2 \Rightarrow^* e_2'$ ならば $e_1' \equiv e_2' : \tau$． □

証明. $e_1 \Rightarrow^* e_1$ より，定理 21 から $e_1 \equiv e_2' : \tau$．よって，T-Sym から $e_2' \equiv e_1 : \tau$ であり，$e_2' \Rightarrow^* e_2'$ より定理 21 から $e_2' \equiv e_1' : \tau$．故に，T-Sym から $e_1' \equiv e_2' : \tau$． ∎

2.5.5 Definability

Product

Product of τ_1 and τ_2:

$$\begin{aligned}
\tau_1 \times \tau_2 &\overset{\text{def}}{=} \forall t. (\tau_1 \to \tau_2 \to t) \to t \\
\langle e_1, e_2 \rangle &\overset{\text{def}}{=} \Lambda t. \lambda x : \tau_1 \to \tau_2 \to t. x\, e_1\, e_2 \\
\pi_1 e &\overset{\text{def}}{=} e\, \tau_1\, \lambda x_1. \lambda x_2. x_1 \\
\pi_2 e &\overset{\text{def}}{=} e\, \tau_2\, \lambda x_1. \lambda x_2. x_2
\end{aligned}$$

Admissible typing rule:

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

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2}\ \text{T-Product} \qquad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \pi_1 e : \tau_1}\ \text{T-Proj-1} \qquad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \pi_2 e : \tau_2}\ \text{T-Proj-2}$$

Admissible equality:

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau}$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \pi_1\langle e_1, e_2\rangle \equiv e_1 : \tau_1}\ \text{Eq-}\beta\text{-Product-1} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \pi_2\langle e_1, e_2\rangle \equiv e_2 : \tau_2}\ \text{Eq-}\beta\text{-Product-2}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \langle \pi_1 e, \pi_2 e\rangle \equiv e : \tau_1 \times \tau_2}\ \text{Eq-}\eta\text{-Product}$$

Existential Type

Existence of $\exists t.\, \tau$:

$$\begin{aligned} \exists t.\, \tau &\stackrel{\text{def}}{=} \forall t'.\, (\forall t.\, \tau \to t') \to t' \\ \text{pack}\langle \tau_t, e\rangle &\stackrel{\text{def}}{=} \Lambda t'.\, \lambda x : (\forall t.\, \tau \to t').\, x\ \tau_t\ e \\ \text{unpack}\langle t, x\rangle = e_1.\, \tau_2.\, e_2 &\stackrel{\text{def}}{=} e_1\ \tau_2\ (\Lambda t.\, \lambda x : \tau.\, e_2) \end{aligned}$$

Admissible typing rule:

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

$$\frac{\Gamma \vdash e : \tau[t \leftarrow \tau_t]}{\Gamma \vdash \text{pack}\langle \tau_t, e\rangle : \exists t.\, \tau}\ \text{T-Pack} \qquad \frac{\Gamma \vdash e_1 : \exists t.\, \tau \quad \Gamma, t : \Omega, x : \tau \vdash e_2 : \tau_2 \quad t \notin tyfv(\tau_2)}{\Gamma \vdash \text{unpack}\langle t, x\rangle = e_1.\, \tau_2.\, e_2 : \tau_2}\ \text{T-Unpack}$$

Admissible equality:

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau}$

$$\frac{\Gamma \vdash e_1 : \tau_1[t \leftarrow \tau_t] \quad \Gamma, t : \Omega, x : \tau_1 \vdash e_2 : \tau_2 \quad t \notin tyfv(\tau_2)}{\Gamma \vdash \text{unpack}\langle t, x\rangle = \text{pack}\langle \tau_t, e_1\rangle.\, \tau_2.\, e_2 \equiv e_2[t \leftarrow \tau_t][x \leftarrow e_1] : \tau_2}\ \text{Eq-}\beta\text{-Exist}$$

$$\frac{\Gamma \vdash e : \exists t'.\, \tau \quad \tau' \equiv_\alpha \exists t'.\, \tau}{\Gamma \vdash \text{unpack}\langle t, x\rangle = e.\, \tau'.\, \text{pack}\langle t, x\rangle \equiv e : \exists t'.\, \tau}\ \text{Eq-}\eta\text{-Exist}$$

2.5.6 Typability

[Wel99]

TODO

2.6 System-F ω

Alias: F ω, $\lambda\omega$ [RRD14]

2.6.1 Syntax

$$
\begin{array}{rcll}
e & ::= & x & \text{(variable)} \\
& | & \lambda x : \tau. e & \text{(abstraction)} \\
& | & e\ e & \text{(application)} \\
& | & \Lambda t : \kappa. e & \text{(universal abstraction)} \\
& | & e\ \tau & \text{(universal application)} \\
\tau & ::= & t & \text{(type variable)} \\
& | & \tau \to \tau & \text{(function type)} \\
& | & \forall t : \kappa. \tau & \text{(polymorphic type)} \\
& | & \lambda t : \kappa. \tau & \text{(type abstraction)} \\
& | & \tau\ \tau & \text{(type application)} \\
\kappa & ::= & \Omega & \text{(type kind)} \\
& | & \kappa \to \kappa & \text{(arrow kind)} \\
\Gamma & ::= & \cdot & \text{(empty)} \\
& | & \Gamma, x : \tau & \text{(variable cons)} \\
& | & \Gamma, t : \kappa & \text{(type variable cons)}
\end{array}
$$

Convention:

$$
\tau_1 \to \tau_2 \to \cdots \to \tau_n \stackrel{\text{def}}{=} \tau_1 \to (\tau_2 \to (\cdots \to \tau_n) \cdots)
$$
$$
e_1\ e_2\ \cdots\ e_n \stackrel{\text{def}}{=} (\cdots (e_1\ e_2) \cdots)\ e_n)
$$

Environment Reference:

$\boxed{\Gamma(x) = \tau}$

$$
\frac{x = x'}{(\Gamma, x' : \tau)(x) = \tau} \qquad \frac{x \neq x' \quad \Gamma(x) = \tau}{(\Gamma, x' : \tau')(x) = \tau} \qquad \frac{\Gamma(x) = \tau}{(\Gamma, t : \kappa)(x) = \tau}
$$
$$
\frac{t = t'}{(\Gamma, t' : \kappa)(t) = \kappa} \qquad \frac{t \neq t' \quad \Gamma(t) = \kappa}{(\Gamma, t' : \kappa')(t) = \kappa} \qquad \frac{\Gamma(t) = \kappa}{(\Gamma, x : \tau)(t) = \kappa}
$$

Free Variable:

$\boxed{fv(e) = \{\overline{x}\}}$

$$
\frac{}{fv(x) = \{x\}} \qquad \frac{fv(e) = X}{fv(\lambda x : \tau. e) = X \setminus \{x\}} \qquad \frac{fv(e_1) = X_1 \quad fv(e_2) = X_2}{fv(e_1\ e_2) = X_1 \cup X_2} \qquad \frac{fv(e) = X}{fv(\Lambda t : \kappa. e) = X} \qquad \frac{fv(e) = X}{fv(e\ \tau) = X}
$$

Substitution:

部分関数 $\{x_1 \mapsto e_1, \ldots, x_n \mapsto e_n\}$ を，$[x_1 \leftarrow e_1, \ldots, x_n \leftarrow e_n]$ または $[x_1, \ldots, x_n \leftarrow e_1, \ldots, e_n]$ と表記する.

$\boxed{e[\overline{x' \leftarrow e'}] = e''}$

$$
\frac{[\overline{x' \leftarrow e'}](x) = e}{x[\overline{x' \leftarrow e'}] = e} \qquad \frac{x \notin \mathrm{dom}([\overline{x' \leftarrow e'}])}{x[\overline{x' \leftarrow e'}] = x}
$$
$$
\frac{e([\overline{x' \leftarrow e'}] \restriction_{\mathrm{dom}([\overline{x' \leftarrow e'}]) \setminus \{x\}}) = e''}{(\lambda x : \tau. e)[\overline{x' \leftarrow e'}] = \lambda x : \tau. e''} \qquad \frac{e_1[\overline{x' \leftarrow e'}] = e_1'' \quad e_2[\overline{x' \leftarrow e'}] = e_2''}{(e_1\ e_2)[\overline{x' \leftarrow e'}] = e_1''\ e_2''}
$$

$$\frac{e[\overline{x'} \leftarrow \overline{e'}] = e''}{(\Lambda t : \kappa.\, e)[\overline{x'} \leftarrow \overline{e'}] = \Lambda t : \kappa.\, e''} \qquad \frac{e[\overline{x'} \leftarrow \overline{e'}] = e''}{(e\ \tau)[\overline{x'} \leftarrow \overline{e'}] = e''\ \tau}$$

Type Free Variable:

$\boxed{tyfv(e) = \{\overline{t}\}}$

$$\frac{}{tyfv(x) = \varnothing} \qquad \frac{tyfv(\tau) = T_1 \quad tyfv(e) = T_2}{tyfv(\lambda x : \tau.\, e) = T_1 \cup T_2} \qquad \frac{tyfv(e_1) = T_1 \quad tyfv(e_2) = T_2}{tyfv(e_1\ e_2) = T_1 \cup T_2}$$

$$\frac{tyfv(e) = T}{tyfv(\Lambda t : \kappa.\, e) = T \setminus \{t\}} \qquad \frac{tyfv(e) = T_1 \quad tyfv(\tau) = T_2}{tyfv(e\ \tau) = T_1 \cup T_2}$$

$\boxed{tyfv(\tau) = \{\overline{t}\}}$

$$\frac{}{tyfv(t) = \{t\}} \qquad \frac{tyfv(\tau_1) = T_1 \quad tyfv(\tau_2) = T_2}{tyfv(\tau_1 \to \tau_2) = T_1 \cup T_2} \qquad \frac{tyfv(\tau) = T}{tyfv(\forall t : \kappa.\, \tau) = T \setminus \{t\}}$$

$$\frac{tyfv(\tau) = T}{tyfv(\lambda t : \kappa.\, \tau) = T \setminus \{t\}} \qquad \frac{tyfv(\tau_1) = T_1 \quad tyfv(\tau_2) = T_2}{tyfv(\tau_1\ \tau_2) = T_1 \cup T_2}$$

Type Substitution:

部分関数 $\{t_1 \mapsto \tau_1, \dots, t_n \mapsto \tau_n\}$ を，$[t_1 \leftarrow \tau_1, \dots, t_n \leftarrow \tau_n]$ または $[t_1, \dots, t_n \leftarrow t_1, \dots, t_n]$ と表記する．

$\boxed{e[\overline{t'} \leftarrow \overline{\tau'}] = e'}$

$$\frac{}{x[\overline{t'} \leftarrow \overline{\tau'}] = x} \qquad \frac{e_1[\overline{t'} \leftarrow \overline{\tau'}] = e_1'' \quad e_2[\overline{t'} \leftarrow \overline{\tau'}] = e_2''}{(e_1\ e_2)[\overline{t'} \leftarrow \overline{\tau'}] = e_1''\ e_2''} \qquad \frac{\tau[\overline{t'} \leftarrow \overline{\tau'}] = \tau'' \quad e[\overline{t'} \leftarrow \overline{\tau'}] = e''}{(\lambda x : \tau.\, e)[\overline{t'} \leftarrow \overline{\tau'}] = \lambda x : \tau''.\, e''}$$

$$\frac{e[\overline{t'} \leftarrow \overline{\tau'}] = e'' \quad \tau[\overline{t'} \leftarrow \overline{\tau'}] = \tau''}{(e\ \tau)[\overline{t'} \leftarrow \overline{\tau'}] = e''\ \tau''} \qquad \frac{e([\overline{t'} \leftarrow \overline{\tau'}] \upharpoonright_{\mathrm{dom}([\overline{t'} \leftarrow \overline{\tau'}]) \setminus \{t\}}) = e''}{(\Lambda t : \kappa.\, e)[\overline{t'} \leftarrow \overline{\tau'}] = \Lambda t : \kappa.\, e''}$$

$\boxed{\tau[\overline{t'} \leftarrow \overline{\tau'}] = \tau''}$

$$\frac{[\overline{t'} \leftarrow \overline{\tau'}](t) = \tau}{t[\overline{t'} \leftarrow \overline{\tau'}] = \tau} \qquad \frac{t \notin \mathrm{dom}([\overline{t'} \leftarrow \overline{\tau'}])}{t[\overline{t'} \leftarrow \overline{\tau'}] = t}$$

$$\frac{\tau_1[\overline{t'} \leftarrow \overline{\tau'}] = \tau_1'' \quad \tau_2[\overline{t'} \leftarrow \overline{\tau'}] = \tau_2''}{(\tau_1 \to \tau_2)[\overline{t'} \leftarrow \overline{\tau'}] = \tau_1'' \to \tau_2''} \qquad \frac{\tau([\overline{t'} \leftarrow \overline{\tau'}] \upharpoonright_{\mathrm{dom}([\overline{t'} \leftarrow \overline{\tau'}]) \setminus \{t\}}) = \tau''}{(\forall t : \kappa.\, \tau)[\overline{t'} \leftarrow \overline{\tau'}] = \forall t : \kappa.\, \tau''}$$

$$\frac{\tau([\overline{t'} \leftarrow \overline{\tau'}] \upharpoonright_{\mathrm{dom}([\overline{t'} \leftarrow \overline{\tau'}]) \setminus \{t\}}) = \tau''}{(\lambda t : \kappa.\, \tau)[\overline{t'} \leftarrow \overline{\tau'}] = \lambda t : \kappa.\, \tau''} \qquad \frac{\tau_1[\overline{t'} \leftarrow \overline{\tau'}] = \tau_1'' \quad \tau_2[\overline{t'} \leftarrow \overline{\tau'}] = \tau_2''}{(\tau_1\ \tau_2)[\overline{t'} \leftarrow \overline{\tau'}] = \tau_1''\ \tau_2''}$$

α-Equality:

$\boxed{e_1 \equiv_\alpha e_2}$

$$\frac{x_1 = x_2}{x_1 \equiv_\alpha x_2} \qquad \frac{e_1 \equiv_\alpha e_2 \quad e_1' \equiv_\alpha e_2'}{e_1\ e_1' \equiv_\alpha e_2\ e_2'} \qquad \frac{\tau_1 \equiv_\alpha \tau_2 \quad x' \notin fv(e_1) \cup fv(e_2) \quad e_1[x_1 \leftarrow x'] \equiv_\alpha e_2[x_2 \leftarrow x']}{\lambda x_1 : \tau_1.\, e_1 \equiv_\alpha \lambda x_2 : \tau_2.\, e_2}$$

$$\frac{e_1 \equiv_\alpha e_2 \quad \tau_1 \equiv_\alpha \tau_2}{e_1\ \tau_1 \equiv_\alpha e_2\ \tau_2} \qquad \frac{t' \notin tyfv(e_1) \cup tyfv(e_2) \quad e_1[t_1 \leftarrow t'] \equiv_\alpha e_2[t_2 \leftarrow t']}{\Lambda t_1 : \kappa.\, e_1 \equiv_\alpha \Lambda t_2 : \kappa.\, e_2}$$

$\boxed{\tau_1 \equiv_\alpha \tau_2}$

$$\frac{t_1 = t_2}{t_1 \equiv_\alpha t_2} \qquad \frac{\tau_1 \equiv_\alpha \tau_2 \quad \tau_1' \equiv_\alpha \tau_2'}{\tau_1 \to \tau_1' \equiv_\alpha \tau_2 \to \tau_2'} \qquad \frac{t' \notin tyfv(\tau_1) \cup tyfv(\tau_2) \quad \tau_1[t_1 \leftarrow t'] \equiv_\alpha \tau_2[t_2 \leftarrow t']}{\forall t_1 : \kappa.\, \tau_1 \equiv_\alpha \forall t_2 : \kappa.\, \tau_2}$$

$$\frac{t' \notin tyfv(\tau_1) \cup tyfv(\tau_2) \quad \tau_1[t_1 \leftarrow t'] \equiv_\alpha \tau_2[t_2 \leftarrow t']}{\lambda t_1 : \kappa.\, \tau_1 \equiv_\alpha \lambda t_2 : \kappa.\, \tau_2} \qquad \frac{\tau_1 \equiv_\alpha \tau_2 \quad \tau_1' \equiv_\alpha \tau_2'}{\tau_1\ \tau_1' \equiv_\alpha \tau_2\ \tau_2'}$$

定理 23 (Correctness of Substitution). 置換 $[\overline{x'} \leftarrow \overline{e'}]$ について，$X = \mathrm{dom}([\overline{x'} \leftarrow \overline{e'}])$ とした時,

$$fv(e[\overline{x'} \leftarrow \overline{e'}]) = (fv(e) \setminus X) \cup \bigcup_{x \in fv(e) \cap X} fv([\overline{x'} \leftarrow \overline{e'}](x)).$$

□

定理 24 (Correctness of Type Substitution). 式 e，型 τ，型置換 $[\overline{t'} \leftarrow \overline{\tau'}]$ について，$T = \mathrm{dom}([\overline{t'} \leftarrow \overline{\tau'}])$ とした時,

$$tyfv(e[\overline{t'} \leftarrow \overline{\tau'}]) = (tyfv(e) \setminus T) \cup \bigcup_{t \in tyfv(e) \cap T} tyfv([\overline{t'} \leftarrow \overline{\tau'}](t))$$
$$tyfv(\tau[\overline{t'} \leftarrow \overline{\tau'}]) = (tyfv(\tau) \setminus T) \cup \bigcup_{t \in tyfv(\tau) \cap T} tyfv([\overline{t'} \leftarrow \overline{\tau'}](t)).$$

□

定理 25 (α-Equality Does Not Touch Free Variables).

- $\tau_1 \equiv_\alpha \tau_2$ ならば $tyfv(\tau_1) = tyfv(\tau_2)$.
- $e_1 \equiv_\alpha e_2$ ならば，$fv(e_1) = fv(e_2)$，$tyfv(e_1) = tyfv(e_2)$.

□

2.6.2 Typing Semantics

Kinding:

$\boxed{\Gamma \vdash \tau : \kappa}$

$$\frac{\Gamma(t) = \kappa}{\Gamma \vdash t : \kappa}\ \text{K-Var}$$

$$\frac{\Gamma \vdash \tau_1 : \Omega \quad \Gamma \vdash \tau_2 : \Omega}{\Gamma \vdash \tau_1 \to \tau_2 : \Omega}\ \text{K-Arrow}$$

$$\frac{\Gamma, t : \kappa \vdash \tau : \Omega}{\Gamma \vdash \forall t : \kappa.\, \tau : \Omega}\ \text{K-Forall}$$

$$\frac{\Gamma, t : \kappa_1 \vdash \tau : \kappa_2}{\Gamma \vdash \lambda t : \kappa_1.\, \tau : \kappa_1 \to \kappa_2}\ \text{K-Abs}$$

$$\frac{\Gamma \vdash \tau_1 : \kappa_2 \to \kappa \quad \Gamma \vdash \tau_2 : \kappa_2}{\Gamma \vdash \tau_1\ \tau_2 : \kappa}\ \text{K-App}$$

Type equivalence:

$\boxed{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}$

$$\frac{\Gamma, t : \kappa_2 \vdash \tau_1 : \kappa \quad \Gamma \vdash \tau_2 : \kappa_2}{\Gamma \vdash (\lambda t : \kappa_2.\, \tau_1)\ \tau_2 \equiv \tau_1[t \leftarrow \tau_2] : \kappa}\ \text{T-Eq-}\beta\text{-Lam} \qquad \frac{t \notin tyfv(\tau) \quad \Gamma \vdash \tau : \kappa_1 \to \kappa_2}{\Gamma \vdash (\lambda t : \kappa_1.\, \tau\ t) \equiv \tau : \kappa_1 \to \kappa_2}\ \text{T-Eq-}\eta\text{-Lam}$$

$$\frac{\tau_1 \equiv_\alpha \tau_2 \quad \Gamma \vdash \tau_1 : \kappa \quad \Gamma \vdash \tau_2 : \kappa}{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}\ \text{T-Eq-}\alpha\text{-Refl}$$

$$\frac{\Gamma \vdash \tau_2 \equiv \tau_1 : \kappa}{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}\ \text{T-Eq-Sym} \qquad \frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa \quad \Gamma \vdash \tau_2 \equiv \tau_3 : \kappa}{\Gamma \vdash \tau_1 \equiv \tau_3 : \kappa}\ \text{T-Eq-Trans}$$

$$\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \Omega \quad \Gamma \vdash \tau_1' \equiv \tau_2' : \Omega}{\Gamma \vdash \tau_1 \to \tau_1' \equiv \tau_2 \to \tau_2' : \Omega}\ \text{T-Eq-Cong-Arrow} \qquad \frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau_2 : \Omega}{\Gamma \vdash \forall t : \kappa.\, \tau_1 \equiv \forall t : \kappa.\, \tau_2 : \Omega}\ \text{Eq-Cong-Forall}$$

$$\frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau_2 : \kappa'}{\Gamma \vdash \lambda t : \kappa.\, \tau_1 \equiv \lambda t : \kappa.\, \tau_2 : \kappa \to \kappa'}\ \text{T-Eq-Cong-Abs} \qquad \frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa' \to \kappa \quad \Gamma \vdash \tau_1' \equiv \tau_2' : \kappa'}{\Gamma \vdash \tau_1\ \tau_1' \equiv \tau_2\ \tau_2' : \kappa}\ \text{Eq-Cong-App}$$

定理 26 (Respect Kinding). $\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa$ ならば，$\Gamma \vdash \tau_1 : \kappa$ かつ $\Gamma \vdash \tau_2 : \kappa$. □

Typing:

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

$$\frac{\Gamma \vdash \tau : \Omega \quad \Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ \text{T-Var}$$

$$\frac{\Gamma \vdash \tau_1 : \Omega \quad \Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1 . e : \tau_1 \to \tau_2}\ \text{T-Abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\, e_2 : \tau}\ \text{T-App}$$

$$\frac{\Gamma, t : \kappa \vdash e : \tau}{\Gamma \vdash \Lambda t : \kappa . e : \forall t : \kappa . \tau}\ \text{T-UnivAbs}$$

$$\frac{\Gamma \vdash e : \forall t : \kappa . \tau_1 \quad \Gamma \vdash \tau_2 : \kappa}{\Gamma \vdash e\, \tau_2 : \tau_1[t \leftarrow \tau_2]}\ \text{T-UnivApp}$$

$$\frac{\Gamma \vdash \tau \equiv \tau' : \Omega \quad \Gamma \vdash e : \tau'}{\Gamma \vdash e : \tau}\ \text{T-Equiv}$$

特に，$\cdot \vdash e : \tau$ の時，$e : \tau$ と表記.

定理 27 (Respect Type Kind). $\Gamma \vdash e : \tau$ ならば，$\Gamma \vdash \tau : \Omega$. □

2.6.3 Evaluation Semantics (Call-By-Value)

$$\begin{array}{rcl} v & ::= & \lambda x : \tau . e \\ & | & \Lambda t : \kappa . e \\ C & ::= & [] \\ & | & C\, e \\ & | & v\, C \\ & | & C\, \tau \end{array}$$

Small Step:

$\boxed{e \Rightarrow e'}$

$$\frac{}{(\lambda x : \tau . e)\, v \Rightarrow e[x \leftarrow v]}$$

$$\frac{}{(\Lambda t : \kappa . e)\, \tau \Rightarrow e[t \leftarrow \tau]}$$

$$\frac{e \Rightarrow e'}{C[e] \Rightarrow C[e']}$$

Big Step:

$\boxed{e \Downarrow v}$

$$\frac{e_1 \Downarrow \lambda x : \tau . e_1' \quad e_2 \Downarrow v_2 \quad e_1'[x \leftarrow v_2] \Downarrow v}{e_1\, e_2 \Downarrow v}$$

$$\frac{e \Downarrow \Lambda t : \kappa . e_1' \quad e_1'[t \leftarrow \tau] \Downarrow v}{e\, \tau \Downarrow v}$$

定理 28 (Adequacy of Small Step and Big Step). $e \Rightarrow^* v$ iff $e \Downarrow v$. □

定理 29 (Type Soundness). $e : \tau$ の時，$e \Rightarrow^* v$，$e \Downarrow v$ となる $v = \mathrm{nf}(\Rightarrow, e)$ が存在し，

- $\tau = \tau_1 \to \tau_2$ の時，$v \equiv_\alpha \lambda x' : \tau_1 . e'$ となる $\lambda x' : \tau_1 . e'$ が存在する.
- $\tau = \forall t : \kappa . \tau_1$ の時，$v \equiv_\alpha \Lambda t : \kappa . e'$ となる $\Lambda t : \kappa . e'$ が存在する.

□

2.6.4 Equational Reasoning

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau}$

$$\frac{\Gamma, x : \tau_2 \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\lambda x : \tau_2 .\, e_1)\, e_2 \equiv e_1[x \leftarrow e_2] : \tau}\ \text{Eq-}\beta\text{-Lam} \qquad \frac{x \notin fv(e) \quad \Gamma \vdash e : \tau_1 \to \tau_2}{\Gamma \vdash (\lambda x : \tau_1 .\, e\ x) \equiv e : \tau_1 \to \tau_2}\ \text{Eq-}\eta\text{-Lam}$$

$$\frac{\Gamma, t : \kappa \vdash e : \tau}{\Gamma \vdash (\Lambda t : \kappa .\, e)\ \tau_2 \equiv e[t \leftarrow \tau_2] : \tau[t \leftarrow \tau_2]}\ \text{Eq-}\beta\text{-UnivLam} \qquad \frac{t \notin tyfv(e) \quad \Gamma \vdash e : \forall t : \kappa .\, \tau}{\Gamma \vdash (\Lambda t : \kappa .\, e\ t) \equiv e : \forall t : \kappa .\, \tau}\ \text{Eq-}\eta\text{-UnivLam}$$

$$\frac{e_1 \equiv_\alpha e_2 \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau}\ \text{Eq-}\alpha\text{-Refl} \qquad \frac{\tau \equiv_\alpha \tau' \quad \Gamma \vdash e_1 \equiv e_2 : \tau'}{\Gamma \vdash e_1 \equiv e_2 : \tau}\ \text{Eq-}\alpha\text{-Type}$$

$$\frac{\Gamma \vdash e_2 \equiv e_1 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau}\ \text{Eq-Sym} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau \quad \Gamma \vdash e_2 \equiv e_3 : \tau}{\Gamma \vdash e_1 \equiv e_3 : \tau}\ \text{Eq-Trans}$$

$$\frac{\Gamma, x : \tau \vdash e_1 \equiv e_2 : \tau'}{\Gamma \vdash \lambda x : \tau .\, e_1 \equiv \lambda x : \tau .\, e_2 : \tau \to \tau'}\ \text{Eq-Cong-Abs} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau' \to \tau \quad \Gamma \vdash e_1' \equiv e_2' : \tau'}{\Gamma \vdash e_1\ e_1' \equiv e_2\ e_2' : \tau}\ \text{Eq-Cong-App}$$

$$\frac{\Gamma, t : \kappa \vdash e_1 \equiv e_2 : \tau}{\Gamma \vdash \Lambda t : \kappa .\, e_1 \equiv \Lambda t : \kappa .\, e_2 : (\forall t : \kappa .\, \tau)}\ \text{Eq-Cong-UnivAbs}$$

$$\frac{\Gamma \vdash e_1 \equiv e_2 : \forall t : \kappa .\, \tau \quad \Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}{\Gamma \vdash e_1\ \tau_1 \equiv e_2\ \tau_2 : \tau[t \leftarrow \tau_1]}\ \text{Eq-Cong-UnivApp}$$

特に，$\cdot \vdash e_1 \equiv e_2 : \tau$ の時，$e_1 \equiv e_2 : \tau$ と表記.

定理 30 (Respect Typing). $\Gamma \vdash e_1 \equiv e_2 : \tau$ ならば，$\Gamma \vdash e_1 : \tau$ かつ $\Gamma \vdash e_2 : \tau$. □

定理 31 (Respect Evaluation). $e_1 \equiv e_2 : \tau$ の時，$e_1' \Rightarrow^* e_1$，$e_2 \Rightarrow^* e_2'$ ならば $e_1' \equiv e_2' : \tau$. □

系 32. $e_1 \equiv e_2 : \tau$ の時，$e_1 \Rightarrow^* e_1'$，$e_2 \Rightarrow^* e_2'$ ならば $e_1' \equiv e_2' : \tau$. □

証明. $e_1 \Rightarrow^* e_1$ より，定理 21 から $e_1 \equiv e_2' : \tau$. よって，T-Sym から $e_2' \equiv e_1 : \tau$ であり，$e_2' \Rightarrow^* e_2'$ より定理 21 から $e_2' \equiv e_1' : \tau$. 故に，T-Sym から $e_1' \equiv e_2' : \tau$. ∎

2.6.5 Definability

Product

Product of τ_1 and τ_2:

$$\begin{aligned}
\tau_1 \times \tau_2 &\stackrel{\text{def}}{=} \forall t : \Omega .\, (\tau_1 \to \tau_2 \to t) \to t \\
\langle e_1, e_2 \rangle &\stackrel{\text{def}}{=} \Lambda t : \Omega .\, \lambda x : \tau_1 \to \tau_2 \to t .\, x\ e_1\ e_2 \\
\pi_1 e &\stackrel{\text{def}}{=} e\ \tau_1\ \lambda x_1 .\, \lambda x_2 .\, x_1 \\
\pi_2 e &\stackrel{\text{def}}{=} e\ \tau_2\ \lambda x_1 .\, \lambda x_2 .\, x_2
\end{aligned}$$

Admissible kinding:

$\boxed{\Gamma \vdash \tau : \kappa}$

$$\frac{\Gamma \vdash \tau_1 : \Omega \quad \Gamma \vdash \tau_2 : \Omega}{\Gamma \vdash \tau_1 \times \tau_2 : \Omega}\ \text{T-Product}$$

Admissible type equality:

$\boxed{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}$

$$\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \Omega \quad \Gamma \vdash \tau_1' \equiv \tau_2' : \Omega}{\Gamma \vdash \tau_1 \times \tau_1' \equiv \tau_2 \times \tau_2' : \Omega} \text{ T-Eq-Product}$$

Admissible typing:

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

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2} \text{ T-Product} \qquad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \pi_1 e : \tau_1} \text{ T-Proj-1} \qquad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \pi_2 e : \tau_2} \text{ T-Proj-2}$$

Admissible equality:

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau}$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \pi_1 \langle e_1, e_2 \rangle \equiv e_1 : \tau_1} \text{ Eq-}\beta\text{-Product-1} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \pi_2 \langle e_1, e_2 \rangle \equiv e_2 : \tau_2} \text{ Eq-}\beta\text{-Product-2}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \langle \pi_1 e, \pi_2 e \rangle \equiv e : \tau_1 \times \tau_2} \text{ Eq-}\eta\text{-Product}$$

Existential Type

Existence of $\exists t : \kappa. \tau$:

$$\begin{aligned} \exists t : \kappa. \tau &\stackrel{\text{def}}{=} \forall t' : \Omega. (\forall t : \kappa. \tau \to t') \to t' \\ \text{pack}\langle \tau_t, e \rangle_{\exists t : \kappa. \tau} &\stackrel{\text{def}}{=} \Lambda t' : \Omega. \lambda x : (\forall t : \kappa. \tau \to t'). x\ \tau_t\ e \\ \text{unpack}\langle t : \kappa, x : \tau \rangle = e_1. \tau_2. e_2 &\stackrel{\text{def}}{=} e_1\ \tau_2\ (\Lambda t : \kappa. \lambda x : \tau. e_2) \end{aligned}$$

Admissible kinding:

$\boxed{\Gamma \vdash \tau : \kappa}$

$$\frac{\Gamma, t : \kappa \vdash \tau : \Omega}{\Gamma \vdash \exists t : \kappa. \tau : \Omega} \text{ T-Exist}$$

Admissible type equality:

$\boxed{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}$

$$\frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau_2 : \Omega}{\Gamma \vdash \exists t : \kappa. \tau_1 \equiv \exists t : \kappa. \tau_2 : \Omega} \text{ T-Eq-Cong-Exist}$$

Admissible typing rule:

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

$$\frac{\Gamma, t : \kappa \vdash \tau : \Omega \quad \Gamma \vdash \tau_t : \kappa \quad \Gamma \vdash e : \tau[t \leftarrow \tau_t]}{\Gamma \vdash \text{pack}\langle \tau_t, e \rangle_{\exists t : \kappa. \tau} : \exists t : \kappa. \tau} \text{ T-Pack}$$

$$\frac{\Gamma \vdash e_1 : \exists t : \kappa. \tau \quad \Gamma, t : \kappa, x : \tau \vdash e_2 : \tau_2 \quad t \notin tyfv(\tau_2)}{\Gamma \vdash \text{unpack}\langle t : \kappa, x : \tau \rangle = e_1. \tau_2. e_2 : \tau_2} \text{ T-Unpack}$$

Admissible equality:

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau}$

$$\frac{\Gamma \vdash \tau_t : \kappa \quad \Gamma \vdash e_1 : \tau_1[t \leftarrow \tau_t] \quad \Gamma, t : \kappa, x : \tau_1 \vdash e_2 : \tau_2 \quad t \notin tyfv(\tau_2)}{\Gamma \vdash \text{unpack}\langle t : \kappa, x : \tau_1 \rangle = \text{pack}\langle \tau_t, e_1 \rangle_{\exists t : \kappa . \tau_1} . \tau_2 . e_2 \equiv e_2[t \leftarrow \tau_t][x \leftarrow e_1] : \tau_2} \text{ Eq-}\beta\text{-Exist}$$

$$\frac{\Gamma \vdash e : (\exists t : \kappa . \tau) \quad \tau' \equiv \exists t : \kappa . \tau}{\Gamma \vdash \text{unpack}\langle t : \kappa, x : \tau \rangle = e . \tau' . \text{pack}\langle t, x \rangle_{\exists t : \kappa . \tau} \equiv e : (\exists t : \kappa . \tau)} \text{ Eq-}\eta\text{-Exist}$$

2.7 λ μ -Calculus

Alias: λ μ [Sel01][Roc05]

2.7.1 Syntax

$$
\begin{array}{lcll}
\tau & ::= & t & \text{(type variable)} \\
 & | & \top & \text{(top type)} \\
 & | & \tau \times \tau & \text{(product type)} \\
 & | & \tau \to \tau & \text{(function type)} \\
 & | & \bot & \text{(bottom type)} \\
e & ::= & x & \text{(variable)} \\
 & | & \langle\rangle & \text{(top value)} \\
 & | & \langle e, e\rangle & \text{(product)} \\
 & | & \pi_1 e & \text{(left projection)} \\
 & | & \pi_2 e & \text{(right projection)} \\
 & | & \lambda x : \tau . e & \text{(abstraction)} \\
 & | & e\ e & \text{(application)} \\
 & | & [\alpha]e & \text{(naming)} \\
 & | & \mu\alpha : \tau . e & \text{(un-naming)} \\
\Gamma & ::= & \cdot & \\
 & | & \Gamma, x : \tau & \\
\Delta & ::= & \cdot & \\
 & | & \alpha : \tau, \Delta &
\end{array}
$$

Environment Reference:

$\boxed{\Gamma(x) = \tau}$

$$\frac{x = x'}{(\Gamma, x' : \tau)(x) = \tau} \qquad \frac{x \neq x' \quad \Gamma(x) = \tau}{(\Gamma, x' : \tau')(x) = \tau}$$

$\boxed{\Delta(\alpha) = \tau}$

$$\frac{\alpha = \alpha'}{(\alpha' : \tau, \Delta)(\alpha) = \tau} \qquad \frac{\alpha \neq \alpha' \quad \Delta(\alpha) = \tau}{(\alpha' : \tau', \Delta)(\alpha) = \tau}$$

2.7.2 Typing Semantics

$\boxed{\Gamma \vdash e : \tau \mid \Delta}$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau \mid \Delta}\ \text{T-Var}$$

$$\frac{}{\Gamma \vdash \langle\rangle : \top \mid \Delta}\ \text{T-Top}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \mid \Delta \quad \Gamma \vdash e_2 : \tau_2 \mid \Delta}{\Gamma \vdash \langle e_1, e_2\rangle : \tau_1 \times \tau_2 \mid \Delta}\ \text{T-Product}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \mid \Delta}{\Gamma \vdash \pi_1 e : \tau_1 \mid \Delta}\ \text{T-Proj-1}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \mid \Delta}{\Gamma \vdash \pi_2 e : \tau_2 \mid \Delta}\ \text{T-Proj-2}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2 \mid \Delta}{\Gamma \vdash \lambda x : \tau_1 . e : \tau_1 \to \tau_2 \mid \Delta}\ \text{T-Abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau \mid \Delta \quad \Gamma \vdash e_2 : \tau_2 \mid \Delta}{\Gamma \vdash e_1\, e_2 : \tau \mid \Delta}\ \text{T-App}$$

$$\frac{\Delta(\alpha) = \tau \quad \Gamma \vdash e : \tau \mid \Delta}{\Gamma \vdash [\alpha]e : \bot \mid \Delta}\ \text{T-Name}$$

$$\frac{\Gamma \vdash e : \bot \mid \alpha : \tau, \Delta}{\Gamma \vdash (\mu\alpha : \tau.\, e) : \tau \mid \Delta}\ \text{T-Unname}$$

2.7.3 Equivalence

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau \mid \Delta}$

$$\frac{\Gamma, x : \tau_2 \vdash e_1 : \tau \mid \Delta \quad \Gamma \vdash e_2 : \tau_2 \mid \Delta}{\Gamma \vdash (\lambda x : \tau_2.\, e_1)\, e_2 \equiv e_1[x \leftarrow e_2] : \tau \mid \Delta}\ \text{Eq-}\beta\text{-Lam}$$

$$\frac{x \notin fv(e) \quad \Gamma \vdash e : \tau_1 \to \tau_2 \mid \Delta}{\Gamma \vdash (\lambda x : \tau_1.\, e\, x) \equiv e : \tau_1 \to \tau_2 \mid \Delta}\ \text{Eq-}\eta\text{-Lam}$$

$$\frac{\Gamma \vdash e : \top \mid \Delta}{\Gamma \vdash \langle\rangle \equiv e : \top \mid \Delta}\ \text{Eq-}\eta\text{-Top}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \mid \Delta \quad \Gamma \vdash e_2 : \tau_2 \mid \Delta}{\Gamma \vdash \pi_1\langle e_1, e_2\rangle \equiv e_1 : \tau_1 \mid \Delta}\ \text{Eq-}\beta\text{-Product-1}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \mid \Delta \quad \Gamma \vdash e_2 : \tau_2 \mid \Delta}{\Gamma \vdash \pi_2\langle e_1, e_2\rangle \equiv e_2 : \tau_2 \mid \Delta}\ \text{Eq-}\beta\text{-Product-2}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \mid \Delta}{\Gamma \vdash \langle \pi_1 e, \pi_2 e\rangle \equiv e : \tau_1 \times \tau_2 \mid \Delta}\ \text{Eq-}\eta\text{-Product}$$

$$\frac{\alpha_1 \notin fv(e) \quad \Gamma \vdash e : \bot \mid \alpha : \tau_1 \times \tau_2, \Delta}{\Gamma \vdash \pi_1(\mu\alpha : \tau_1 \times \tau_2.\, e) \equiv \mu\alpha_1 : \tau_1.\, e[[\alpha](-) \leftarrow [\alpha_1](\pi_1(-))] : \tau_1 \mid \Delta}\ \text{Eq-}\zeta\text{-Product-1}$$

$$\frac{\alpha_2 \notin fv(e) \quad \Gamma \vdash e : \bot \mid \alpha : \tau_1 \times \tau_2, \Delta}{\Gamma \vdash \pi_2(\mu\alpha : \tau_1 \times \tau_2.\, e) \equiv \mu\alpha_2 : \tau_2.\, e[[\alpha](-) \leftarrow [\alpha_2](\pi_2(-))] : \tau_2 \mid \Delta}\ \text{Eq-}\zeta\text{-Product-2}$$

$$\frac{\Gamma \vdash e : \bot \mid \alpha_2 : \tau_\alpha, \Delta}{\Gamma \vdash [\alpha_1](\mu\alpha_2 : \tau_\alpha.\, e) \equiv e[\alpha_2 \leftarrow \alpha_1] : \bot \mid \Delta}\ \text{Eq-}\beta\text{-Mu}$$

$$\frac{\Gamma \vdash e : \tau \mid \Delta}{\Gamma \vdash (\mu\alpha : \tau.\, [\alpha]e) \equiv e : \tau \mid \Delta}\ \text{Eq-}\eta\text{-Mu}$$

$$\frac{\alpha_2 \notin fv(e_1) \cup fv(e_2) \quad \Gamma \vdash e_1 : \bot \mid \alpha : \tau_1 \to \tau_2, \Delta \quad \Gamma \vdash e_2 : \tau_2 \mid \Delta}{\Gamma \vdash (\mu\alpha : \tau_1 \to \tau_2.\, e_1)\, e_2 \equiv \mu\alpha_2 : \tau_2.\, e_1[[\alpha](-) \leftarrow [\alpha_2]((-)\, e_2)] : \tau_2 \mid \Delta}\ \text{Eq-}\zeta\text{-Mu}$$

2.7.4 Elaboration (Call-By-Value)

$\boxed{\Gamma \vdash e : \tau \rightsquigarrow e'}$

$$\frac{\Gamma(x_{x_0}) = V_\tau}{\Gamma \vdash x_0 : \tau \rightsquigarrow \lambda x_k : K_\tau.\, x_k\, x_{x_0}}$$

$$\frac{}{\Gamma \vdash \langle\rangle : \top \rightsquigarrow \lambda x_k : K_\top.\, x_k\, \langle\rangle}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightsquigarrow e_1' \quad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow e_2'}{\Gamma \vdash \langle e_1, e_2\rangle : \tau_1 \times \tau_2 \rightsquigarrow \lambda x_k : K_{\tau_1 \times \tau_2}.\, e_1'\, (\lambda x_1 : V_{\tau_1}.\, e_2'\, (\lambda x_2 : V_{\tau_2}.\, x_k\, \langle x_1, x_2\rangle))}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \rightsquigarrow e'}{\Gamma \vdash \pi_1 e : \tau_1 \rightsquigarrow \lambda x_k : K_{\tau_1}.\, e'\, (\lambda x : V_{\tau_1} \times V_{\tau_2}.\, x_k\, (\pi_1 x))}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \rightsquigarrow e'}{\Gamma \vdash \pi_2 e : \tau_2 \rightsquigarrow \lambda x_k : K_{\tau_2}.\, e'\, (\lambda x : V_{\tau_1} \times V_{\tau_2}.\, x_k\, (\pi_2 x))}$$

$$\frac{\Gamma, x_{x_0} : V_{\tau_1} \vdash e : \tau_2 \rightsquigarrow e'}{\Gamma \vdash (\lambda x_0 : \tau_1.\, e) : \tau_1 \to \tau_2 \rightsquigarrow \lambda x_k : K_{\tau_1 \to \tau_2}.\, x_k\, (\lambda x : V_{\tau_1} \times K_{\tau_2}.\, (\lambda x_{x_0} : V_{\tau_1}.\, e')\, (\pi_1 x)\, (\pi_2 x))}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau \rightsquigarrow e_1' \quad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow e_2'}{\Gamma \vdash e_1\ e_2 : \tau \rightsquigarrow \lambda x_k : K_\tau.\, e_1'\ (\lambda x_1 : V_{\tau_2 \to \tau}.\, e_2'\ (\lambda x_2 : V_{\tau_2}.\, x_1\ \langle x_2, x_k \rangle))}$$

$$\frac{\Gamma, x_\alpha : K_\tau \vdash e : \bot \rightsquigarrow e'}{\Gamma \vdash (\mu\alpha : \tau.\, e) : \tau \rightsquigarrow \lambda x_\alpha : K_\tau.\, e'\ (\lambda x : \bot.\, \mathrm{case}\ x\ \{\})}$$

$$\frac{\Gamma(x_\alpha) = K_\tau \quad \Gamma \vdash e : \tau \rightsquigarrow e'}{\Gamma \vdash [\alpha]e : \tau \rightsquigarrow \lambda x_k : K_\bot.\, e'\ x_\alpha}$$

$\boxed{V_\tau = \tau'}$

$$\frac{}{V_\top = \top}$$

$$\frac{V_{\tau_1} = \tau_1' \quad V_{\tau_2} = \tau_2'}{V_{\tau_1 \times \tau_2} = V_{\tau_1'} \times V_{\tau_2'}}$$

$$\frac{V_{\tau_1} = \tau_1' \quad K_{\tau_2} = \tau_2'}{V_{\tau_1 \to \tau_2} = \tau_1' \times \tau_2' \to R}$$

$$\frac{}{V_\bot = \bot}$$

Abbreviation:

$$K_\tau \overset{\mathrm{def}}{=} V_\tau \to R$$
$$C_\tau \overset{\mathrm{def}}{=} K_\tau \to R$$

定理 33. $\Gamma \vdash e : \tau \rightsquigarrow e'$ ならば，$\Gamma \vdash e' : C_\tau$. □

定理 34. $\Gamma \vdash e : \tau \mid \Delta \iff V(\Gamma), K(\Delta) \vdash e : \tau \rightsquigarrow e'$．ただし，

$$V(\Gamma) \overset{\mathrm{def}}{=} \begin{cases} V(\Gamma'), x_{x'} : V_{\tau'} & (\Gamma = \Gamma', x' : \tau') \\ \cdot & (\Gamma = \cdot) \end{cases}$$
$$K(\Delta) \overset{\mathrm{def}}{=} \begin{cases} x_\alpha : K_\tau, K(\Delta') & (\Delta = \alpha : \tau, \Delta') \\ \cdot & (\Delta = \cdot) \end{cases} .$$

□

2.7.5 Elaboration (Call-By-Name)

$\boxed{\Gamma \vdash e : \tau \rightsquigarrow e'}$

$$\frac{\Gamma(x_{x_0}) = C_\tau}{\Gamma \vdash x_0 : \tau \rightsquigarrow \lambda x_k : K_\tau.\, x_{x_0}\ x_k}$$

$$\frac{}{\Gamma \vdash \langle\rangle : \top \rightsquigarrow \lambda x_k : \bot.\, \mathrm{case}\ x_k\ \{\}}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightsquigarrow e_1' \quad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow e_2'}{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2 \rightsquigarrow \lambda x_k : K_{\tau_1} + K_{\tau_2}.\, \mathrm{case}\ x_k\ \{x_{k_1}.\, e_1'\ x_{k_1} \mid x_{k_2}.\, e_2'\ x_{k_2}\}}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \rightsquigarrow e'}{\Gamma \vdash \pi_1 e : \tau_1 \rightsquigarrow \lambda x_k : K_{\tau_1}.\, e'\ (i_1 x_k)}$$

$$\frac{\Gamma, x_{x_1} : C_{\tau_1} \vdash e : \tau_2 \rightsquigarrow e'}{\Gamma \vdash (\lambda x_1 : \tau_1.\, e) : \tau_1 \to \tau_2 \rightsquigarrow \lambda x_k : C_{\tau_1} \times K_{\tau_2}.\, e'[x_{x_1} \leftarrow \pi_1 x_k]\ (\pi_2 x_k)}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau \rightsquigarrow e_1' \quad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow e_2'}{\Gamma \vdash e_1\ e_2 : \tau \rightsquigarrow \lambda x_k : K_\tau.\, e_1'\ \langle e_2', x_k \rangle}$$

$$\frac{\Gamma(x_\alpha) = K_\tau \quad \Gamma \vdash e : \tau \rightsquigarrow e'}{\Gamma \vdash [\alpha]e : \bot \rightsquigarrow \lambda x_k : K_\bot.\, e'\ x_\alpha}$$

$$\frac{\Gamma, x_\alpha : K_\tau \vdash e : \bot \rightsquigarrow e'}{\Gamma \vdash (\mu\alpha : \tau.\, e) : \tau \rightsquigarrow \lambda x_\alpha : K_\tau.\, e'\ \langle\rangle}$$

$\boxed{K_\tau = \tau'}$

$$\frac{}{K_\top = \bot}$$

$$\frac{K_{\tau_1} = \tau_1' \quad K_{\tau_2} = \tau_2'}{K_{\tau_1 \times \tau_2} = \tau_1' + \tau_2'}$$

$$\frac{C_{\tau_1} = \tau_1' \quad K_{\tau_2} = \tau_2'}{K_{\tau_1 \to \tau_2} = \tau_1' \times \tau_2'}$$

$$\frac{}{K_\bot = \top}$$

Abbreviation:

$$C_\tau \stackrel{\text{def}}{=} K_\tau \to R$$

定理 35. $\Gamma \vdash e : \tau \rightsquigarrow e'$ ならば，$\Gamma \vdash e' : C_\tau$. □

定理 36. $\Gamma \vdash e : \tau \mid \Delta \iff C(\Gamma), K(\Delta) \vdash e : \tau \rightsquigarrow e'$．ただし，

$$C(\Gamma) \stackrel{\text{def}}{=} \begin{cases} C(\Gamma'), x_{x'} : C_{\tau'} & (\Gamma = \Gamma', x' : \tau') \\ \cdot & (\Gamma = \cdot) \end{cases}$$
$$K(\Delta) \stackrel{\text{def}}{=} \begin{cases} x_\alpha : K_\tau, K(\Delta') & (\Delta = \alpha : \tau, \Delta') \\ \cdot & (\Delta = \cdot) \end{cases}.$$

□

2.8 WIP: Lambda Bar Mu Mu Tilde Calculus

$\bar{\lambda}\mu\tilde{\mu}$-Calculus

2.9 WIP: π-Calculus

第 3 章

Basic Algorithms

3.1 Martelli-Montanari Algorithm

[MM82]

$$\frac{}{\mathcal{U}(x, x) = \varnothing}$$

$$\frac{x_1 \neq x_2}{\mathcal{U}(x_1, x_2) = \{x_1 \mapsto x_2\}}$$

$$\frac{}{\mathcal{U}(f(a_1, \dots, a_n), f(b_1, \dots, b_n)) = \bigcup_{1 \leq i \leq n} \mathcal{U}(a_i, b_i)}$$

$$\frac{x \notin \mathrm{fv}(f(a_1, \dots, a_n))}{\mathcal{U}(x, f(a_1, \dots, a_n)) = \{x \mapsto f(a_1, \dots, a_n)\}}$$

$$\frac{x \notin \mathrm{fv}(f(a_1, \dots, a_n))}{\mathcal{U}(f(a_1, \dots, a_n), x) = \{x \mapsto f(a_1, \dots, a_n)\}}$$

第 4 章

Modules and Phase Distinction

4.1 Light-Weight F-ing modules

[RRD14]

4.1.1 Internal Language

Having same power as System F ω

Syntax:

$$
\begin{array}{lcl}
\kappa & ::= & \Omega \mid \kappa \to \kappa \\
\tau & ::= & t \mid \tau \to \tau \mid \{\overline{l : \tau}\} \mid \forall t : \kappa.\, \tau \mid \exists t : \kappa.\, \tau \mid \lambda t : \kappa.\, \tau \mid \tau\ \tau \\
e & ::= & x \mid \lambda x : \tau.\, e \mid e\ e \mid \{\overline{l = e}\} \mid e.l \mid \Lambda t : \kappa.\, e \mid e\ \tau \mid \mathrm{pack}\langle \tau, e\rangle_\tau \mid \mathrm{unpack}\langle t : \kappa, x : \tau\rangle = e \text{ in } e \\
\Gamma & ::= & \cdot \mid \Gamma, t : \kappa \mid \Gamma, x : \tau
\end{array}
$$

Abbreviation:

$$
\Sigma.\bar{l} \stackrel{\text{def}}{=} \begin{cases} (\Sigma.l).\overline{l'} & (\bar{l} = l\ \overline{l'}) \\ \Sigma & (\bar{l} = \epsilon) \end{cases}
$$

$$
\overline{\tau_1} \to \tau_2 \stackrel{\text{def}}{=} \begin{cases} \tau_1 \to (\overline{\tau_1'} \to \tau_2) & (\overline{\tau_1} = \tau_1\ \overline{\tau_1'}) \\ \tau_2 & (\overline{\tau_1} = \epsilon) \end{cases}
$$

$$
\lambda \overline{x : \tau}.\, e \stackrel{\text{def}}{=} \begin{cases} \lambda x : \tau.\, \lambda \overline{x' : \tau'}.\, e & (\overline{x : \tau} = x : \tau\ \overline{x' : \tau'}) \\ e & (\overline{x : \tau} = \epsilon) \end{cases}
$$

$$
e_0\ \overline{e_1} \stackrel{\text{def}}{=} \begin{cases} e_0\ e_1\ \overline{e_1'} & (\overline{e_1} = e_1\ \overline{e_1'}) \\ e_0 & (\overline{e_1} = \epsilon) \end{cases}
$$

$$
\forall \overline{t : \kappa}.\, \tau \stackrel{\text{def}}{=} \begin{cases} \forall t : \kappa.\, \forall \overline{t' : \kappa'}.\, \tau & (\overline{t : \kappa} = t : \kappa\ \overline{t' : \kappa'}) \\ \tau & (\overline{t : \kappa} = \epsilon) \end{cases}
$$

$$
\Lambda \overline{t : \kappa}.\, e \stackrel{\text{def}}{=} \begin{cases} \Lambda t : \kappa.\, \Lambda \overline{t' : \kappa'}.\, e & (\overline{t : \kappa} = t : \kappa\ \overline{t' : \kappa'}) \\ e & (\overline{t : \kappa} = \epsilon) \end{cases}
$$

$$
e\ \overline{\tau} \stackrel{\text{def}}{=} \begin{cases} e\ \tau\ \overline{\tau'} & (\overline{\tau} = \tau\ \overline{\tau'}) \\ e & (\overline{\tau} = \epsilon) \end{cases}
$$

$$
\mathrm{let}\, \overline{x : \tau = e_1}\ \overline{t : \kappa = \tau} \text{ in } e_2 \stackrel{\text{def}}{=} (\lambda \overline{x : \tau}.\, \Lambda \overline{t : \kappa}.\, e_2)\ \overline{e_1}\ \overline{\tau}
$$

$$
\exists \overline{t : \kappa}.\, \tau \stackrel{\text{def}}{=} \begin{cases} \exists t : \kappa.\, \exists \overline{t' : \kappa'}.\, \tau & (\overline{t : \kappa} = t : \kappa\ \overline{t' : \kappa'}) \\ \tau & (\overline{t : \kappa} = \epsilon) \end{cases}
$$

$$
\mathrm{pack}\langle \overline{\tau}, e\rangle_{\exists \overline{t : \kappa}.\tau_0} \stackrel{\text{def}}{=} \begin{cases} \mathrm{pack}\langle \tau, \mathrm{pack}\langle \overline{\tau'}, e\rangle_{\exists \overline{t' : \kappa'}.\tau_0}\rangle_{\exists \overline{t : \kappa}.\tau_0} & (\overline{\tau} = \tau\ \overline{\tau'}, \overline{t : \kappa} = t : \kappa\ \overline{t' : \kappa'}) \\ e & (\overline{\tau} = \epsilon, \overline{t : \kappa} = \epsilon) \end{cases}
$$

$$
(\mathrm{unpack}\langle \overline{t : \kappa}, x : \tau\rangle = e_1 \text{ in } e_2) \stackrel{\text{def}}{=} \begin{cases} \begin{array}{l} \mathrm{unpack}\langle t : \kappa, x_1 : \exists \overline{t' : \kappa'}.\, \tau\rangle = e_1 \text{ in} \\ \quad \mathrm{unpack}\langle \overline{t' : \kappa'}, x_2 : \tau\rangle = x_1 \text{ in } e_2 \end{array} & (\overline{t : \kappa} = t : \kappa\ \overline{t' : \kappa'}) \\ \mathrm{let}\, x : \tau = e_1 \text{ in } e_2 & (\overline{t : \kappa} = \epsilon) \end{cases}
$$

Kinding:

$\boxed{\Gamma \vdash \tau : \kappa}$

$$
\frac{\Gamma(t) = \kappa}{\Gamma \vdash t : \kappa} \qquad \frac{\Gamma \vdash \tau_1 : \Omega \quad \Gamma \vdash \tau_2 : \Omega}{\Gamma \vdash \tau_1 \to \tau_2 : \Omega} \qquad \frac{\bigwedge_l \Gamma \vdash \tau_l : \Omega}{\Gamma \vdash \{\overline{l : \tau_l}\} : \Omega}
$$

$$
\frac{\Gamma, t : \kappa \vdash \tau : \Omega}{\Gamma \vdash \forall t : \kappa.\, \tau : \Omega} \qquad \frac{\Gamma, t : \kappa \vdash \tau : \Omega}{\Gamma \vdash \exists t : \kappa.\, \tau : \Omega} \qquad \frac{\Gamma, t : \kappa_1 \vdash \tau : \kappa_2}{\Gamma \vdash \lambda t : \kappa_1.\, \tau : \kappa_1 \to \kappa_2} \qquad \frac{\Gamma \vdash \tau_1 : \kappa_2 \to \kappa \quad \Gamma \vdash \tau_2 : \kappa_2}{\Gamma \vdash \tau_1\ \tau_2 : \kappa}
$$

Type equivalence:

$\boxed{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}$

$$\frac{\Gamma, t : \kappa_2 \vdash \tau_1 : \kappa \quad \Gamma \vdash \tau_2 : \kappa_2}{\Gamma \vdash (\lambda t : \kappa_2 .\, \tau_1)\ \tau_2 \equiv \tau_1[t \leftarrow \tau_2] : \kappa} \qquad \frac{t \notin tyfv(\tau) \quad \Gamma \vdash \tau : \kappa_1 \to \kappa_2}{\Gamma \vdash (\lambda t : \kappa_1 .\, \tau\ t) \equiv \tau : \kappa_1 \to \kappa_2}$$

$$\frac{\tau_1 \equiv_\alpha \tau_2 \quad \Gamma \vdash \tau_1 : \kappa \quad \Gamma \vdash \tau_2 : \kappa}{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa} \qquad \frac{\Gamma \vdash \tau_2 \equiv \tau_1 : \kappa}{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa} \qquad \frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa \quad \Gamma \vdash \tau_2 \equiv \tau_3 : \kappa}{\Gamma \vdash \tau_1 \equiv \tau_3 : \kappa}$$

$$\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \Omega \quad \Gamma \vdash \tau_1' \equiv \tau_2' : \Omega}{\Gamma \vdash \tau_1 \to \tau_1' \equiv \tau_2 \to \tau_2' : \Omega} \qquad \frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau_2 : \Omega}{\Gamma \vdash \forall t : \kappa .\, \tau_1 \equiv \forall t : \kappa .\, \tau_2 : \Omega}$$

$$\frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau_2 : \kappa'}{\Gamma \vdash \lambda t : \kappa .\, \tau_1 \equiv \lambda t : \kappa .\, \tau_2 : \kappa \to \kappa'} \qquad \frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa' \to \kappa \quad \Gamma \vdash \tau_1' \equiv \tau_2' : \kappa'}{\Gamma \vdash \tau_1\ \tau_1' \equiv \tau_2\ \tau_2' : \kappa}$$

Typing:

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

$$\frac{\Gamma \vdash \tau : \Omega \quad \Gamma(x) = \tau}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma \vdash \tau \equiv \tau' : \Omega \quad \Gamma \vdash e : \tau'}{\Gamma \vdash e : \tau}$$

$$\frac{\Gamma \vdash \tau_1 : \Omega \quad \Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1 .\, e : \tau_1 \to \tau_2} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau}$$

$$\frac{\bigwedge_l \Gamma \vdash e_l : \tau_l}{\Gamma \vdash \{\overline{l = e_l}\} : \{\overline{l = \tau_l}\}} \qquad \frac{\Gamma \vdash e : \{\overline{l' = \tau_{l'}}\}}{\Gamma \vdash e.l : \tau_l}$$

$$\frac{\Gamma, t : \kappa \vdash e : \tau}{\Gamma \vdash \Lambda t : \kappa .\, e : (\forall t : \kappa .\, \tau)} \qquad \frac{\Gamma \vdash e : (\forall t : \kappa .\, \tau_1) \quad \Gamma \vdash \tau_2 : \kappa}{\Gamma \vdash e\ \tau_2 : \tau_1[t \leftarrow \tau_2]}$$

$$\frac{\Gamma, t : \kappa \vdash \tau : \Omega \quad \Gamma \vdash \tau_t : \kappa \quad \Gamma \vdash e : \tau[t \leftarrow \tau_t]}{\Gamma \vdash \mathrm{pack}\langle \tau_t, e\rangle_{\exists t : \kappa .\tau} : (\exists t : \kappa .\, \tau)} \qquad \frac{\Gamma \vdash e_1 : (\exists t : \kappa .\, \tau_1) \quad \Gamma, t : \kappa, x : \tau_1 \vdash e_2 : \tau}{\Gamma \vdash \mathrm{unpack}\langle t : \kappa, x : \tau_1\rangle = e_1 \ \mathrm{in}\ e_2 : \tau}$$

Reduction:

$$\begin{array}{lcl} v & ::= & \lambda x : \tau .\, e \mid \{\overline{l = e}\} \mid \Lambda t : \kappa .\, e \mid \mathrm{pack}\langle \tau_t, e\rangle_{\exists t : \kappa .\tau} \\ C & ::= & [\,] \mid C\ e \mid v\ C \mid \{\overline{l = v}, l = C, \overline{l = e}\} \mid C.l \mid C\ \tau \mid \mathrm{pack}\langle \tau, C\rangle_\tau \mid \mathrm{unpack}\langle t : \kappa, x : \tau\rangle = C \ \mathrm{in}\ e \end{array}$$

$\boxed{e \Rightarrow e'}$

$$\frac{}{(\lambda x : \tau .\, e) v \Rightarrow e[x \leftarrow v]} \qquad \frac{}{\{\overline{l' = v_{l'}}\}.l \Rightarrow v_l} \qquad \frac{}{(\Lambda t : \kappa .\, e)\tau \Rightarrow e[t \leftarrow \tau]}$$

$$\frac{}{\mathrm{unpack}\langle t : \kappa, x : \tau\rangle = \mathrm{pack}\langle \tau_t, v\rangle_{\tau_3} \ \mathrm{in}\ e \Rightarrow e[t \leftarrow \tau_t][x \leftarrow v]} \qquad \frac{e \Rightarrow e'}{C[e] \Rightarrow C[e']}$$

Equivalence:

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau}$

$$\frac{\Gamma, x : \tau_2 \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\lambda x : \tau_2 .\, e_1)\ e_2 \equiv e_1[x \leftarrow e_2] : \tau} \qquad \frac{x \notin fv(e) \quad \Gamma \vdash e : \tau_1 \to \tau_2}{\Gamma \vdash (\lambda x : \tau_1 .\, e\ x) \equiv e : \tau_1 \to \tau_2}$$

$$\frac{\bigwedge_{l'} \Gamma \vdash e_{l'} : \tau_{l'}}{\Gamma \vdash \{\overline{l' = e_{l'}}\}.l \equiv e_l : \tau_l} \qquad \frac{\Gamma \vdash e : \{\overline{l : \tau_l}\}}{\Gamma \vdash \{\overline{l = e.l}\} \equiv e : \{\overline{l : \tau_l}\}}$$

$$\frac{\Gamma, t : \kappa \vdash e : \tau}{\Gamma \vdash (\Lambda t : \kappa .\, e)\ \tau_2 \equiv e[t \leftarrow \tau_2] : \tau[t \leftarrow \tau_2]} \qquad \frac{t \notin tyfv(e) \quad \Gamma \vdash e : \forall t : \kappa .\, \tau}{\Gamma \vdash (\Lambda t : \kappa .\, e\ t) \equiv e : \forall t : \kappa .\, \tau}$$

$$\frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau_1' : \Omega \quad \Gamma \vdash \tau_t : \kappa \quad \Gamma \vdash e_1 : \tau_1[t \leftarrow \tau_t] \quad \Gamma, t : \kappa, x : \tau_1 \vdash e_2 : \tau}{\Gamma \vdash \mathrm{unpack}\langle t : \kappa, x : \tau_1'\rangle = \mathrm{pack}\langle \tau_t, e_1\rangle_{\exists t : \kappa .\tau_1} \ \mathrm{in}\ e_2 \equiv e_2[t \leftarrow \tau_t][x \leftarrow e_1] : \tau}$$

$$\frac{\Gamma \vdash e : \exists t : \kappa .\, \tau \quad \Gamma, t : \kappa \vdash \tau \equiv \tau' : \Omega}{\Gamma \vdash \mathrm{unpack}\langle t : \kappa, x : \tau'\rangle = e \ \mathrm{in}\ \mathrm{pack}\langle t, x\rangle_{\exists t : \kappa .\tau} \equiv e : (\exists t : \kappa .\, \tau)}$$

$$\frac{e_1 \equiv_\alpha e_2 \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau} \qquad \frac{\Gamma \vdash \tau \equiv \tau' : \Omega \quad \Gamma \vdash e_1 \equiv e_2 : \tau'}{\Gamma \vdash e_1 \equiv e_2 : \tau}$$

$$\frac{\Gamma \vdash e_2 \equiv e_1 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau \quad \Gamma \vdash e_2 \equiv e_3 : \tau}{\Gamma \vdash e_1 \equiv e_3 : \tau}$$

$$\frac{\Gamma, x : \tau \vdash e_1 \equiv e_2 : \tau'}{\Gamma \vdash \lambda x : \tau.\, e_1 \equiv \lambda x : \tau.\, e_2 : \tau \to \tau'} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau' \to \tau \quad \Gamma \vdash e_1' \equiv e_2' : \tau'}{\Gamma \vdash e_1\, e_1' \equiv e_2\, e_2' : \tau}$$

$$\frac{\bigwedge_l \Gamma \vdash e_{l,1} \equiv e_{l,2} : \tau_l}{\Gamma \vdash \{\overline{l = e_{l,1}}\} \equiv \{\overline{l = e_{l,2}}\} : \{\overline{l : \tau_l}\}} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \{l : \tau_l, \overline{l' : \tau'}\}}{\Gamma \vdash e_1.l \equiv e_2.l : \tau_l}$$

$$\frac{\Gamma, t : \kappa \vdash e_1 \equiv e_2 : \tau}{\Gamma \vdash \Lambda t : \kappa.\, e_1 \equiv \Lambda t : \kappa.\, e_2 : (\forall t : \kappa.\, \tau)} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \forall t : \kappa.\, \tau \quad \Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}{\Gamma \vdash e_1\, \tau_1 \equiv e_2\, \tau_2 : \tau[t \leftarrow \tau_1]}$$

$$\frac{\Gamma \vdash \tau_1' \equiv \tau_2' : \kappa \quad \Gamma \vdash e_1 \equiv e_2 : \tau_1[t \leftarrow \tau_1'] \quad \Gamma, t : \kappa.\, \tau_1 \equiv \tau_2 : \Omega}{\Gamma \vdash \mathrm{pack}\langle \tau_1', e_1 \rangle_{\exists t : \kappa.\tau_1} \equiv \mathrm{pack}\langle \tau_2', e_2 \rangle_{\exists t : \kappa.\tau_2} : (\exists t : \kappa.\, \tau_1)}$$

$$\frac{\Gamma, t : \kappa \vdash \tau_1' \equiv \tau_2' : \Omega \quad \Gamma \vdash e_1' \equiv e_2' : (\exists t : \kappa.\, \tau_1') \quad \Gamma, t : \kappa, x : \tau_1' \vdash e_1 \equiv e_2 : \tau}{\Gamma \vdash \mathrm{unpack}\langle t : \kappa, x : \tau_1' \rangle = e_1' \text{ in } e_1 \equiv \mathrm{unpack}\langle t : \kappa, x : \tau_2' \rangle = e_2' \text{ in } e_2 : \tau}$$

4.1.2 Syntax

$$\begin{array}{rcll}
X & ::= & \cdots & \text{(identifier)} \\
K & ::= & \cdots & \text{(kind)} \\
T & ::= & \cdots \mid P & \text{(type)} \\
E & ::= & \cdots \mid P & \text{(expression)} \\
P & ::= & M & \text{(path)} \\
M & ::= & X & \text{(identifier)} \\
 & \mid & \{B\} & \text{(bindings)} \\
 & \mid & M.X & \text{(projection)} \\
B & ::= & \mathrm{val}\, X = E & \text{(value binding)} \\
 & \mid & \mathrm{type}\, X = T & \text{(type binding)} \\
 & \mid & \mathrm{module}\, X = M & \text{(module binding)} \\
 & \mid & \mathrm{signature}\, X = S & \text{(signature binding)} \\
 & \mid & \mathrm{include}\, M & \text{(module including)} \\
 & \mid & \epsilon & \text{(empty binding)} \\
 & \mid & B; B & \text{(binding concatenation)} \\
S & ::= & P & \text{(signature path)} \\
 & \mid & \{D\} & \text{(declarations)} \\
D & ::= & \mathrm{val}\, X : T & \text{(value declaration)} \\
 & \mid & \mathrm{type}\, X = T & \text{(type binding)} \\
 & \mid & \mathrm{module}\, X : S & \text{(module declaration)} \\
 & \mid & \mathrm{signature}\, X = S & \text{(signature binding)} \\
 & \mid & \mathrm{include}\, S & \text{(signature including)} \\
 & \mid & \epsilon & \text{(empty declaration)} \\
 & \mid & D; D & \text{(declaration concatenation)}
\end{array}$$

4.1.3 Signature

$$\begin{array}{rcll}
\Sigma & ::= & [\tau] & \text{(anonymous value declaration)} \\
 & \mid & [= \tau : \kappa] & \text{(anonymous type declaration)} \\
 & \mid & [= \Sigma] & \text{(anonymous signature declaration)} \\
 & \mid & \{\overline{l_X : \Sigma}\} & \text{(structural signature)}
\end{array}$$

Atomic Signature:

$$[\tau] \stackrel{\mathrm{def}}{=} \{\mathrm{val} : \tau\}$$

$$[e] \stackrel{\text{def}}{=} \{\text{val} = e\}$$
$$[= \tau : \kappa] \stackrel{\text{def}}{=} \{\text{type} : \forall t : (\kappa \to \Omega).\, t\ \tau \to t\ \tau\}$$
$$[\tau : \kappa] \stackrel{\text{def}}{=} \{\text{type} = \Lambda t : (\kappa \to \Omega).\, \lambda x : (t\ \tau).\, x\}$$
$$[= \Sigma] \stackrel{\text{def}}{=} \{\text{sig} : \Sigma \to \Sigma\}$$
$$[\Sigma] \stackrel{\text{def}}{=} \{\text{sig} = \lambda x : \Sigma.\, x\}$$

$\boxed{\text{NotAtomic}(\Sigma)}$

$$\frac{}{\text{NotAtomic}(\{\overline{l_X : \Sigma}\})}$$

Admissible kinding:

$\boxed{\Gamma \vdash \tau : \kappa}$

$$\frac{\Gamma \vdash \tau : \Omega}{\Gamma \vdash [\tau] : \Omega}\ \text{K-A-Val}$$
$$\frac{\Gamma \vdash \tau : \kappa}{\Gamma \vdash [= \tau : \kappa] : \Omega}\ \text{K-A-Typ}$$
$$\frac{\Gamma \vdash \Sigma : \Omega}{\Gamma \vdash [= \Sigma] : \Omega}\ \text{K-A-Sig}$$

Admissible type equivalence:

$\boxed{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}$

$$\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \Omega}{\Gamma \vdash [\tau_1] \equiv [\tau_2] : \Omega}\ \text{T-Eq-Cong-A-Val}$$
$$\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}{\Gamma \vdash [= \tau_1 : \kappa] \equiv [= \tau_2 : \kappa] : \Omega}\ \text{T-Eq-Cong-A-Typ}$$
$$\frac{\Gamma \vdash \Sigma_1 \equiv \Sigma_2 : \Omega}{\Gamma \vdash [= \Sigma_1] \equiv [= \Sigma_2] : \Omega}\ \text{T-Eq-Cong-A-Sig}$$

Admissible typing:

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

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash [e] : [\tau]}\ \text{T-A-Val}$$
$$\frac{\Gamma \vdash \tau : \kappa}{\Gamma \vdash [\tau : \kappa] : [= \tau : \kappa]}\ \text{T-A-Typ}$$
$$\frac{\Gamma \vdash \Sigma : \Omega}{\Gamma \vdash [\Sigma] : [= \Sigma]}\ \text{T-A-Sig}$$

Admissible equivalence:

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau}$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash [e].\text{val} \equiv e : \tau}\ \text{Eq-}\beta\text{-A-Val} \qquad \frac{\Gamma \vdash e : [\tau]}{\Gamma \vdash [e.\text{val}] \equiv e : [\tau]}\ \text{Eq-}\eta\text{-A-Val} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau}{\Gamma \vdash [e_1] \equiv [e_2] : [\tau]}\ \text{Eq-Cong-A-Val}$$
$$\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}{\Gamma \vdash [\tau_1 : \kappa] \equiv [\tau_2 : \kappa] : [= \tau_1 : \kappa]}\ \text{Eq-Cong-A-Typ}$$
$$\frac{\Gamma \vdash \Sigma_1 \equiv \Sigma_2 : \Omega}{\Gamma \vdash [\Sigma_1] \equiv [\Sigma_2] : [= \Sigma_1]}\ \text{Eq-Cong-A-Sig}$$

4.1.4 Elaboration

Signature:

$\boxed{\Gamma \vdash S \rightsquigarrow \Sigma}$

$$\frac{\Gamma \vdash P : [= \Sigma] \rightsquigarrow e}{\Gamma \vdash P \rightsquigarrow \Sigma}\ \text{S-Path}$$

$$\frac{\Gamma \vdash D \rightsquigarrow \Sigma}{\Gamma \vdash \{D\} \rightsquigarrow \Sigma}\ \text{S-Struct}$$

Declarations:

$\boxed{\Gamma \vdash D \rightsquigarrow \Sigma}$

$$\frac{\Gamma \vdash T : \Omega \rightsquigarrow \tau}{\Gamma \vdash \mathsf{val}\, X : T \rightsquigarrow \{l_X : [\tau]\}}\ \text{D-Val}$$

$$\frac{\Gamma \vdash T : \kappa \rightsquigarrow \tau}{\Gamma \vdash \mathsf{type}\, X = T \rightsquigarrow \{l_X : [= \tau : \kappa]\}}\ \text{D-Typ-Eq}$$

$$\frac{\Gamma \vdash S \rightsquigarrow \Sigma}{\Gamma \vdash \mathsf{module}\, X : S \rightsquigarrow \{l_X : \Sigma\}}\ \text{D-Mod}$$

$$\frac{\Gamma \vdash S \rightsquigarrow \Sigma}{\Gamma \vdash \mathsf{signature}\, X = S \rightsquigarrow \{l_X : [= \Sigma]\}}\ \text{D-Sig-Eq}$$

$$\frac{\Gamma \vdash S \rightsquigarrow \{\overline{l_X : \Sigma}\}}{\Gamma \vdash \mathsf{include}\, S \rightsquigarrow \{\overline{l_X : \Sigma}\}}\ \text{D-Incl}$$

$$\frac{}{\Gamma \vdash \epsilon \rightsquigarrow \{\}}\ \text{D-Emt}$$

$$\frac{\{\overline{l_{X_1}}\} \cap \{\overline{l_{X_2}}\} = \varnothing \quad \Gamma \vdash D_1 \rightsquigarrow \{\overline{l_{X_1} : \Sigma_1}\} \quad \Gamma, \overline{x_{X_1} : \Sigma_1} \vdash D_2 \rightsquigarrow \{\overline{l_{X_2} : \Sigma_2}\}}{\Gamma \vdash D_1; D_2 \rightsquigarrow \{\overline{l_{X_1} : \Sigma_1}, \overline{l_{X_2} : \Sigma_2}\}}\ \text{D-Seq}$$

Module:

$\boxed{\Gamma \vdash M : \Sigma \rightsquigarrow e}$

$$\frac{\Gamma(x_X) = \Sigma}{\Gamma \vdash X : \Sigma \rightsquigarrow x_X}\ \text{M-Var}$$

$$\frac{\Gamma \vdash B : \Sigma \rightsquigarrow e}{\Gamma \vdash \{B\} : \Sigma \rightsquigarrow e}\ \text{M-Struct}$$

$$\frac{\Gamma \vdash M : \{l_X : \Sigma, \overline{l_{X'} : \Sigma'}\} \rightsquigarrow e}{\Gamma \vdash M.X : \Sigma \rightsquigarrow e.l_X}\ \text{M-Dot}$$

Bindings:

$\boxed{\Gamma \vdash B : \Sigma \rightsquigarrow e}$

$$\frac{\Gamma \vdash E : \tau \rightsquigarrow e}{\Gamma \vdash \mathsf{val}\, X = E : \{l_X : [\tau]\} \rightsquigarrow \{l_X = [e]\}}\ \text{B-Val}$$

$$\frac{\Gamma \vdash T : \kappa \rightsquigarrow \tau}{\Gamma \vdash \mathsf{type}\, X = T : \{l_X : [= \tau : \kappa]\} \rightsquigarrow \{l_X = [\tau : \kappa]\}}\ \text{B-Typ}$$

$$\frac{\Gamma \vdash M : \Sigma \rightsquigarrow e \quad \mathrm{NotAtomic}(\Sigma)}{\Gamma \vdash \mathsf{module}\, X = M : \{l_X : \Sigma\} \rightsquigarrow \{l_X = e\}}\ \text{B-Mod}$$

$$\frac{\Gamma \vdash S \rightsquigarrow \Sigma}{\Gamma \vdash \mathsf{signature}\, X = S : \{l_X : [= \Sigma]\} \rightsquigarrow \{l_X = [\Sigma]\}}\ \text{B-Sig}$$

$$\frac{\Gamma \vdash M : \{\overline{l_X : \Sigma}\} \rightsquigarrow e}{\Gamma \vdash \mathsf{include}\, M : \{\overline{l_X : \Sigma}\} \rightsquigarrow e}\ \text{B-Incl}$$

$$\frac{}{\Gamma \vdash \epsilon : \{\} \rightsquigarrow \{\}} \text{ B-Emt}$$

$$\frac{\begin{array}{ll} \overline{l'_{X_1}} = \overline{l_{X_1}} \setminus \overline{l_{X_2}} \quad \overline{l'_{X_1} : \Sigma'_1} \subseteq \overline{l_{X_1} : \Sigma_1} & \Gamma \vdash B_1 : \{\overline{l_{X_1} : \Sigma_1}\} \rightsquigarrow e_1 \\ \Sigma = \{\overline{l'_{X_1} : \Sigma'_1}, \overline{l_{X_2} : \Sigma_2}\} & \Gamma, \overline{x_{X_1} : \Sigma_1} \vdash B_2 : \{\overline{l_{X_2} : \Sigma_2}\} \rightsquigarrow e_2 \end{array}}{\Gamma \vdash B_1; B_2 : \Sigma \rightsquigarrow \begin{array}{c} \text{let } x_1 = e_1 \text{ in} \\ \text{let } x_2 = (\text{let } \overline{x_{X_1} : \Sigma_1 = x_1.l_{X_1}} \text{ in } e_2) \text{ in} \\ \{\overline{l'_{X_1} = x_1.l'_{X_1}}, \overline{l_{X_2} = x_2.l_{X_2}}\} \end{array}} \text{ B-Seq}$$

Path:

$\boxed{\Gamma \vdash P : \Sigma \rightsquigarrow e}$

Use M-Dot.

$\boxed{\Gamma \vdash T : \kappa \rightsquigarrow \tau}$

$$\frac{\Gamma \vdash P : [= \tau : \kappa] \rightsquigarrow e}{\Gamma \vdash P : \kappa \rightsquigarrow \tau} \text{ T-Elab-Path}$$

$\boxed{\Gamma \vdash E : \tau \rightsquigarrow e}$

$$\frac{\Gamma \vdash P : [\tau] \rightsquigarrow e}{\Gamma \vdash P : \tau \rightsquigarrow e.\text{val}} \text{ E-Path}$$

4.2 F-ing modules

[RRD14]

4.2.1 Internal Language

See 第 4.1.1 小節.

4.2.2 Syntax

$$
\begin{array}{rcll}
X & ::= & \cdots & \text{(identifier)} \\
K & ::= & \cdots & \text{(kind)} \\
T & ::= & \cdots \mid P & \text{(type)} \\
E & ::= & \cdots \mid P & \text{(expression)} \\
P & ::= & M & \text{(path)} \\
M & ::= & X & \text{(identifier)} \\
 & \mid & \{B\} & \text{(bindings)} \\
 & \mid & M.X & \text{(projection)} \\
 & \mid & \mathrm{fun}\, X : S \Rightarrow M & \text{(functor)} \\
 & \mid & X\, X & \text{(functor application)} \\
 & \mid & X :> S & \text{(sealing)} \\
B & ::= & \mathrm{val}\, X = E & \text{(value binding)} \\
 & \mid & \mathrm{type}\, X = T & \text{(type binding)} \\
 & \mid & \mathrm{module}\, X = M & \text{(module binding)} \\
 & \mid & \mathrm{signature}\, X = S & \text{(signature binding)} \\
 & \mid & \mathrm{include}\, M & \text{(module including)} \\
 & \mid & \epsilon & \text{(empty binding)} \\
 & \mid & B; B & \text{(binding concatenation)} \\
S & ::= & P & \text{(signature path)} \\
 & \mid & \{D\} & \text{(declarations)} \\
 & \mid & (X : S) \to S & \text{((generative) functor signature)} \\
 & \mid & S\ \mathrm{where\ type}\, \overline{X} = T & \text{(bounded signature)} \\
D & ::= & \mathrm{val}\, X : T & \text{(value declaration)} \\
 & \mid & \mathrm{type}\, X = T & \text{(type binding)} \\
 & \mid & \mathrm{type}\, X : K & \text{(type declaration)} \\
 & \mid & \mathrm{module}\, X : S & \text{(module declaration)} \\
 & \mid & \mathrm{signature}\, X = S & \text{(signature binding)} \\
 & \mid & \mathrm{include}\, S & \text{(signature including)} \\
 & \mid & \epsilon & \text{(empty declaration)} \\
 & \mid & D; D & \text{(declaration concatenation)}
\end{array}
$$

4.2.3 Signature

$$
\begin{array}{rcll}
\Xi & ::= & \exists \overline{t : \kappa}. \Sigma & \text{(abstract signature)} \\
\Sigma & ::= & [\tau] & \text{(atomic value declaration)} \\
 & \mid & [= \tau : \kappa] & \text{(atomic type declaration)} \\
 & \mid & [= \Xi] & \text{(atomic signature declaration)} \\
 & \mid & \{\overline{l_X : \Sigma}\} & \text{(structure signature)} \\
 & \mid & \forall \overline{t : \kappa}. \Sigma \to \Xi & \text{(functor signature)}
\end{array}
$$

Atomic Signature:

$$[\tau] \stackrel{\mathrm{def}}{=} \{\mathrm{val} : \tau\}$$

$$
\begin{aligned}
[e] &\stackrel{\text{def}}{=} \{\text{val} = e\} \\
[= \tau : \kappa] &\stackrel{\text{def}}{=} \{\text{type} : \forall t : (\kappa \to \Omega).\, t\ \tau \to t\ \tau\} \\
[\tau : \kappa] &\stackrel{\text{def}}{=} \{\text{type} = \Lambda t : (\kappa \to \Omega).\, \lambda x : (t\ \tau).\, x\} \\
[= \Xi] &\stackrel{\text{def}}{=} \{\text{sig} : \Xi \to \Xi\} \\
[\Xi] &\stackrel{\text{def}}{=} \{\text{sig} = \lambda x : \Xi.\, x\}
\end{aligned}
$$

$\boxed{\text{NotAtomic}(\Sigma)}$

$$
\frac{}{\text{NotAtomic}(\{\overline{l_X : \Sigma}\})} \qquad \frac{}{\text{NotAtomic}(\forall \overline{t : \kappa}.\, \Sigma \to \Xi)}
$$

Admissible kinding:

$\boxed{\Gamma \vdash \tau : \kappa}$

$$
\frac{\Gamma \vdash \tau : \Omega}{\Gamma \vdash [\tau] : \Omega}\ \text{K-A-Val}
$$

$$
\frac{\Gamma \vdash \tau : \kappa}{\Gamma \vdash [= \tau : \kappa] : \Omega}\ \text{K-A-Typ}
$$

$$
\frac{\Gamma \vdash \Xi : \Omega}{\Gamma \vdash [= \Xi] : \Omega}\ \text{K-A-Sig}
$$

Admissible type equivalence:

$\boxed{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}$

$$
\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \Omega}{\Gamma \vdash [\tau_1] \equiv [\tau_2] : \Omega}\ \text{T-Eq-Cong-A-Val}
$$

$$
\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}{\Gamma \vdash [= \tau_1 : \kappa] \equiv [= \tau_2 : \kappa] : \Omega}\ \text{T-Eq-Cong-A-Typ}
$$

$$
\frac{\Gamma \vdash \Xi_1 \equiv \Xi_2 : \Omega}{\Gamma \vdash [= \Xi_1] \equiv [= \Xi_2] : \Omega}\ \text{T-Eq-Cong-A-Sig}
$$

Admissible typing:

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

$$
\frac{\Gamma \vdash e : \tau}{\Gamma \vdash [e] : [\tau]}\ \text{T-A-Val}
$$

$$
\frac{\Gamma \vdash \tau : \kappa}{\Gamma \vdash [\tau : \kappa] : [= \tau : \kappa]}\ \text{T-A-Typ}
$$

$$
\frac{\Gamma \vdash \Xi : \Omega}{\Gamma \vdash [\Xi] : [= \Xi]}\ \text{T-A-Sig}
$$

Admissible equivalence:

$\boxed{\Gamma \vdash e_1 \equiv e_2 : \tau}$

$$
\frac{\Gamma \vdash e : \tau}{\Gamma \vdash [e].\text{val} \equiv e : \tau}\ \text{Eq-}\beta\text{-A-Val} \qquad \frac{\Gamma \vdash e : [\tau]}{\Gamma \vdash [e.\text{val}] \equiv e : [\tau]}\ \text{Eq-}\eta\text{-A-Val} \qquad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau}{\Gamma \vdash [e_1] \equiv [e_2] : [\tau]}\ \text{Eq-Cong-A-Val}
$$

$$
\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}{\Gamma \vdash [\tau_1 : \kappa] \equiv [\tau_2 : \kappa] : [= \tau_1 : \kappa]}\ \text{Eq-Cong-A-Typ}
$$

$$
\frac{\Gamma \vdash \Xi_1 \equiv \Xi_2 : \Omega}{\Gamma \vdash [\Xi_1] \equiv [\Xi_2] : [= \Xi_1]}\ \text{Eq-Cong-A-Sig}
$$

4.2.4 (Generative) Elaboration

Signature:

$\boxed{\Gamma \vdash S \rightsquigarrow \Xi}$

$$\frac{\Gamma \vdash P : [= \Xi] \rightsquigarrow e}{\Gamma \vdash P \rightsquigarrow \Xi} \text{ S-Path}$$

$$\frac{\Gamma \vdash D \rightsquigarrow \Xi}{\Gamma \vdash \{D\} \rightsquigarrow \Xi} \text{ S-Struct}$$

$$\frac{\Gamma \vdash S_1 \rightsquigarrow \exists \overline{t : \kappa}. \Sigma \quad \Gamma, \overline{t : \kappa}, x_X : \Sigma \vdash S_2 \rightsquigarrow \Xi}{\Gamma \vdash (X : S_1) \to S_2 \rightsquigarrow \forall \overline{t : \kappa}. \Sigma \to \Xi} \text{ S-Funct}$$

$$\frac{\Gamma \vdash S \rightsquigarrow \exists \overline{t_1 : \kappa_1}\; t : \kappa\; \overline{t_2 : \kappa_2}. \Sigma \quad \Sigma.\overline{l_X} = [= t : \kappa] \quad \Gamma \vdash T : \kappa \rightsquigarrow \tau}{\Gamma \vdash S \text{ where type} \overline{X} = T \rightsquigarrow \exists \overline{t_1 : \kappa_1}\; \overline{t_2 : \kappa_2}. \Sigma[t \leftarrow \tau]} \text{ S-Where-Typ}$$

Declarations:

$\boxed{\Gamma \vdash D \rightsquigarrow \Xi}$

$$\frac{\Gamma \vdash T : \Omega \rightsquigarrow \tau}{\Gamma \vdash \text{val}\, X : T \rightsquigarrow \{l_X : [\tau]\}} \text{ D-Val}$$

$$\frac{\Gamma \vdash T : \kappa \rightsquigarrow \tau}{\Gamma \vdash \text{type}\, X = T \rightsquigarrow \{l_X : [= \tau : \kappa]\}} \text{ D-Typ-Eq}$$

$$\frac{\Gamma \vdash K \rightsquigarrow \kappa}{\Gamma \vdash \text{type}\, X : K \rightsquigarrow \exists t : \kappa. \{l_X : [= t : \kappa]\}} \text{ D-Typ}$$

$$\frac{\Gamma \vdash S \rightsquigarrow \exists \overline{t : \kappa}. \Sigma}{\Gamma \vdash \text{module}\, X : S \rightsquigarrow \exists \overline{t : \kappa}. \{l_X : \Sigma\}} \text{ D-Mod}$$

$$\frac{\Gamma \vdash S \rightsquigarrow \Xi}{\Gamma \vdash \text{signature}\, X = S \rightsquigarrow \{l_X : [= \Xi]\}} \text{ D-Sig-Eq}$$

$$\frac{\Gamma \vdash S \rightsquigarrow \exists \overline{t : \kappa}. \{\overline{l_X : \Sigma}\}}{\Gamma \vdash \text{include}\, S \rightsquigarrow \exists \overline{t : \kappa}. \{\overline{l_X : \Sigma}\}} \text{ D-Incl}$$

$$\frac{}{\Gamma \vdash \epsilon \rightsquigarrow \{\}} \text{ D-Emt}$$

$$\frac{\{\overline{l_{X_1}}\} \cap \{\overline{l_{X_2}}\} = \varnothing \quad \Gamma \vdash D_1 \rightsquigarrow \exists \overline{t_1 : \kappa_1}. \{\overline{l_{X_1} : \Sigma_1}\} \quad \Gamma, \overline{t_1 : \kappa_1}, \overline{x_{X_1} : \Sigma_1} \vdash D_2 \rightsquigarrow \exists \overline{t_2 : \kappa_2}. \{\overline{l_{X_2} : \Sigma_2}\}}{\Gamma \vdash D_1; D_2 \rightsquigarrow \exists \overline{t_1 : \kappa_1}\; \overline{t_2 : \kappa_2}. \{\overline{l_{X_1} : \Sigma_1}\; \overline{l_{X_2} : \Sigma_2}\}} \text{ D-Seq}$$

Matching:

$\boxed{\Gamma \vdash \Sigma_1 \leq \exists \overline{t : \kappa}. \Sigma_2 \uparrow \overline{\tau} \rightsquigarrow e}$

$$\frac{\Gamma \vdash \Sigma_1 \leq \Sigma_2[\overline{t \leftarrow \tau_t}] \rightsquigarrow e \quad \bigwedge_t \Gamma \vdash \tau_t : \kappa_t}{\Gamma \vdash \Sigma_1 \leq \exists \overline{t : \kappa_t}. \Sigma_2 \uparrow \overline{\tau_t} \rightsquigarrow e} \text{ U-Match}$$

Subtyping:

$\boxed{\Gamma \vdash \Xi_1 \leq \Xi_2 \rightsquigarrow e}$

$$\frac{\Gamma \vdash \tau_1 \leq \tau_2 \rightsquigarrow e}{\Gamma \vdash [\tau_1] \leq [\tau_2] \rightsquigarrow \lambda x : [\tau_1]. [e\,(x.\text{val})]} \text{ U-Val}$$

$$\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}{\Gamma \vdash [= \tau_1 : \kappa] \leq [= \tau_2 : \kappa] \rightsquigarrow \lambda x : [= \tau_1 : \kappa]. x} \text{ U-Typ}$$

$$\frac{\Gamma \vdash \Xi_1 \leq \Xi_2 \rightsquigarrow e_1 \quad \Gamma \vdash \Xi_2 \leq \Xi_1 \rightsquigarrow e_2}{\Gamma \vdash [= \Xi_1] \leq [= \Xi_2] \rightsquigarrow \lambda x : [= \Xi_1]. [\Xi_2]} \text{ U-Sig}$$

$$\frac{\bigwedge_l \Gamma \vdash \Sigma_{l_1} \leq \Sigma_{l_2} \rightsquigarrow e_l}{\Gamma \vdash \{\overline{l : \Sigma_{l_1}}, \overline{l' : \Sigma'}\} \leq \{\overline{l : \Sigma_{l_2}}\} \rightsquigarrow \lambda x : \{\overline{l : \Sigma_{l_1}}, \overline{l' : \Sigma'}\}. \{\overline{l = e_l\ (x.l)}\}}\ \text{U-Struct}$$

$$\frac{\Gamma, \overline{t_2 : \kappa_2} \vdash \Sigma_2 \leq \exists\overline{t_1 : \kappa_1}. \Sigma_1 \uparrow \overline{\tau} \rightsquigarrow e_1 \quad \Gamma, \overline{t_2 : \kappa_2} \vdash \Xi_1[\overline{t_1 \leftarrow \tau}] \leq \Xi_2 \rightsquigarrow e_2}{\Gamma \vdash \forall\overline{t_1 : \kappa_1}. \Sigma_1 \to \Xi_1 \leq \forall\overline{t_2 : \kappa_2}. \Sigma_2 \to \Xi_2 \rightsquigarrow \begin{array}{l}\lambda x_1 : (\forall\overline{t_1 : \kappa_1}. \Sigma_1 \to \Xi_1). \\ \quad \lambda x_2 : \Sigma_2. e_2\ (x_1\ \overline{\tau}\ (e_1\ x_2))\end{array}}\ \text{U-Funct}$$

$$\frac{\Gamma, \overline{t_1 : \kappa_1} \vdash \Sigma_1 \leq \exists\overline{t_2 : \kappa_2}. \Sigma_2 \uparrow \overline{\tau} \rightsquigarrow e}{\Gamma \vdash \exists\overline{t_1 : \kappa_1}. \Sigma_1 \leq \exists\overline{t_2 : \kappa_2}. \Sigma_2 \rightsquigarrow \begin{array}{l}\lambda x_1 : (\exists\overline{t_1 : \kappa_1}. \Sigma_1). \\ \quad \text{unpack}\langle\overline{t_1 : \kappa_1}, x_1' : \Sigma_1\rangle = x_1 \text{ in pack}\langle\overline{\tau}, e\ x_1'\rangle_{\exists\overline{t_2 : \kappa_2}. \Sigma_2}\end{array}}\ \text{U-Abs}$$

Module:

$\boxed{\Gamma \vdash M : \Xi \rightsquigarrow e}$

$$\frac{\Gamma(x_X) = \Sigma}{\Gamma \vdash X : \Sigma \rightsquigarrow x_X}\ \text{M-Var}$$

$$\frac{\Gamma \vdash B : \Xi \rightsquigarrow e}{\Gamma \vdash \{B\} : \Xi \rightsquigarrow e}\ \text{M-Struct}$$

$$\frac{\Gamma \vdash M : \exists\overline{t : \kappa}. \{l_X : \Sigma, \overline{l_{X'} : \Sigma'}\} \rightsquigarrow e}{\Gamma \vdash M.X : \exists\overline{t : \kappa}. \Sigma \rightsquigarrow \text{unpack}\langle\overline{t : \kappa}, x : \{l_X : \Sigma, \overline{l_{X'} : \Sigma'}\}\rangle = e \text{ in pack}\langle\overline{t}, x.l_X\rangle_{\exists\overline{t : \kappa}. \Sigma}}\ \text{M-Dot}$$

$$\frac{\Sigma \vdash S \rightsquigarrow \exists\overline{t : \kappa}. \Sigma \quad \Gamma, \overline{t : \kappa}, x_X : \Sigma \vdash M : \Xi \rightsquigarrow e}{\Gamma \vdash \text{fun}\, X : S \Rightarrow M : \forall\overline{t : \kappa}. \Sigma \to \Xi \rightsquigarrow \Lambda\overline{t : \kappa}. \lambda x_X : \Sigma. e}\ \text{M-Funct}$$

$$\frac{\Gamma(x_{X_1}) = \forall\overline{t : \kappa}. \Sigma' \to \Xi \quad \Gamma(x_{X_2}) = \Sigma \quad \Gamma \vdash \Sigma \leq \exists\overline{t : \kappa}. \Sigma' \uparrow \overline{\tau} \rightsquigarrow e}{\Gamma \vdash X_1\ X_2 : \Xi[\overline{t \leftarrow \tau}] \rightsquigarrow x_{X_1}\ \overline{\tau}\ (e\ x_{X_2})}\ \text{M-App}$$

$$\frac{\Gamma(x_X) = \Sigma \quad \Gamma \vdash S \rightsquigarrow \exists\overline{t : \kappa}. \Sigma' \quad \Gamma \vdash \Sigma \leq \exists\overline{t : \kappa}. \Sigma' \uparrow \overline{\tau} \rightsquigarrow e}{\Gamma \vdash X :> S : \exists\overline{t : \kappa}. \Sigma' \rightsquigarrow \text{pack}\langle\overline{\tau}, e\ x_X\rangle_{\exists\overline{t : \kappa}. \Sigma'}}\ \text{M-Seal}$$

Bindings:

$\boxed{\Gamma \vdash B : \Xi \rightsquigarrow e}$

$$\frac{\Gamma \vdash E : \tau \rightsquigarrow e}{\Gamma \vdash \text{val}\, X = E : \{l_X : [\tau]\} \rightsquigarrow \{l_X = [e]\}}\ \text{B-Val}$$

$$\frac{\Gamma \vdash T : \kappa \rightsquigarrow \tau}{\Gamma \vdash \text{type}\, X = T : \{l_X : [= \tau : \kappa]\} \rightsquigarrow \{l_X = [\tau : \kappa]\}}\ \text{B-Typ}$$

$$\frac{\Gamma \vdash M : \exists\overline{t : \kappa}. \Sigma \rightsquigarrow e \quad \text{NotAtomic}(\Sigma)}{\Gamma \vdash \text{module}\, X = M : \exists\overline{t : \kappa}. \{l_X : \Sigma\} \rightsquigarrow \text{unpack}\langle\overline{t : \kappa}, x : \Sigma\rangle = e \text{ in pack}\langle\overline{t}, \{l_X = x\}\rangle_{\exists\overline{t : \kappa}. \{l_X : \Sigma\}}}\ \text{B-Mod}$$

$$\frac{\Gamma \vdash S \rightsquigarrow \Xi}{\Gamma \vdash \text{signature}\, X = S : \{l_X : [= \Xi]\} \rightsquigarrow \{l_X = [\Xi]\}}\ \text{B-Sig}$$

$$\frac{\Gamma \vdash M : \exists\overline{t : \kappa}. \{\overline{l_X : \Sigma}\} \rightsquigarrow e}{\Gamma \vdash \text{include}\, M : \exists\overline{t : \kappa}. \{\overline{l_X : \Sigma}\} \rightsquigarrow e}\ \text{B-Incl}$$

$$\frac{}{\Gamma \vdash \epsilon : \{\} \rightsquigarrow \{\}}\ \text{B-Emt}$$

$$\frac{\begin{array}{ll}\overline{l'_{X_1}} = \overline{l_{X_1}} \setminus \overline{l_{X_2}} \quad \overline{l'_{X_1} : \Sigma'_1} \subseteq \overline{l_{X_1} : \Sigma_1} & \Gamma \vdash B_1 : \exists\overline{t_1 : \kappa_1}. \{\overline{l_{X_1} : \Sigma_1}\} \rightsquigarrow e_1 \\ \Sigma = \{\overline{l'_{X_1} : \Sigma'_1}, \overline{l_{X_2} : \Sigma_2}\} & \Gamma, \overline{t_1 : \kappa_1}, \overline{x_{X_1} : \Sigma_1} \vdash B_2 : \exists\overline{t_2 : \kappa_2}. \{\overline{l_{X_2} : \Sigma_2}\} \rightsquigarrow e_2\end{array}}{\Gamma \vdash B_1; B_2 : \exists\overline{t_1 : \kappa_1}\ \overline{t_2 : \kappa_2}. \Sigma \rightsquigarrow \begin{array}{l}\text{unpack}\langle\overline{t_1 : \kappa_1}, x_1\rangle = e_1 \text{ in} \\ \quad \text{unpack}\langle\overline{t_2 : \kappa_2}, x_2\rangle = (\text{let}\, \overline{x_{X_1} : \Sigma_1 = x_1.l_{X_1}} \text{ in } e_2) \text{ in} \\ \quad\quad \text{pack}\langle\overline{t_1}\ \overline{t_2}, \{\overline{l'_{X_1} = x_1.l'_{X_1}}, \overline{l_{X_2} = x_2.l_{X_2}}\}\rangle_{\exists\overline{t_1 : \kappa_1}\ \overline{t_2 : \kappa_2}. \Sigma}\end{array}}\ \text{B-Seq}$$

Path:

$\boxed{\Gamma \vdash P : \Sigma \rightsquigarrow e}$

$$\frac{\Gamma \vdash P : \exists\overline{t : \kappa}. \Sigma \quad \Gamma \vdash \Sigma : \Omega}{\Gamma \vdash P : \Sigma \rightsquigarrow \text{unpack}\langle\overline{t : \kappa}, x\rangle = e \text{ in } x}\ \text{P-Mod}$$

$\boxed{\Gamma \vdash T : \kappa \rightsquigarrow \tau}$

$$\frac{\Gamma \vdash P : [= \tau : \kappa] \rightsquigarrow e}{\Gamma \vdash P : \kappa \rightsquigarrow \tau}\ \text{T-Elab-Path}$$

$\boxed{\Gamma \vdash E : \tau \rightsquigarrow e}$

$$\frac{\Gamma \vdash P : [\tau] \rightsquigarrow e}{\Gamma \vdash P : \tau \rightsquigarrow e.\mathrm{val}}\ \text{E-Path}$$

4.2.5 Modules as First-Class Values

$$\begin{array}{rcl} T & ::= & \cdots \mid \mathrm{pack}\, S \\ E & ::= & \cdots \mid \mathrm{pack}\, M : S \\ M & ::= & \cdots \mid \mathrm{unpack}\, E : S \end{array}$$

Rootedness:

$\boxed{t : \kappa \text{ rooted in } \Sigma \text{ at } \overline{l_X}}$

$$\frac{t = \tau'}{t : \kappa \text{ rooted in } [= \tau : \kappa] \text{ at } \epsilon} \qquad \frac{t : \kappa \text{ rooted in } \{\overline{l_X : \Sigma}\}.l \text{ at } \overline{l'}}{t : \kappa \text{ rooted in } \{\overline{l_X : \Sigma}\} \text{ at } l\,\overline{l'}}$$

Rooted ordering:

$$t_1 : \kappa_1 \leq_\Sigma t_2 : \kappa_2 \iff \min\{\bar{l} \mid t_1 : \kappa_1 \text{ rooted in } \Sigma \text{ at } \bar{l}\} \leq \min\{\bar{l} \mid t_2 : \kappa_2 \text{ rooted in } \Sigma \text{ at } \bar{l}\}$$

Signature normalization:

$$\frac{\mathrm{norm}_0(\tau) = \tau'}{\mathrm{norm}([\tau]) = [\tau']}$$

$$\frac{}{\mathrm{norm}([= \tau : \kappa]) = [= \tau : \kappa]}$$

$$\frac{\mathrm{norm}(\Xi) = \Xi'}{\mathrm{norm}([= \Xi]) = [= \Xi']}$$

$$\frac{\bigwedge_X \mathrm{norm}(\Sigma_X) = \Sigma'_X}{\mathrm{norm}(\{\overline{l_X : \Sigma_X}\}) = \{\overline{l_X : \Sigma'_X}\}}$$

$$\frac{\mathrm{sort}_{\leq_{\Sigma'}}(\overline{t : \kappa}) = \overline{t' : \kappa'} \quad \mathrm{norm}(\Sigma) = \Sigma' \quad \mathrm{norm}(\Xi) = \Xi'}{\mathrm{norm}(\forall \overline{t : \kappa}.\, \Sigma \to \Xi) = \forall \overline{t' : \kappa'}.\, \Sigma' \to \Xi'}$$

$$\frac{\mathrm{sort}_{\leq_{\Sigma'}}(\overline{t : \kappa}) = \overline{t' : \kappa'} \quad \mathrm{norm}(\Sigma) = \Sigma'}{\mathrm{norm}(\exists \overline{t : \kappa}.\, \Sigma) = \exists \overline{t' : \kappa'}.\, \Sigma'}$$

Type:

$\boxed{\Gamma \vdash T : \kappa \rightsquigarrow \tau}$

$$\frac{\Gamma \vdash S \rightsquigarrow \Xi}{\Gamma \vdash \mathrm{pack}\, S : \Omega \rightsquigarrow \mathrm{norm}(\Xi)}\ \text{T-Pack}$$

Expression:

$\boxed{\Gamma \vdash E : \tau \rightsquigarrow e}$

$$\frac{\Gamma \vdash S \rightsquigarrow \Xi \quad \Gamma \vdash \Xi' \leq \text{norm}(\Xi) \rightsquigarrow e_1 \quad \Gamma \vdash M : \Xi' \rightsquigarrow e_2}{\Gamma \vdash (\text{pack}\, M : S) : \text{norm}(\Xi) \rightsquigarrow e_1\ e_2}\ \text{E-Pack}$$

Module:

$\boxed{\Gamma \vdash M : \Xi \rightsquigarrow e}$

$$\frac{\Gamma \vdash S \rightsquigarrow \Xi \quad \Gamma \vdash E : \text{norm}(\Xi) \rightsquigarrow e}{\Gamma \vdash (\text{unpack}\, E : S) : \text{norm}(\Xi) \rightsquigarrow e}\ \text{M-Unpack}$$

4.2.6 Elaboration with Applicative Functor

$$\begin{array}{lcll} S & ::= & \cdots & \\ & | & (X : S) \Rightarrow S & \text{(applicative functor signature)} \end{array}$$

$$\begin{array}{lcll} \varphi & ::= & \text{I} & \text{(impure effect)} \\ & | & \text{P} & \text{(pure effect)} \\ \Sigma & ::= & \cdots & \\ & | & \{\overline{l_X : \Sigma}\} & \\ & | & \forall \overline{t : \kappa}.\, \Sigma \to_{\text{I}} \Xi & \text{(generative functor signature)} \\ & | & \forall \overline{t : \kappa}.\, \Sigma \to_{\text{P}} \Sigma & \text{(applicative functor signature)} \end{array}$$

Abbreviation:

$$\tau_1 \to_\varphi \tau_2 \stackrel{\text{def}}{=} \tau_1 \to \{l_\varphi : \tau_2\}$$

$$\lambda_\varphi x : \tau.\, e \stackrel{\text{def}}{=} \lambda x : \tau.\, \{l_\varphi = e\}$$

$$(e_1\ e_2)_\varphi \stackrel{\text{def}}{=} (e_1\ e_2).l_\varphi$$

$$\Gamma^\varphi \stackrel{\text{def}}{=} \begin{cases} \cdot & (\varphi = \text{I}) \\ \Gamma & (\varphi = \text{P}) \end{cases}$$

$$tyenv(\Gamma) \stackrel{\text{def}}{=} \begin{cases} tyenv(\Gamma')\ t : \kappa & (\Gamma = \Gamma', t : \kappa) \\ tyenv(\Gamma') & (\Gamma = \Gamma', x : \tau) \\ \epsilon & (\Gamma = \cdot) \end{cases}$$

$$\forall_{\text{P}}\Gamma.\, \tau_0 \stackrel{\text{def}}{=} \begin{cases} \forall_{\text{P}}\Gamma'.\, \forall t : \kappa.\, \tau_0 & (\Gamma = \Gamma', t : \kappa) \\ \forall_{\text{P}}\Gamma'.\, \tau \to_{\text{P}} \tau_0 & (\Gamma = \Gamma', x : \tau) \\ \tau_0 & (\Gamma = \cdot) \end{cases}$$

$$\Lambda_{\text{P}}\Gamma.\, e \stackrel{\text{def}}{=} \begin{cases} \Lambda_{\text{P}}\Gamma'.\, \Lambda t : \kappa.\, e & (\Gamma = \Gamma', t : \kappa) \\ \Lambda_{\text{P}}\Gamma'.\, \lambda_{\text{P}} x : \tau.\, e & (\Gamma = \Gamma', x : \tau) \\ e & (\Gamma = \cdot) \end{cases}$$

$$(e\ \Gamma)_{\text{P}} \stackrel{\text{def}}{=} \begin{cases} (e\ \Gamma')_{\text{P}}\ t & (\Gamma = \Gamma', t : \kappa) \\ ((e\ \Gamma')_{\text{P}}\ x)_{\text{P}} & (\Gamma = \Gamma', x : \tau) \\ e & (\Gamma = \cdot) \end{cases}$$

Effect combining:

$\boxed{\varphi_1 \vee \varphi_2 = \varphi}$

$$\frac{}{\varphi \vee \varphi = \varphi} \qquad \frac{}{\text{I} \vee \text{P} = \text{I}} \qquad \frac{}{\text{P} \vee \text{I} = \text{I}}$$

Subeffects:

$\boxed{\varphi_1 \le \varphi_2}$

$$\frac{}{\varphi \le \varphi}\ \text{F-Refl} \qquad \frac{}{\mathrm{P} \le \mathrm{I}}\ \text{F-Sub}$$

Signature:

$\boxed{\Gamma \vdash S \rightsquigarrow \Xi}$

$$\frac{\Gamma \vdash S_1 \rightsquigarrow \exists\overline{t_1 : \kappa_1}.\,\Sigma \qquad \Gamma, \overline{t_1 : \kappa_1}, x_X : \Sigma \vdash S_2 \rightsquigarrow \Xi}{\Gamma \vdash (X : S_1) \to S_2 \rightsquigarrow \forall\overline{t_1 : \kappa_1}.\,\Sigma \to_{\mathrm{I}} \Xi}\ \text{S-Funct-I}$$

$$\frac{\Gamma \vdash S_1 \rightsquigarrow \exists\overline{t_1 : \kappa_1}.\,\Sigma_1 \qquad \Gamma, \overline{t_1 : \kappa_1}, x_X : \Sigma_1 \vdash S_2 \rightsquigarrow \exists\overline{t_2 : \kappa_2}.\,\Sigma_2}{\Gamma \vdash (X : S_1) \Rightarrow S_2 \rightsquigarrow \exists\overline{t_2' : \overline{\kappa_1} \to \kappa_2}.\forall\overline{t_1 : \kappa_1}.\,\Sigma_1 \to_{\mathrm{P}} \Sigma_2[\overline{t_2 \leftarrow t_2'\ \overline{t_1}}]}\ \text{S-Funct-P}$$

Subtyping:

$\boxed{\Gamma \vdash \Xi_1 \le \Xi_2 \rightsquigarrow e}$

$$\frac{\Gamma, \overline{t_2 : \kappa_2} \vdash \Sigma_2 \le \exists\overline{t_1 : \kappa_1}.\,\Sigma_1 \uparrow \overline{\tau} \rightsquigarrow e_1 \qquad \Gamma, \overline{t_2 : \kappa_2} \vdash \Xi_1[\overline{t_1 \leftarrow \tau}] \le \Xi_2 \rightsquigarrow e_2 \qquad \varphi_1 \le \varphi_2}{\Gamma \vdash (\forall\overline{t_1 : \kappa_1}.\,\Sigma_1 \to_{\varphi_1} \Xi_1) \le (\forall\overline{t_2 : \kappa_2}.\,\Sigma_2 \to_{\varphi_2} \Xi_2) \rightsquigarrow \begin{array}{l}\lambda x_1 : (\forall\overline{t_1 : \kappa_1}.\,\Sigma_1 \to_{\varphi_1} \Xi_1).\\ \Lambda\overline{t_2 : \kappa_2}.\,\lambda_{\varphi_2} x_2 : \Sigma_2.\, e_2\ (x_1\ \overline{\tau}\ (e_1\ x_2))_{\varphi_1}\end{array}}\ \text{U-Funct}$$

Module:

$\boxed{\Gamma \vdash M :_{\varphi} \Xi \rightsquigarrow e}$

$$\frac{\Gamma(x_X) = \Sigma}{\Gamma \vdash X :_{\mathrm{P}} \Sigma \rightsquigarrow \Lambda_{\mathrm{P}}\Gamma.x_X}\ \text{M-Var}$$

$$\frac{\Gamma \vdash B :_{\varphi} \Xi \rightsquigarrow e}{\Gamma \vdash \{B\} :_{\varphi} \Xi \rightsquigarrow e}\ \text{M-Struct}$$

$$\frac{\Gamma \vdash M :_{\varphi} \exists\overline{t : \kappa}.\{l_X : \Sigma, \overline{l_{X'} : \Sigma'}\} \rightsquigarrow e}{\Gamma \vdash M.X :_{\varphi} \exists\overline{t : \kappa}.\,\Sigma \rightsquigarrow \mathrm{unpack}\langle\overline{t : \kappa}, x\rangle = e \ \mathrm{in}\ \mathrm{pack}\langle\overline{t}, \Lambda_{\mathrm{P}}\Gamma^{\varphi}.\,(x\ \Gamma^{\varphi})_{\mathrm{P}}.l_X\rangle}\ \text{M-Dot}$$

$$\frac{\Sigma \vdash S \rightsquigarrow \exists\overline{t : \kappa}.\,\Sigma \qquad \Gamma, \overline{t : \kappa}, x_X : \Sigma \vdash M :_{\mathrm{I}} \Xi \rightsquigarrow e}{\Gamma \vdash \mathrm{fun}\, X : S \Rightarrow M :_{\mathrm{P}} \forall\overline{t : \kappa}.\,\Sigma \to_{\mathrm{I}} \Xi \rightsquigarrow \Lambda_{\mathrm{P}}\Gamma.\Lambda\overline{t : \kappa}.\,\lambda_{\mathrm{I}} x_X : \Sigma.\, e}\ \text{M-Funct-I}$$

$$\frac{\Sigma \vdash S \rightsquigarrow \exists\overline{t : \kappa}.\,\Sigma \qquad \Gamma, \overline{t : \kappa}, x_X : \Sigma \vdash M :_{\mathrm{P}} \exists\overline{t_2 : \kappa_2}.\,\Sigma_2 \rightsquigarrow e}{\Gamma \vdash \mathrm{fun}\, X : S \Rightarrow M :_{\mathrm{P}} \exists\overline{t_2 : \kappa_2}.\forall\overline{t : \kappa}.\,\Sigma \to_{\mathrm{P}} \Sigma_2 \rightsquigarrow e}\ \text{M-Funct-P}$$

$$\frac{\Gamma(x_{X_1}) = \forall\overline{t : \kappa}.\,\Sigma' \to_{\varphi} \Xi \qquad \Gamma(x_{X_2}) = \Sigma \qquad \Gamma \vdash \Sigma \le \exists\overline{t : \kappa}.\,\Sigma' \uparrow \overline{\tau} \rightsquigarrow e}{\Gamma \vdash X_1\ X_2 :_{\varphi} \Xi[\overline{t \leftarrow \tau}] \rightsquigarrow \Lambda_{\mathrm{P}}\Gamma^{\varphi}.\,(x_{X_1}\ \overline{\tau}\ (e\ x_{X_2}))_{\varphi}}\ \text{M-App}$$

$$\frac{\overline{t_\Gamma : \kappa_\Gamma} = tyenv(\Gamma) \qquad \Gamma(x_X) = \Sigma \qquad \Gamma \vdash S \rightsquigarrow \exists\overline{t : \kappa}.\,\Sigma' \qquad \Gamma \vdash \Sigma \le \exists\overline{t : \kappa}.\,\Sigma' \uparrow \overline{\tau} \rightsquigarrow e}{\Gamma \vdash X :> S :_{\mathrm{P}} \exists\overline{t' : \overline{t_\Gamma : \kappa_\Gamma} \to \kappa}.\,\Sigma'[\overline{t \leftarrow t'\ \overline{t_\Gamma}}] \rightsquigarrow \mathrm{pack}\langle\overline{\lambda\overline{t_\Gamma : \kappa_\Gamma}.\tau}, \Lambda_{\mathrm{P}}\Gamma.e\ x_X\rangle}\ \text{M-Seal}$$

$$\frac{\Gamma \vdash S \rightsquigarrow \Xi \qquad \Gamma \vdash E : \mathrm{norm}(\Xi) \rightsquigarrow e}{\Gamma \vdash (\mathrm{unpack}\, E : S) :_{\mathrm{I}} \mathrm{norm}(\Xi) \rightsquigarrow e}\ \text{M-Unpack}$$

定理 37 (Typing for module elaboration).

- $\Gamma \vdash M :_{\mathrm{I}} \Xi \rightsquigarrow e$ ならば，$\Gamma \vdash e : \Xi$.
- $\Gamma \vdash M :_{\mathrm{P}} \exists\overline{t : \kappa}.\,\Sigma \rightsquigarrow e$ ならば，$\cdot \vdash e : \exists\overline{t : \kappa}.\,\forall_{\mathrm{P}}\Gamma.\,\Sigma$.

□

Bindings:

$\boxed{\Gamma \vdash B :_{\varphi} \Xi \rightsquigarrow e}$

$$\frac{\Gamma \vdash E : \tau \leadsto e}{\Gamma \vdash \mathrm{val}\, X = E :_{\mathrm{P}} \{l_X : [\tau]\} \leadsto \Lambda_{\mathrm{P}}\Gamma.\{l_X = e\}}\ \text{B-Val}$$

$$\frac{\Gamma \vdash T : \kappa \leadsto \tau}{\Gamma \vdash \mathrm{type}\, X = T :_{\mathrm{P}} \{l_X : [= \tau : \kappa]\} \leadsto \Lambda_{\mathrm{P}}\Gamma.\{l_X = [\tau : \kappa]\}}\ \text{B-Typ}$$

$$\frac{\Gamma \vdash M :_{\varphi} \exists \overline{t : \kappa}.\Sigma \leadsto e \quad \mathrm{NotAtomic}(\Sigma)}{\Gamma \vdash \mathrm{module}\, X = M :_{\varphi} \exists \overline{t : \kappa}.\{l_X : \Sigma\} \leadsto \mathrm{unpack}\langle \overline{t : \kappa}, x\rangle = e \ \mathrm{in}\ \mathrm{pack}\langle \overline{t}, \Lambda_{\mathrm{P}}\Gamma^{\varphi}.\{l_X = x\ \Gamma^{\varphi}\}\rangle}\ \text{B-Mod}$$

$$\frac{\Gamma \vdash S \leadsto \Xi}{\Gamma \vdash \mathrm{signature}\, X = S :_{\mathrm{P}} \{l_X : [= \Xi]\} \leadsto \Lambda_{\mathrm{P}}\Gamma.\{l_X = [\Xi]\}}\ \text{B-Sig}$$

$$\frac{\Gamma \vdash M :_{\varphi} \exists \overline{t : \kappa}.\{\overline{l_X : \Sigma}\} \leadsto e}{\Gamma \vdash \mathrm{include}\, M :_{\varphi} \exists \overline{t : \kappa}.\{\overline{l_X : \Sigma}\} \leadsto e}\ \text{B-Incl}$$

$$\frac{}{\Gamma \vdash \epsilon :_{\mathrm{P}} \{\} \leadsto \Lambda_{\mathrm{P}}\Gamma.\{\}}\ \text{B-Emt}$$

$$\frac{\begin{array}{ll} \overline{l'_{X_1}} = \overline{l_{X_1}} \setminus \overline{l_{X_2}} \quad \overline{l'_{X_1} : \Sigma'_1} \subseteq \overline{l_{X_1} : \Sigma_1} & \Gamma \vdash B_1 :_{\varphi_1} \exists \overline{t_1 : \kappa_1}.\{\overline{l_{X_1} : \Sigma_1}\} \leadsto e_1 \\ \Sigma = \{\overline{l'_{X_1} : \Sigma'_1}, \overline{l_{X_2} : \Sigma_2}\} & \Gamma, \overline{t_1 : \kappa_1}, \overline{x_{X_1} : \Sigma_1} \vdash B_2 :_{\varphi_2} \exists \overline{t_2 : \kappa_2}.\{\overline{l_{X_2} : \Sigma_2}\} \leadsto e_2 \end{array}}{\begin{array}{l} \Gamma \vdash B_1; B_2 :_{\varphi_1 \vee \varphi_2} \exists \overline{t_1 : \kappa_1}\ \overline{t_2 : \kappa_2}.\Sigma \\ \quad \leadsto \mathrm{unpack}\langle \overline{t_1 : \kappa_1}, x_1\rangle = e_1 \ \mathrm{in} \\ \qquad \mathrm{unpack}\langle \overline{t_2 : \kappa_2}, x_2\rangle = (\mathrm{let}\ \overline{x_{X_1} = \Lambda_{\mathrm{P}}\Gamma^{\varphi_1 \vee \varphi_2}.(x_1\ \Gamma^{\varphi_1})_{\mathrm{P}}.l_{X_1}}\ \mathrm{in}\ e_2)\ \mathrm{in} \\ \qquad \mathrm{pack}\langle \overline{t_1}\ \overline{t_2}, \Lambda_{\mathrm{P}}\Gamma^{\varphi_1 \vee \varphi_2}.\ \mathrm{let}\ \overline{x_{X_1} = (x_1\ \Gamma^{\varphi_1})_{\mathrm{P}}.l_{X_1}}\ \mathrm{in} \\ \qquad\quad \{\overline{l'_{X_1} = (x_1\ \Gamma^{\varphi_1})_{\mathrm{P}}.l'_{X_1}}, \overline{l_{X_2} = (x_2\ (\Gamma, \overline{t_1 : \kappa_1}, \overline{x_{X_1} : \Sigma_1})^{\varphi_2})_{\mathrm{P}}.l_{X_2}}\}\rangle \end{array}}\ \text{B-Seq}$$

Path:

$$\boxed{\Gamma \vdash P : \Sigma \leadsto e}$$

$$\frac{\Gamma \vdash P :_{\varphi} \exists \overline{t : \kappa}.\Sigma \quad \Gamma \vdash \Sigma : \Omega}{\Gamma \vdash P : \Sigma \leadsto \mathrm{unpack}\langle \overline{t : \kappa}, x\rangle = e\ \mathrm{in}\ (x\ \Gamma^{\varphi})_{\mathrm{P}}}\ \text{P-Mod}$$

Expression:

$$\boxed{\Gamma \vdash E : \tau \leadsto e}$$

$$\frac{\Gamma \vdash S \leadsto \Xi \quad \Gamma \vdash \exists \overline{t : \kappa}.\Sigma \leq \mathrm{norm}(\Xi) \leadsto e_1 \quad \Gamma \vdash M :_{\varphi} \exists \overline{t : \kappa}.\Sigma \leadsto e_2}{\Gamma \vdash (\mathrm{pack}\, M : S) : \mathrm{norm}(\Xi) \leadsto e_1\ (\mathrm{unpack}\langle \overline{t : \kappa}, x\rangle = e_2\ \mathrm{in}\ \mathrm{pack}\langle \overline{t : \kappa}, (x\ \Gamma^{\varphi})_{\mathrm{P}}\rangle)}\ \text{E-Unpack}$$

第 5 章

Control Operators

第 6 章

Implicit Parameters and Coherence

第 7 章

Records and Polymorphism

第 8 章

Type Checking and Inference

8.1 Hindley/Milner Type System

[LY98]

8.1.1 Language

$X = \{x, y, z, \ldots\}$, $\mathcal{A} = \{\alpha, \beta, \ldots\}$

E

$$\begin{array}{rcl} e & ::= & () \\ & | & x \\ & | & \lambda x.\, e \\ & | & e\ e \\ & | & \textbf{let}\ x = e\ \textbf{in}\ e \\ & | & \textbf{fix}\ f\ \lambda x.\, e \end{array}$$

T

$$\begin{array}{rcl} \tau & ::= & \textbf{unit} \\ & | & \alpha \\ & | & \tau \to \tau \end{array}$$

Σ

$$\sigma \quad ::= \quad \forall \vec{\alpha}.\, \sigma$$

$\Gamma = \mathcal{A} \xrightarrow{\text{fin}} \Sigma$

8.1.2 Type System

$$\forall \vec{\alpha}.\, \tau_1 \succ \tau_2 \iff \exists S.\, S(\tau_1) = \tau_2 \land \mathrm{dom}(S)$$

$$\mathrm{Gen}(\Gamma, \tau) = \forall \vec{\alpha}.\, \tau \qquad (\vec{\alpha} = \mathit{ftv}(\tau) \backslash \mathit{ftv}(\Gamma))$$

$$\frac{}{\Gamma \vdash () : \textbf{unit}}$$

$$\frac{\Gamma(x) \succ \tau}{\Gamma \vdash x : \tau}$$

$$\frac{\Gamma + x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.\, e : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma + x : \mathrm{Gen}(\Gamma, \tau_1) \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}$$

$$\frac{\Gamma + f : \tau \vdash \lambda x.\, e : \tau}{\Gamma \vdash \textbf{fix}\ f\ \lambda x.\, e : \tau}$$

第 3.1 節

定理 38. $\mathcal{U}(\tau_1, \tau_2) = S$ ならば，$S(\tau_1) = S(\tau_2)$. □

8.1.3 Algorithm W

定理 39. 以下は同値

- $\mathcal{U}(\tau_1, \tau_2) = S$ を満たす S が存在する．
- $S(\tau_1) = S(\tau_2)$ を満たす S が存在する．

□

$$\frac{}{\mathcal{W}(\Gamma, ()) = (\emptyset, \mathbf{unit})}$$

$$\frac{\Gamma(x) = \forall\vec{\alpha}.\,\tau \quad \text{fresh } \vec{\beta}}{\mathcal{W}(\Gamma, x) = (\emptyset, [\vec{\alpha} \leftarrow \vec{\beta}]\tau)}$$

$$\frac{\text{fresh } \beta \quad \mathcal{W}(\Gamma + x : \beta, e) = (S_1, \tau_1)}{\mathcal{W}(\Gamma, \lambda x.\, e) = (S_1, S_1(\beta) \to \tau_1)}$$

$$\frac{\mathcal{W}(\Gamma, e_1) = (S_1, \tau_1) \quad \mathcal{W}(S_1(\Gamma), e_2) = (S_2, \tau_2) \quad \text{fresh } \beta \quad \mathcal{U}(S_2(\tau_1), \tau_2 \to \beta) = S_3}{\mathcal{W}(\Gamma, e_1\ e_2) = (S_3 S_2 S_1, S_3(\beta))}$$

$$\frac{\mathcal{W}(\Gamma, e_1) = (S_1, \tau_1) \quad \Gamma_1 = S_1(\Gamma) \quad \mathcal{W}(\Gamma_1 + x : \mathrm{Gen}(\Gamma_1, \tau_1), e_2) = (S_2, \tau_2)}{\mathcal{W}(\Gamma, \mathbf{let}\ x = e_1\ \mathbf{in}\ e_2) = (S_2 S_1, \tau_2)}$$

$$\frac{\text{fresh } \beta \quad \mathcal{W}(\Gamma + f : \beta, \lambda x.\, e) = (S_1, \tau_1) \quad \mathcal{U}(S_1(\beta), \tau_1) = S_2}{\mathcal{W}(\Gamma, \mathbf{fix}\ f\ \lambda x.\, e) = (S_2 S_1, S_2(\tau_1))}$$

定理 40. 以下は同値

- $\mathcal{W}(\Gamma_0, e) = (S, \tau_0)$，$S(\Gamma_0) = \Gamma$，$S(\tau_0) = \tau$ を満たす S，Γ_0，τ_0 が存在する．
- $\Gamma \vdash e : \tau$.

□

8.1.4 Algorithm M

$$\frac{\mathcal{U}(\rho, \mathbf{unit}) = S}{\mathcal{M}(\Gamma, (), \rho) = S}$$

$$\frac{\mathcal{U}(\rho, [\vec{\beta} \leftarrow \vec{\alpha}]\tau) = S \quad \Gamma(x) = \forall\vec{\alpha}.\,\tau \quad \text{fresh } \vec{\beta}}{\mathcal{M}(\Gamma, x, \rho) = S}$$

$$\frac{\mathcal{U}(\rho, \beta_1 \to \beta_2) = S_1 \quad \text{fresh } \beta_1, \beta_2 \quad \mathcal{M}(S_1(\Gamma) + x : S_1(\beta_1), e, S_1(\beta_2)) = S_2}{\mathcal{M}(\Gamma, \lambda x.\, e, \rho) = S_2 S_1}$$

$$\frac{\mathcal{M}(\Gamma, e_1, \beta \to \rho) = S_1 \quad \text{fresh } \beta \quad \mathcal{M}(S_1(\Gamma), e_2, S_1(\beta)) = S_2}{\mathcal{M}(\Gamma, e_1\ e_2, \rho) = S_2 S_1}$$

$$\frac{\mathcal{M}(\Gamma, e_1, \beta) = S_1 \quad \text{fresh } \beta \quad \mathcal{M}(S_1(\Gamma) + x : \mathrm{Gen}(\Gamma, S_1(\beta)), e_2, S_1(\rho)) = S_2}{\mathcal{M}(\Gamma, \mathbf{let}\ x = e_1\ \mathbf{in}\ e_2, \rho) = S_2 S_1}$$

$$\frac{\mathcal{M}(\Gamma + f : \rho, \lambda x.\, e, \rho) = S}{\mathcal{M}(\Gamma, \mathbf{fix}\ f\ \lambda x.\, e, \rho) = S}$$

定理 41. 以下は同値

- $\mathcal{M}(\Gamma_0, e, \rho) = S$，$S(\Gamma_0) = \Gamma$，$S(\rho) = \tau$ を満たす S，Γ_0，ρ が存在する．
- $\Gamma \vdash e : \tau$.

□

8.1.5 Alternative Type System

$$\frac{}{\Gamma \vdash () : \textbf{unit}}$$

$$\frac{\Gamma(x) = \sigma}{\Gamma \vdash x : \sigma}$$

$$\frac{\Gamma + x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.\, e : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau}$$

$$\frac{\Gamma \vdash e_1 : \sigma_1 \quad \Gamma + x : \sigma_1 \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}$$

$$\frac{\Gamma + f : \tau \vdash \lambda x.\, e : \tau}{\Gamma \vdash \textbf{fix}\ f\ \lambda x.\, e : \tau}$$

$$\frac{\Gamma \vdash e : \tau \quad \vec{\alpha} \notin \mathrm{ftv}(\tau)}{\Gamma \vdash e : \forall \vec{\alpha}.\, \tau}$$

$$\frac{\Gamma \vdash e : \forall \vec{\alpha}.\, \tau}{\Gamma \vdash e : [\vec{\alpha} \leftarrow \vec{\tau'}]\tau}$$

8.2 HM(X): HM Type System with Constraint System

[OSW99]

8.2.1 制約システム

定義 42 (単純制約システム (simple constraint system)). 単純制約システムとは，以下の組 $(\Omega, \Vdash)$ のこと．

- 非空のアルファベット Ω.
- 関係 $(\Vdash) \subseteq \mathcal{P}(\Omega) \times \Omega$ で，以下を満たすもの．
  - 任意の $C \in \mathcal{P}(\Omega)$，$P \in C$ について，$C \Vdash P$.
  - 任意の $C, D \in \mathcal{P}(\Omega)$，$Q \in \Omega$ について，$(\forall P \in D.\, C \Vdash P)$ かつ $D \Vdash Q$ ならば $C \Vdash Q$.

この時，$C \in \mathcal{P}(\Omega)$ を制約 (*constraint*) と呼ぶ．また，$(\Vdash) \subseteq (\mathcal{P}(\Omega))^2$ への拡張を，$C \Vdash D \overset{\text{def}}{\iff} \forall P \in D.\, C \Vdash P$ と定義する．$C \Vdash D$ かつ $D \Vdash C$ の時，$C \dashVdash D$ と表記する．さらに，$C \wedge D = C \cup D$ と表記する． □

命題 43. 単純制約システム $(\Omega, \Vdash)$ は，以下を admissible にする．

$$\frac{}{C \Vdash C}$$

$$\frac{C_1 \Vdash C_2 \quad C_2 \Vdash C_3}{C_1 \Vdash C_3}$$

$$\frac{C \Vdash D}{C \wedge C' \Vdash D}$$

□

証明.

$$\begin{aligned}
C \Vdash C &\iff \forall P \in C.\, C \Vdash P \\
C_1 \Vdash C_2 \wedge C_2 \Vdash C_3 &\implies \forall Q \in C_3.\, C_1 \Vdash C_2 \wedge C_2 \Vdash Q \\
&\implies \forall Q \in C_3.\, (\forall P \in C_2.\, C_1 \Vdash P) \wedge C_2 \Vdash Q \\
&\implies \forall Q \in C_3.\, C_1 \Vdash Q && (\because \text{単純制約システムの公理}) \\
&\implies C_1 \Vdash C_3 \\
\forall P \in C \wedge C'.\, C \in P &\implies C \wedge C' \Vdash C \\
C \Vdash D &\implies C \wedge C' \Vdash C \wedge C \Vdash D \implies C \wedge C' \Vdash D
\end{aligned}$$

より明らか． ∎

定義 44 (Cylindric 制約システム (cylindric constraint system)). Cylindric 制約システムとは，以下の組 $(\Omega, \Vdash, \mathcal{A}, \exists)$ のこと．

- 単純制約システム $(\Omega, \Vdash)$.
- 変数の無限集合 $\mathcal{A}$.
- 関数の族 $\{\exists\alpha\}_{\alpha \in \mathcal{A}} \in \prod_{\alpha \in \mathcal{A}} \mathcal{P}(\Omega) \to \mathcal{P}(\Omega)$ で以下を満たすもの．
  - 任意の $C \in \mathcal{P}(\Omega)$，$\alpha \in \mathcal{A}$ について，$C \Vdash \exists\alpha.\, C$.
  - 任意の $C, D \in \mathcal{P}(\Omega)$，$\alpha \in \mathcal{A}$ について，$C \Vdash D$ ならば，$\exists\alpha.\, C \Vdash \exists\alpha.\, D$.
  - 任意の $C, D \in \mathcal{P}(\Omega)$，$\alpha \in \mathcal{A}$ について，$\exists\alpha.\, (C \wedge \exists\alpha.\, C) \dashVdash (\exists\alpha.\, C) \wedge (\exists\alpha.\, D)$.
  - 任意の $C \in \mathcal{P}(\Omega)$，$\alpha, \beta \in \mathcal{A}$ について，$\exists\alpha.\, \exists\beta.\, C \dashVdash \exists\beta.\, \exists\alpha.\, C$.

  ただし，$\exists\alpha.\, C = (\exists\alpha)(C)$ である．

□

定義 45 (自由変数). Cylindric 制約システム $(\Omega, \Vdash, \mathcal{A}, \exists)$，制約 $C \in \mathcal{P}(\Omega)$ について，自由変数の集合を $\mathrm{fv}(C) = \{\alpha \mid \exists\alpha.\, C \not\dashVdash C\}$ とおく． □

定義 46 (充足可能 (satisfiable)). Cylindric 制約システム $(\Omega, \Vdash, \mathcal{A}, \exists)$，制約 $C \in \mathcal{P}(\Omega)$ について，$\Vdash \exists \mathrm{fv}(C).\, C$ の時，C は充足可能であるという． □

補題 47. Cylindric 制約システム $(\Omega, \Vdash, \mathcal{A}, \exists)$，制約 $C \in \mathcal{P}(\Omega)$ について，以下は同値．

- C は充足可能．
- $\exists \alpha.\, C$ は充足可能．

□

定義 48 (項制約システム (term constraint system)). 項制約システムとは，

- 項代数 (Σ, X)．
- 述語のランク付きアルファベット P．
- Cylindric 制約システム $(\Omega, \Vdash, X, \exists)$，ただし，$\Omega = \{p(\tau_1, \dots, \tau_n) \mid p^{(n)} \in P, \tau_1, \dots, \tau_n \in [\![(\Sigma, X)]\!]\}$．

の組 $(\Sigma, P, \Omega, \Vdash, X, \exists)$ で，以下を満たすもの．

- 任意の $\alpha \in X$ について，$\Vdash \alpha = \alpha$．
- 任意の $\alpha_1, \alpha_2 \in X$ について，$(\alpha_1 = \alpha_2) \Vdash (\alpha_2 = \alpha_1)$．
- 任意の $\alpha_1, \alpha_2, \alpha_3 \in X$ について，$(\alpha_1 = \alpha_2) \wedge (\alpha_2 = \alpha_3) \Vdash (\alpha_1 = \alpha_3)$．
- 任意の $\alpha_1, \alpha_2 \in X$，$C \in \mathcal{P}(\Omega)$ について，$(\alpha_1 = \alpha_2) \wedge \exists \alpha_1.\, (C \wedge (\alpha_1 = \alpha_2)) \Vdash C$．
- 任意のコンテキスト $T[] \in \mathcal{C}(\mathcal{T})$，$\tau_1, \tau_2 \in [\![(\Sigma, X)]\!]$ について，$(\tau_1 = \tau_2) \Vdash (T[\tau_1] = T[\tau_2])$．
- 任意の $P \in \Omega$，$\tau \in [\![(\Sigma, X)]\!]$，$\alpha \in X$，$\alpha \notin fv(\tau)$ について，$P[\alpha \leftarrow \tau] \dashv\Vdash \exists \alpha.\, (P \wedge (\alpha = \tau))$．

□

定義 49 (置換の拡張). $(P_1 \wedge \cdots \wedge P_n)[\vec{\alpha} \leftarrow \vec{\tau}] = P_1[\vec{\alpha} \leftarrow \vec{\tau}] \wedge \cdots \wedge P_n[\vec{\alpha} \leftarrow \vec{\tau}]$ と表記する． □

補題 50 (改名 (renaming)). 項制約システム $(\Sigma, P, \Omega, \Vdash, X, \exists)$，$C \in \mathcal{P}(\Omega)$，$\alpha_1, \alpha_2 \in X$ について，α_2 が C に出現しない時，$\exists \alpha_1.\, C \dashv\Vdash \exists \alpha_2.\, C[\alpha_1 \leftarrow \alpha_2]$． □

補題 51 (正規形 (normal form)). 項制約システム $(\Sigma, P, \Omega, \Vdash, X, \exists)$，$C \in \mathcal{P}(\Omega)$ について，以下が成り立つ．

$$C[\alpha_1 \leftarrow \tau_1, \dots, \alpha_n \leftarrow \tau_n] \dashv\Vdash \exists \alpha_1, \dots, \alpha_n.\, C \wedge (\alpha_1 = \tau_1) \wedge \cdots \wedge (\alpha_n = \tau_n)$$

□

補題 52 (置換 (substitution)). 項制約システム $(\Sigma, P, \Omega, \Vdash, X, \exists)$，$C, D \in \mathcal{P}(\Omega)$，置換 ϕ について，以下が成り立つ．

$$C \Vdash D \implies \phi C \Vdash \phi D$$

□

8.2.2 型システム

定義 53 (包含 (subsumption)). 項制約システム $(\Sigma, P, \Omega, \Vdash, X, \exists)$ について，包含付きであるとは，$\precsim \in P^{(2)}$ で以下を満たすことを言う．

$$\frac{}{(\alpha_1 = \alpha_2) \Vdash (\alpha_1 \precsim \alpha_2) \wedge (\alpha_2 \precsim \alpha_1)}$$

$$\frac{}{(\alpha_1 \precsim \alpha_2) \wedge (\alpha_2 \precsim \alpha_1) \Vdash (\alpha_1 = \alpha_2)}$$

$$\frac{D \Vdash (\alpha_1 \precsim \alpha_2) \quad D \Vdash (\alpha_2 \precsim \alpha_3)}{D \Vdash (\alpha_1 \precsim \alpha_3)}$$

$$\frac{D \Vdash (\alpha_1 \precsim \alpha_2) \quad D \Vdash (\beta_1 \precsim \beta_2)}{D \Vdash (\alpha_1 \to \beta_1 \precsim \alpha_2 \to \beta_2)}$$

□

定義 54 (型システム). 包含付き項制約システム $(\Sigma, P, \Omega, \Vdash, X, \exists)$ について，制約 $C \in \mathcal{P}(\Omega)$，環境 Γ，式 e，型スキーム σ の型判定 $C, \Gamma \vdash e : \sigma$ を以下のように定義する．

$$\frac{x : \sigma \in \Gamma}{C, \Gamma \vdash x : \sigma}$$

$$\frac{C, \Gamma \vdash e : \tau_1 \quad C \Vdash \tau_1 \lesssim \tau_2}{C, \Gamma \vdash e : \tau_2}$$

$$\frac{C, \Gamma + x : \tau_1 \vdash e : \tau_2}{C, \Gamma \vdash \lambda x.\, e : \tau_1 \to \tau_2}$$

$$\frac{C, \Gamma \vdash e_1 : \tau_1 \to \tau_2 \quad C, \Gamma \vdash e_2 : \tau_1}{C, \Gamma \vdash e_1\, e_2 : \tau_2}$$

$$\frac{C, \Gamma \vdash e_1 : \sigma_1 \quad C, \Gamma + x : \sigma_1 \vdash e_2 : \tau_2}{C, \Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}$$

$$\frac{C \wedge D, \Gamma \vdash e : \tau \quad \vec{\alpha} \notin fv(C) \cup fv(\Gamma)}{C \wedge \exists \vec{\alpha}.\, D, \Gamma \vdash e : \forall \vec{\alpha}.\, D \Rightarrow \tau}$$

$$\frac{C, \Gamma \vdash e : \forall \vec{\alpha}.\, D \Rightarrow \tau' \quad C \Vdash D[\vec{\alpha} \leftarrow \vec{\tau}]}{C, \Gamma \vdash e : \tau'[\vec{\alpha} \leftarrow \vec{\tau}]}$$

□

8.2.3 推論アルゴリズム

定義 55. 変数の集合 U，置換 ϕ，$x \in U$ について，$\phi|_U$ を以下のようにおく．

$$\phi|_U(x) = \begin{cases} \sigma & (x : \sigma \in \phi) \\ x & (\text{otherwise}) \end{cases}$$

また，

$$\begin{aligned} \Vdash \psi =_U \phi &\overset{\text{def}}{\iff} \forall x \in U.\, \Vdash \psi|_U(x) = \phi|_U(x) \\ \Vdash \psi \leq_U^{\chi} \phi &\overset{\text{def}}{\iff} \Vdash \chi \circ \psi =_U \phi \\ \Vdash \psi \leq_U \phi &\overset{\text{def}}{\iff} \exists \chi.\, \Vdash \psi \leq_U^{\chi} \phi \end{aligned}$$

と表記する． □

定義 56 (正規形). 項制約システム $(\Sigma, P, \Omega, \Vdash, X, \exists)$，制約 $C, D \in \mathcal{P}(\Omega)$，置換 ϕ, ψ について，(C, ψ) が (D, ϕ) の正規形とは，$\phi \leq \psi$，$C \Vdash \psi D$，$\psi C = C$ を満たすことを言う． □

定義 57 (制約付き Algorithm W). 項制約システム $(\Sigma, P, \Omega, \Vdash, X, \exists)$ について，$norm$ を制約 $C \in \mathcal{P}(\Omega)$，置換 ψ において $norm(C, \psi) = (D, \phi)$ が (C, ψ) の正規形になる関数とする．また，gen を制約 $C \in \mathcal{P}(\Omega)$，環境 Γ，型スキーム σ，変数列 $\vec{\alpha} = (fv(\sigma) \cup fv(C)) \backslash fv(\Gamma)$，$C \dashv\Vdash C' \wedge D$，$fv(D) \wedge \vec{\alpha} = \emptyset$ を満たす制約 $C', D \in \mathcal{P}(\Omega)$ について，

$$gen(C, \Gamma, \sigma) = (D \wedge \exists \vec{\alpha}.\, C', \forall \vec{\alpha}.\, C' \Rightarrow \sigma)$$

を満たす関数とする．この時，置換 ψ，$C \in \mathcal{P}(\Omega)$，環境 Γ，式 e，型スキーム σ について，判定 $\psi, C, \Gamma \vdash^W e : \sigma$ を以下のように定義する．

$$\frac{x : \forall \vec{\alpha}.\, D \Rightarrow \tau \in \Gamma \quad \text{fresh}\ \vec{\beta} \quad norm(D, [\vec{\alpha} \leftarrow \vec{\beta}]) = (C, \psi)}{\psi|_{fv(\Gamma)}, C, \Gamma \vdash^W x : \psi\tau}$$

$$\frac{\psi, C, \Gamma + x : \alpha \vdash^W e : \tau \quad \text{fresh}\ \alpha}{\psi|_{\{\alpha\}}, C, \Gamma \vdash^W \lambda x.\, e : \psi(\alpha) \to \tau}$$

$$\frac{\psi_1, C_1, \Gamma \vdash^W e_1 : \tau_1 \quad \psi_2, C_2, \Gamma \vdash^W e_2 : \tau_2 \quad D = C_1 \wedge C_2 \wedge \tau_1 \precsim \tau_2 \to \alpha \quad \text{fresh } \alpha \quad norm(D, \psi_1 \sqcup \psi_2) = (C, \psi)}{\psi|_{fv(\Gamma)}, C, \Gamma \vdash^W e_1\ e_2 : \psi(\alpha)}$$

$$\frac{\psi_1, C_1, \Gamma \vdash^W e_1 : \tau_1 \quad (C_2, \sigma) = gen(C_1, \psi_1\Gamma, \tau_1) \quad \psi_2, C_3, \Gamma + x : \sigma \vdash^W e_2 : \tau_2 \quad norm(C_2 \wedge C_3, \psi_1 \sqcup \psi_2) = (C, \psi)}{\psi|_{fv(\Gamma)}, C, \Gamma \vdash^W \textbf{let } x = e_1 \textbf{ in } e_2 : \psi\tau_2}$$

□

8.2.4 自由構成

構文:

$$
\begin{array}{rcl}
T & ::= & \to \mid \cdots \\
D & ::= & \simeq \mid \precsim \mid \cdots \\
Q & ::= & \epsilon \\
 & \mid & Q_1 \wedge Q_2 \\
 & \mid & D\vec{\tau} \\
C & ::= & Q \\
\tau & ::= & \alpha \\
 & \mid & T\vec{\tau} \\
\sigma & ::= & \forall\vec{\alpha}.\, Q \Rightarrow \tau \\
e & ::= & x \\
 & \mid & \lambda x.\, e \\
 & \mid & e_1\ e_2 \\
 & \mid & \textbf{let } x = e_1 \textbf{ in } e_2
\end{array}
$$

制約推論:

$$\frac{}{(D\vec{\tau}) \in (D\vec{\tau})} \quad \frac{(D\vec{\tau}) \in C_1}{(D\vec{\tau}) \in C_1 \wedge C_2} \quad \frac{(D\vec{\tau}) \in C_2}{(D\vec{\tau}) \in C_1 \wedge C_2}$$

$$\frac{(D\vec{\tau}) \in C}{C \Vdash D\vec{\tau}} \quad \frac{C \Vdash Q_1 \quad C \Vdash Q_2}{C \Vdash Q_1 \wedge Q_2}$$

$$\frac{}{C \Vdash \tau \simeq \tau} \quad \frac{C \Vdash \tau_2 \simeq \tau_1}{C \Vdash \tau_1 \simeq \tau_2} \quad \frac{C \Vdash \tau_1 \simeq \tau_2 \quad C \Vdash \tau_2 \simeq \tau_3}{C \Vdash \tau_1 \simeq \tau_3}$$

$$\frac{C \Vdash \tau_1 \simeq \tau_2}{C \Vdash \tau_1 \precsim \tau_2}$$

$$\frac{\overrightarrow{C \Vdash \tau_1 \simeq \tau_2}}{C \Vdash T\vec{\tau_1} \simeq T\vec{\tau_2}} \quad \frac{C \Vdash T\vec{\tau_1} \simeq T\vec{\tau_2}}{C \Vdash \tau_1 \simeq \tau_2}$$

$$\frac{C \Vdash D\vec{\tau_1} \quad \overrightarrow{C \Vdash \tau_1 \simeq \tau_2}}{C \Vdash D\vec{\tau_2}}$$

制約解決:

$$\frac{\begin{array}{c} C \vdash \text{flat}(Q_1) \to^* W_2 \not\to \\ \theta_2 = [\alpha_2 \leftarrow \theta_2\tau_2 \mid (\alpha_2 \simeq \tau_2) \in W_2] \quad Q_2 = \bigwedge\{D\vec{\tau} \mid (D\vec{\tau}) \in W_2\} \\ \vec{\alpha_3} = ftv(Q_2) \quad \theta_3 = [\overline{\alpha_3 \leftarrow \tau_3}] \quad C \Vdash \theta_3\theta_2 Q_2 \end{array}}{\text{solv}(C, Q_1) = \theta_3\theta_2}$$

$$\frac{}{\text{flat}(\epsilon) = \emptyset}$$

$$\frac{\text{flat}(Q_1) = W_1 \quad \text{flat}(Q_2) = W_2}{\text{flat}(Q_1 \wedge Q_2) = W_1 \cup W_2}$$

$$\frac{}{\text{flat}(D\vec{\tau}) = \{D\vec{\tau}\}}$$

$$\frac{\alpha \leq \beta (\text{lexicographically})}{\alpha \prec \beta}$$

$$\frac{}{\alpha \prec T\vec{\tau}}$$

$$\frac{\alpha \prec \tau \quad \alpha \notin \mathit{ftv}(\tau)}{\alpha \sim \tau}$$

$$\frac{Q = (\tau \simeq \tau) \in W}{C \vdash W \to W \backslash \{Q\}}$$

$$\frac{Q = (T\vec{\tau_1} \simeq T\vec{\tau_2}) \in W}{C \vdash W \to (W \backslash \{Q\}) \cup \overrightarrow{\tau_1 \simeq \tau_2}}$$

$$\frac{(T\vec{\tau_1} \simeq S\vec{\tau_2}) \in W \quad T \neq S}{C \vdash W \to \bot}$$

$$\frac{(\beta \simeq \tau) \in W \quad \beta \in \mathit{ftv}(\tau)}{C \vdash W \to \bot}$$

$$\frac{Q = (\tau_1 \simeq \tau_2) \in W \quad \tau_2 \prec \tau_1}{C \vdash W \to (W \backslash \{Q\}) \cup \{\tau_2 \simeq \tau_1\}}$$

$$\frac{\{\beta \simeq \tau_1, \beta \simeq \tau_2\} \subseteq W \quad \tau_1 \neq \tau_2 \quad \beta \sim \tau_1 \quad \beta \sim \tau_2}{C \vdash W \to (W \backslash \{\beta \simeq \tau_2\}) \cup \{\tau_1 \simeq \tau_2\}}$$

$$\frac{\{\beta_1 \simeq \tau_1, \beta_2 \simeq \tau_2\} \subseteq W \quad \beta_1 \in \mathit{ftv}(\tau_2) \quad \beta_1 \sim \tau_1 \quad \beta_2 \sim \tau_2}{C \vdash W \to (W \backslash \{\beta_2 \simeq \tau_2\}) \cup \{\beta_2 \simeq \tau_2[\beta_1 \leftarrow \tau_1]\}}$$

$$\frac{\{\beta_1 \simeq \tau_1, D\vec{\tau_2}\} \subseteq W \quad \beta_1 \in \mathit{ftv}(\vec{\tau_2}) \quad \beta_1 \sim \tau_1}{C \vdash W \to (W \backslash \{D\vec{\tau_2}\}) \cup \{(D\vec{\tau_2})[\beta_1 \leftarrow \tau_1]\}}$$

$$\frac{Q = D\vec{\tau} \in W \quad D\vec{\tau} \in C}{C \vdash W \to W \backslash \{Q\}}$$

補題 58. $C \vdash \mathrm{flat}(Q_1) \to^* W_2 \not\to$ の時，$W_3 = \{\tau_1 \simeq \tau_2 \mid \tau_1 \simeq \tau_2 \mid W_2\}$，$W_4 = W_2 \backslash W_3$ とすると，以下が成り立つ:

- $\tau_1 \simeq \tau_2 \in W_3$ について，$\tau_1 = \alpha$.
- $\alpha_1 \simeq \tau_2 \in W_3$ について，$\alpha_1 \notin \mathit{ftv}(\tau_2)$.
- $\alpha_1 \simeq \tau_2 \in W_3$ について，$\tau_2 = \alpha_2$ ならば，$\alpha_1 \lnsim \alpha_2$.
- $\alpha \simeq \tau_1, \alpha \simeq \tau_2 \in W_3$ について，$\tau_1 = \tau_2$.
- $Q \in W_2$，$\alpha \in \mathit{ftv}(Q)$ について，$\alpha \simeq \tau \in W_3$ となる τ は存在しない.
- $D\vec{\tau} \in W_4$ について，$D\vec{\tau} \notin C$.

□

8.3 OutsideIn(X): Modular Type Inference with Local Assumptions

[VJSS11]

8.3.1 Syntax

x, y, z, f, g, h 変数

α, β, γ 型変数

K コンストラクタ

T 型コンストラクタ

D 制約コンストラクタ

F 型関数

$$
\begin{array}{rcl}
P & ::= & \epsilon \\
& | & f = e, P \\
& | & f : \sigma = e, P \\
\nu & ::= & x \mid K \\
e & ::= & \nu \\
& | & \lambda x.\, e \\
& | & e_1\ e_2 \\
& | & \mathbf{case}(e, \overrightarrow{K\vec{x} \mapsto e}) \\
& | & \mathbf{let}(x : \sigma = e_1, e_2) \\
& | & \mathbf{let}(x = e_1, e_2) \\
\sigma & ::= & \forall \vec{\alpha}.\, Q \Rightarrow \tau \\
P & ::= & \tau_1 \simeq \tau_2 \\
& | & D\vec{\tau} \\
Q & ::= & \epsilon \\
& | & Q_1 \wedge Q_2 \\
& | & P \\
\mathcal{T} & ::= & \alpha \\
& | & \rightarrow \\
& | & T \\
\tau, \upsilon & ::= & \alpha \\
& | & \tau_1 \rightarrow \tau_2 \\
& | & T\vec{\tau} \\
& | & F\vec{\tau} \\
\Gamma & ::= & \epsilon \\
& | & \nu : \sigma, \Gamma \\
\mathcal{Q} & ::= & Q \\
& | & \mathcal{Q} \wedge \mathcal{Q} \\
& | & \forall \vec{\alpha}.\, Q \Rightarrow Q \\
& | & \forall \vec{\alpha}.\, F\vec{\tau_1} \simeq \tau_2
\end{array}
$$

8.3.2 Entailment

Concrete:

$$
\frac{\mathcal{Q} \Vdash Q_1 \quad \mathcal{Q} \Vdash Q_2}{\mathcal{Q} \Vdash Q_1 \wedge Q_2}
$$

$$
\frac{}{\mathcal{Q} \Vdash \tau \simeq \tau} \quad \frac{\mathcal{Q} \Vdash \tau_2 \simeq \tau_1}{\mathcal{Q} \Vdash \tau_1 \simeq \tau_2} \quad \frac{\mathcal{Q} \Vdash \tau_1 \simeq \tau_2 \quad \mathcal{Q} \Vdash \tau_2 \simeq \tau_3}{\mathcal{Q} \Vdash \tau_1 \simeq \tau_3}
$$

$$
\frac{\mathcal{Q} \Vdash T\vec{\tau_1} \simeq T\vec{\tau_2}}{\mathcal{Q} \Vdash \bigwedge \overrightarrow{\tau_1 \simeq \tau_2}} \quad \frac{\mathcal{Q} \Vdash \bigwedge \overrightarrow{\tau_1 \simeq \tau_2}}{\mathcal{Q} \Vdash T\vec{\tau_1} \simeq T\vec{\tau_2}} \quad \frac{\mathcal{Q} \Vdash \bigwedge \overrightarrow{\tau_1 \simeq \tau_2}}{\mathcal{Q} \Vdash F\vec{\tau_1} \simeq F\vec{\tau_2}}
$$

$$\frac{(\forall\vec{\alpha}.\,Q_1 \Rightarrow Q_2) \in \mathcal{Q} \quad \mathcal{Q} \Vdash Q_1[\overrightarrow{\alpha \leftarrow \tau}]}{\mathcal{Q} \Vdash Q_2[\overrightarrow{\alpha \leftarrow \tau}]}$$

$$\frac{\mathcal{Q} \Vdash D\vec{\tau_1} \quad \mathcal{Q} \Vdash \bigwedge \overline{\tau_1 \simeq \tau_2}}{\mathcal{Q} \Vdash D\vec{\tau_2}}$$

- projection って必要ないん？

Requirements:

$$\frac{}{\mathcal{Q} \wedge Q \Vdash Q} \quad \frac{\mathcal{Q} \wedge Q_1 \Vdash Q_2 \quad \mathcal{Q} \wedge Q_2 \Vdash Q_3}{\mathcal{Q} \wedge Q_1 \Vdash Q_3} \quad \frac{\mathcal{Q} \Vdash Q}{\mathcal{Q} \Vdash Q[\alpha \leftarrow \tau]}$$

$$\frac{}{\mathcal{Q} \Vdash \tau \simeq \tau} \quad \frac{\mathcal{Q} \Vdash \tau_2 \simeq \tau_1}{\mathcal{Q} \Vdash \tau_1 \simeq \tau_2} \quad \frac{\mathcal{Q} \Vdash \tau_1 \simeq \tau_2 \quad \mathcal{Q} \Vdash \tau_2 \simeq \tau_3}{\mathcal{Q} \Vdash \tau_1 \simeq \tau_3}$$

$$\frac{\mathcal{Q} \Vdash Q_1 \quad \mathcal{Q} \Vdash Q_2}{\mathcal{Q} \Vdash Q_1 \wedge Q_2}$$

$$\frac{\mathcal{Q} \Vdash \tau_1 \simeq \tau_2}{\mathcal{Q} \Vdash \tau[\alpha \leftarrow \tau_1] \simeq \tau[\alpha \leftarrow \tau_2]}$$

8.3.3 Type System

$$\frac{(\nu : \forall\vec{\alpha}.\,Q_1 \Rightarrow \tau_1) \in \Gamma \quad Q \Vdash Q_1[\overrightarrow{\alpha \leftarrow \tau_2}]}{Q;\Gamma \vdash \nu : \tau_1[\overrightarrow{\alpha \leftarrow \tau_2}]}$$

$$\frac{Q;\Gamma \vdash e : \tau_1 \quad Q \Vdash \tau_1 \simeq \tau_2}{Q;\Gamma \vdash e : \tau_2}$$

$$\frac{Q;\Gamma, x : \tau_1 \vdash e : \tau_2}{Q;\Gamma \vdash \lambda x.\,e : \tau_1 \to \tau_2}$$

$$\frac{Q;\Gamma \vdash e_1 : \tau_1 \to \tau_2 \quad Q;\Gamma \vdash e_2 : \tau_1}{Q;\Gamma \vdash e_1\ e_2 : \tau_2}$$

$$\frac{Q;\Gamma \vdash e_1 : \tau_1 \quad Q;\Gamma, x : \tau_1 \vdash e_2 : \tau_2}{Q;\Gamma \vdash \mathbf{let}(x = e_1, e_2) : \tau_2}$$

$$\frac{Q \wedge Q_1;\Gamma \vdash e_1 : \tau_1 \quad \vec{\alpha} \wedge (\mathit{ftv}(Q) \cup \mathit{ftv}(\Gamma)) = \varnothing \quad Q;\Gamma, x : \forall\vec{\alpha}.\,Q_1 \Rightarrow \tau_1 \vdash e_2 : \tau_2}{Q;\Gamma \vdash \mathbf{let}(x : \forall\vec{\alpha}.\,Q_1 \Rightarrow \tau_1 = e_1, e_2) : \tau_2}$$

$$\frac{\begin{array}{c} Q;\Gamma \vdash e : T\vec{\tau_1} \\ \bigwedge_i (K_i : \forall\vec{\alpha_i}\vec{\beta}.\,Q_i \Rightarrow \vec{\upsilon_i} \to T\vec{\alpha_i}) \in \Gamma \\ \vec{\beta} \wedge (\mathit{ftv}(Q) \cup \mathit{ftv}(\Gamma) \cup \overrightarrow{\mathit{ftv}(\tau_1)} \cup \mathit{ftv}(\tau_2)) = \varnothing \\ \bigwedge_i Q \wedge Q_i[\overrightarrow{\alpha_i \leftarrow \tau}];\Gamma, \overrightarrow{x_i : \upsilon_i[\alpha_i \leftarrow \tau]} \vdash e_i : \tau_2 \end{array}}{Q;\Gamma \vdash \mathbf{case}(e, \overrightarrow{K_i\vec{x_i} \mapsto e_i}) : \tau_2}$$

$$\frac{(\mathit{ftv}(\Gamma) \cup \mathit{ftv}(\mathcal{Q})) = \varnothing}{\mathcal{Q};\Gamma \vdash \epsilon}$$

$$\frac{\mathcal{Q} \wedge Q_1 \Vdash Q_2 \quad Q_2;\Gamma \vdash e : \tau \quad \vec{\alpha} = \mathit{ftv}(Q_1) \cup \mathit{ftv}(\tau) \quad \mathcal{Q};\Gamma, (f : \forall\vec{\alpha}.\,Q_1 \Rightarrow \tau) \vdash P}{\mathcal{Q};\Gamma \vdash f = e, P}$$

$$\frac{\mathcal{Q} \wedge Q_1 \Vdash Q_2 \quad Q_2;\Gamma \vdash e : \tau \quad \vec{\alpha} = \mathit{ftv}(Q_1) \cup \mathit{ftv}(\tau) \quad \mathcal{Q};\Gamma, (f : \forall\vec{\alpha}.\,Q_1 \Rightarrow \tau) \vdash P}{\mathcal{Q};\Gamma \vdash f : \forall\vec{\alpha}.\,Q_1 \Rightarrow \tau = e, P}$$

8.3.4 Type Inference

$$\begin{array}{rcl} C & ::= & Q \\ & \mid & C_1 \land C_2 \\ & \mid & \exists \vec{\alpha}.\,(Q \supset C) \end{array}$$

$$\frac{\text{fresh}\,\vec{\beta} \quad (\nu : \forall\vec{\alpha}.\,Q \Rightarrow \tau) \in \Gamma}{\Gamma \rhd \nu \rightsquigarrow Q[\overrightarrow{\alpha \leftarrow \beta}] \Rightarrow \tau[\overrightarrow{\alpha \leftarrow \beta}]}$$

$$\frac{\text{fresh}\,\beta \quad \Gamma, x : \beta \rhd e \rightsquigarrow C \Rightarrow \tau}{\Gamma \rhd \lambda x.\,e \rightsquigarrow C \Rightarrow \beta \to \tau}$$

$$\frac{\Gamma \rhd e_1 \rightsquigarrow C_1 \Rightarrow \tau_1 \quad \Gamma \rhd e_2 \rightsquigarrow C_2 \Rightarrow \tau_2 \quad \text{fresh}\,\beta}{\Gamma \rhd e_1\;e_2 \rightsquigarrow C_1 \land C_2 \land (\tau_1 \simeq (\tau_2 \to \beta)) \Rightarrow \beta}$$

$$\frac{\Gamma \rhd e_1 \rightsquigarrow C_1 \Rightarrow \tau_1 \quad \Gamma, x : \tau_1 \rhd e_2 \rightsquigarrow C_2 \Rightarrow \tau_2}{\Gamma \rhd \textbf{let}(x = e_1, e_2) \rightsquigarrow C_1 \land C_2 \Rightarrow \tau_2}$$

$$\frac{\begin{array}{c}\Gamma \rhd e_1 \rightsquigarrow C_1' \Rightarrow \tau_1' \\ \vec{\beta_1} = (\text{ftv}(\tau_1') \cup \text{ftv}(C_1')) \backslash \text{ftv}(\Gamma) \\ \Gamma, x : \forall\vec{\alpha_1}.\,Q_1 \Rightarrow \tau_1 \rhd e_2 \rightsquigarrow C_2 \Rightarrow \tau_2\end{array}}{\Gamma \rhd \textbf{let}(x : \forall\vec{\alpha_1}.\,Q_1 \Rightarrow \tau_1 = e_1, e_2) \rightsquigarrow (\exists\vec{\beta_1}.\,Q_1 \supset C_1' \land \tau_1 \simeq \tau_1') \land C_2 \Rightarrow \tau_2}$$

$$\frac{\begin{array}{c}\Gamma \rhd e \rightsquigarrow C \Rightarrow \tau \\ \text{fresh}\,\beta, \vec{\gamma} \\ \bigwedge_i \text{fresh}\,\vec{\beta_i} \\ \bigwedge_i (K_i : \forall\vec{\alpha_i}\vec{\beta_i}.\,Q_i \Rightarrow \vec{\upsilon_i} \to T\vec{\alpha_i}) \in \Gamma \\ \bigwedge_i \Gamma, (\overrightarrow{x_i : \upsilon_i[\alpha_i \leftarrow \gamma]}) \rhd e_i \rightsquigarrow C_i \Rightarrow \tau_i \\ \bigwedge_i \delta_i = (\text{ftv}(\tau_i) \cup \text{ftv}(C_i)) \backslash (\text{ftv}(\Gamma) \cup \bigcup \overrightarrow{\text{ftv}(\gamma)})\end{array}}{\Gamma \rhd \textbf{case}(e, \overrightarrow{K_i\vec{x_i} \mapsto e_i}) \rightsquigarrow C \land (\tau \simeq T\vec{\gamma}) \land (\bigwedge_i \exists\delta_i.\,Q_i[\overrightarrow{\alpha \leftarrow \gamma}] \supset C_i \land \tau_i \simeq \beta) \Rightarrow \beta}$$

制約解決 $\mathcal{Q}; Q; \vec{\alpha} \vdash C_1 \overset{\text{solv}}{\rightsquigarrow} Q_2 \mid \theta$ については，後述する．

$$\frac{}{\mathcal{Q}; \Gamma \rhd \epsilon \rightsquigarrow \top}$$

$$\frac{\begin{array}{c}\Gamma \rhd e \rightsquigarrow C \Rightarrow \tau \\ \mathcal{Q}; \epsilon; \text{ftv}(\tau) \cup \text{ftv}(C) \vdash C \overset{solv}{\rightsquigarrow} Q \mid \theta \\ \vec{\alpha} = \text{ftv}(\theta\tau) \cup \text{ftv}(Q) \\ \text{fresh}\,\vec{\beta} \\ \mathcal{Q}; \Gamma, f : \forall\vec{\beta}.\,(Q \Rightarrow \theta\tau)[\overrightarrow{\alpha \leftarrow \beta}] \rhd P \rightsquigarrow \top\end{array}}{\mathcal{Q}; \Gamma \rhd f = e, P \rightsquigarrow \top}$$

$$\frac{\begin{array}{c}\Gamma \rhd e \rightsquigarrow C' \Rightarrow \tau' \\ \mathcal{Q}; Q; \text{ftv}(\tau') \cup \text{ftv}(C') \vdash C' \land (\tau \simeq \tau') \overset{solv}{\rightsquigarrow} \epsilon \mid \theta \\ \mathcal{Q}; \Gamma, f : \forall\vec{\alpha}.\,Q \Rightarrow \tau \rhd P \rightsquigarrow \top\end{array}}{\mathcal{Q}; \Gamma \rhd f : \forall\vec{\alpha}.\,Q \Rightarrow \tau = e, P \rightsquigarrow \top}$$

8.3.5 Constraint Solving

$$\frac{}{\text{split}(Q) = \langle Q, \varnothing \rangle}$$

$$\frac{\text{split}(C_1) = \langle Q_1, I_1 \rangle \quad \text{split}(C_2) = \langle Q_2, I_2 \rangle}{\text{split}(C_1 \land C_2) = \langle Q_1 \land Q_2, I_1 \cup I_2 \rangle}$$

$$\frac{}{\mathrm{split}(\exists\vec{\alpha}.\,Q \supset C) = \langle \epsilon, \{\exists\vec{\alpha}.\,Q \supset C\}\rangle}$$

$$\frac{\mathrm{split}(C_1) = \langle Q_1, I_1\rangle \quad \mathcal{Q}; Q; \vec{\alpha} \vdash Q_1 \overset{\mathrm{simpl}}{\rightsquigarrow} Q_2 \mid \theta \quad \bigwedge_{(\exists\vec{\alpha'}.Q'\supset C')\in\theta I_1} \mathcal{Q}; Q \wedge Q_2 \wedge Q'; \vec{\alpha'} \vdash C' \overset{\mathrm{solv}}{\rightsquigarrow} \epsilon \mid \theta'}{\mathcal{Q}; Q; \vec{\alpha} \vdash C_1 \overset{\mathrm{solv}}{\rightsquigarrow} Q_2 \mid \theta}$$

Simplification:

$$\frac{\mathrm{canon_g}(P_1) = \langle \vec{\alpha_2}, \theta_2, W_2\rangle \quad \mathrm{dom}(\theta_1)\cap\mathrm{dom}(\theta_2) = \varnothing}{\mathcal{Q} \vdash \langle \vec{\alpha_1}, \theta_1, W_g \uplus \{P_1\}, W_w\rangle \to \langle \vec{\alpha_1}\vec{\alpha_2}, \theta_1\cup\theta_2, W_g \cup W_2, W_w\rangle}$$

$$\frac{\mathrm{canon_w}(P_1) = \langle \vec{\alpha_2}, \theta_2, W_2\rangle \quad \mathrm{dom}(\theta_1)\cap\mathrm{dom}(\theta_2) = \varnothing}{\mathcal{Q} \vdash \langle \vec{\alpha_1}, \theta_1, W_g, W_w \uplus \{P_1\}\rangle \to \langle \vec{\alpha_1}\vec{\alpha_2}, \theta_1\cup\theta_2, W_g, W_w \cup W_2\rangle}$$

$$\frac{\mathrm{interact_g}(P_1, P_2) = W_3}{\mathcal{Q} \vdash \langle \vec{\alpha}, \theta, W_g \uplus \{P_1, P_2\}, W_w\rangle \to \langle \vec{\alpha}, \theta, W_g \cup W_3, W_w\rangle}$$

$$\frac{\mathrm{interact_w}(P_1, P_2) = W_3}{\mathcal{Q} \vdash \langle \vec{\alpha}, \theta, W_g, W_w \uplus \{P_1, P_2\}\rangle \to \langle \vec{\alpha}, \theta, W_g, W_w \cup W_3\rangle}$$

$$\frac{\mathrm{simplify}(P, P_1) = W_2}{\mathcal{Q} \vdash \langle \vec{\alpha}, \theta, W_g \uplus \{P\}, W_w \uplus \{P_1\}\rangle \to \langle \vec{\alpha}, \theta, W_g \uplus \{P\}, W_w \cup W_2\rangle}$$

$$\frac{\mathrm{topreact_g}(\mathcal{Q}, P_1) = \langle \epsilon, W_2\rangle}{\mathcal{Q} \vdash \langle \vec{\alpha}, \theta, W_g \uplus \{P_1\}, W_w\rangle \to \langle \vec{\alpha}, \theta, W_g \cup W_2, W_w\rangle}$$

$$\frac{\mathrm{topreact_w}(\mathcal{Q}, P_1) = \langle \vec{\alpha_2}, W_2\rangle}{\mathcal{Q} \vdash \langle \vec{\alpha_1}, \theta, W_g, W_w \uplus \{P_1\}\rangle \to \langle \vec{\alpha_1}\vec{\alpha_2}, \theta, W_g, W_w \cup W_2\rangle}$$

$$\frac{\beta_1 \in \vec{\alpha} \quad \beta_1 \notin \mathrm{ftv}(\tau_2)}{\mathrm{extract}(\beta_1 \simeq \tau_2, \vec{\alpha}) = \langle \epsilon, \{\beta_1 \mapsto \tau_2\}\rangle}$$

$$\frac{\beta_2 \in \vec{\alpha} \quad \beta_2 \notin \mathrm{ftv}(\tau_1)}{\mathrm{extract}(\tau_1 \simeq \beta_2, \vec{\alpha}) = \langle \epsilon, \{\beta_2 \mapsto \tau_1\}\rangle}$$

$$\frac{(\tau_1 \notin \vec{\alpha} \vee \tau_1 \in \mathrm{ftv}(\tau_2)) \quad (\tau_2 \notin \vec{\alpha} \vee \tau_2 \in \mathrm{ftv}(\tau_1))}{\mathrm{extract}(\tau_1 \simeq \tau_2, \vec{\alpha}) = \langle \tau_1 \simeq \tau_2, \varnothing\rangle}$$

$$\frac{}{\mathrm{extract}(D\vec{\tau}, \vec{\alpha}) = \langle D\vec{\tau}, \varnothing\rangle}$$

$$\frac{}{\mathrm{flat}(\epsilon) = \varnothing}$$

$$\frac{\mathrm{flat}(Q_1) = W_1 \quad \mathrm{flat}(Q_2) = W_2}{\mathrm{flat}(Q_1 \wedge Q_2) = W_1 \cup W_2}$$

$$\frac{}{\mathrm{flat}(\tau_1 \simeq \tau_2) = \{\tau_1 \simeq \tau_2\}}$$

$$\frac{}{\mathrm{flat}(D\vec{\tau}) = \{D\vec{\tau}\}}$$

$$\frac{\begin{array}{c}\mathcal{Q} \vdash \langle \vec{\alpha}, \varnothing, \mathrm{flat}(Q), \mathrm{flat}(Q_1)\rangle \to^* \langle \vec{\alpha'}, \theta', W', W_2'\rangle \not\to \\ W_2 = \bigcup\{W \mid P_2' \in W_2', \mathrm{extract}(\theta' P_2', \vec{\alpha'}) = \langle W, R\rangle\} \\ R_2 = \bigcup\{R \mid P_2' \in W_2', \mathrm{extract}(\theta' P_2', \vec{\alpha'}) = \langle W, R\rangle\} \\ \theta = \{\beta \mapsto \tau \mid \beta \in \mathrm{dom}(R_2), \forall \beta \mapsto \tau' \in R_2.\, \tau = \theta\tau'\}\end{array}}{\mathcal{Q}; Q; \vec{\alpha} \vdash Q_1 \overset{\mathrm{simpl}}{\rightsquigarrow} \theta \bigwedge W_2 \mid \theta}$$

Canonicalization:

$$\frac{}{\text{canon}_l(\mathcal{T}\vec{\tau_1} \simeq \mathcal{T}\vec{\tau_2}) = \langle \epsilon, \emptyset, \{\tau_1 \simeq \tau_2 \in \overrightarrow{\tau_1 \simeq \tau_2} \mid \tau_1 \neq \tau_2\}\rangle}$$

$$\frac{}{\text{canon}_l(F\vec{\tau} \simeq F\vec{\tau}) = \langle \epsilon, \emptyset, \emptyset \rangle}$$

8.4 ML Type Inference by HM(X)

[EL04]

8.5 Bidirectional Type Checking for System-F

[DK13][JVWS07]

8.5.1 Language

Syntax:

$$
\begin{array}{llll}
\text{Type Variables} & \alpha & ::= & \cdots \\
\text{Variables} & x & ::= & \cdots \\
\text{Mono Types} & \tau & ::= & \alpha \\
 & & | & \tau_1 \to \tau_2 \\
\text{Types} & \sigma & ::= & \alpha \\
 & & | & \sigma_1 \to \sigma_2 \\
 & & | & \forall \alpha.\, \sigma \\
\text{Terms} & e & ::= & x \\
 & & | & \lambda x.\, e \\
 & & | & \lambda x : \sigma.\, e \\
 & & | & e_1\ e_2 \\
 & & | & e : \sigma \\
\text{Contexts} & \Gamma & ::= & \epsilon \\
 & & | & \Gamma_1 + \Gamma_2 \\
 & & | & x : \sigma \\
 & & | & \alpha
\end{array}
$$

Context Member:

$$
\frac{}{x : \sigma \in x : \sigma} \quad \frac{x : \sigma \in \Gamma_1}{x : \sigma \in \Gamma_1 + \Gamma_2} \quad \frac{x : \sigma \in \Gamma_2}{x : \sigma \in \Gamma_1 + \Gamma_2}
$$

$$
\frac{}{\alpha \in \alpha} \quad \frac{\alpha \in \Gamma_1}{\alpha \in \Gamma_1 + \Gamma_2} \quad \frac{\alpha \in \Gamma_2}{\alpha \in \Gamma_1 + \Gamma_2}
$$

Type Validity:

$$
\frac{\alpha \in \Gamma}{\Gamma \vdash \alpha}
$$

$$
\frac{\Gamma \vdash \sigma_1 \quad \Gamma \vdash \sigma_2}{\Gamma \vdash \sigma_1 \to \sigma_2}
$$

$$
\frac{\Gamma, \alpha \vdash \sigma}{\Gamma \vdash \forall \alpha.\, \sigma}
$$

Term Typing (predicative):

$$
\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{Var}
$$

$$
\frac{\Gamma \vdash \sigma_1 \quad \Gamma, x : \sigma_1 \vdash e : \sigma_2}{\Gamma \vdash \lambda x.\, e : \sigma_1 \to \sigma_2}\ \text{Abs}
$$

$$
\frac{\Gamma \vdash \sigma_1 \quad \Gamma, x : \sigma_1 \vdash e : \sigma_2}{\Gamma \vdash \lambda x : \sigma_1.\, e : \sigma_1 \to \sigma_2}\ \text{AnnAbs}
$$

$$
\frac{\Gamma \vdash e_1 : \sigma_2 \to \sigma \quad \Gamma \vdash e_2 : \sigma_2}{\Gamma \vdash e_1\ e_2 : \sigma}\ \text{App}
$$

$$
\frac{\Gamma, \alpha \vdash e : \sigma}{\Gamma \vdash e : \forall \alpha.\, \sigma}\ \text{Gen}
$$

$$
\frac{\Gamma \vdash e : \forall \alpha.\, \sigma \quad \Gamma \vdash \tau}{\Gamma \vdash e : \sigma[\alpha \leftarrow \tau]}\ \text{Inst}
$$

8.5.2 Bidirectional Typing

Bidirectional Typing:

$$\frac{\Gamma \vdash e \Rightarrow \sigma_1 \quad \Gamma \vdash \sigma_1 \le \sigma_2}{\Gamma \vdash e \Leftarrow \sigma_2}\ \text{Sub}$$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x \Rightarrow \sigma}\ \text{Var}$$

$$\frac{\Gamma \vdash \sigma \quad \Gamma \vdash e \Leftarrow \sigma}{\Gamma \vdash (e : \sigma) \Rightarrow \sigma}\ \text{Ann}$$

$$\frac{\Gamma, \alpha \vdash e \Leftarrow \sigma}{\Gamma \vdash e \Leftarrow \forall \alpha.\, \sigma}\ \text{TyAbs}$$

$$\frac{\Gamma, x : \sigma_1 \vdash e \Leftarrow \sigma_2}{\Gamma \vdash \lambda x.\, e \Leftarrow \sigma_1 \to \sigma_2}\ \text{Abs}$$

$$\frac{\Gamma \vdash \tau_1 \to \tau_2 \quad \Gamma, x : \tau_1 \vdash e \Leftarrow \tau_2}{\Gamma \vdash \lambda x.\, e \Rightarrow \tau_1 \to \tau_2}\ \text{AbsSyn}$$

$$\frac{\Gamma, x : \sigma_1 \vdash e \Leftarrow \sigma_2}{\Gamma \vdash \lambda x : \sigma_1.\, e \Leftarrow \sigma_1 \to \sigma_2}\ \text{AnnAbs}$$

$$\frac{\Gamma \vdash \sigma_1 \to \tau_2 \quad \Gamma, x : \sigma_1 \vdash e \Leftarrow \tau_2}{\Gamma \vdash \lambda x : \sigma_1.\, e \Rightarrow \sigma_1 \to \tau_2}\ \text{AnnAbsSyn}$$

$$\frac{\Gamma \vdash e_1 \Rightarrow \sigma_1 \quad \Gamma \vdash \sigma_1 \le \sigma_2 \to \sigma \quad \Gamma \vdash e_2 \Leftarrow \sigma_2}{\Gamma \vdash e_1\ e_2 \Rightarrow \sigma}\ \text{App}$$

Subtyping:

$$\frac{\alpha \in \Gamma}{\Gamma \vdash \alpha \le \alpha}\ \text{Var}$$

$$\frac{\Gamma \vdash \sigma_1' \le \sigma_1 \quad \Gamma \vdash \sigma_2 \le \sigma_2'}{\Gamma \vdash \sigma_1 \to \sigma_2 \le \sigma_1' \to \sigma_2'}\ \text{Arrow}$$

$$\frac{\Gamma \vdash \tau_1 \quad \Gamma \vdash \sigma_1[\alpha_1 \leftarrow \tau_1] \le \sigma_2}{\Gamma \vdash \forall \alpha_1.\, \sigma_1 \le \sigma_2}\ \text{Spec}$$

$$\frac{\Gamma, \alpha_2 \vdash \sigma_1 \le \sigma_2}{\Gamma \vdash \sigma_1 \le \forall \alpha_2.\, \sigma_2}\ \text{Skol}$$

Subsumption:

$$\frac{}{\Gamma \vdash \sigma \le \sigma}\ \text{Refl}$$

$$\frac{\Gamma \vdash \tau_1 \quad \Gamma \vdash \sigma_1[\alpha_1 \leftarrow \tau_1] \le \sigma_2}{\Gamma \vdash \forall \alpha_1.\, \sigma_1 \le \sigma_2}\ \text{Spec}$$

$$\frac{\alpha \notin \mathit{ftv}(\sigma_1)}{\Gamma \vdash \forall \alpha.\, \sigma_1 \to \sigma_2 \le \sigma_1 \to \forall \alpha.\, \sigma_2}\ \text{WeakSpec}$$

8.5.3 Algorithmic Type Inference

Algorithmic context:

$$\begin{array}{rcl} \Gamma & ::= & \epsilon \\ & | & \Gamma, \alpha \\ & | & \Gamma, x : \sigma \\ & | & \Gamma, \hat{\alpha} \\ & | & \Gamma, \hat{\alpha} = \tau \\ & | & \Gamma, \alpha \mapsto \hat{\alpha} \end{array}$$

Substitution:

$$\frac{}{[\Gamma]\alpha = \alpha}$$

$$\frac{\hat{\alpha} = \tau \in \Gamma}{[\Gamma]\hat{\alpha} = \tau}$$

$$\frac{[\Gamma](\sigma_1) = \sigma_1' \quad [\Gamma](\sigma_2) = \sigma_2'}{[\Gamma](\sigma_1 \to \sigma_2) = \sigma_1' \to \sigma_2'}$$

$$\frac{[\Gamma]\sigma = \sigma'}{[\Gamma](\forall \alpha.\, \sigma) = \forall \alpha.\, \sigma'}$$

Bidirectional typing:

$$\frac{\Gamma \vdash e \Rightarrow \sigma_1 \mid \Theta \quad \Theta \vdash [\Theta]\sigma_1 \leq [\Theta]\sigma_2 \mid \Delta}{\Gamma \vdash e \Leftarrow \sigma_2 \mid \Delta}\ \text{Sub}$$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x \Rightarrow \sigma \mid \Gamma}\ \text{Var}$$

$$\frac{\Gamma \vdash e \Leftarrow \sigma \mid \Delta}{\Gamma \vdash e : \sigma \Rightarrow \sigma \mid \Delta}\ \text{Ann}$$

$$\frac{\Gamma, \alpha \vdash e \Leftarrow \sigma \mid \Delta, \alpha, \Theta}{\Gamma \vdash e \Leftarrow \forall \alpha.\, \sigma \mid \Delta}\ \text{TyAbs}$$

$$\frac{\Gamma, x : \sigma_1 \vdash e \Leftarrow \sigma_2 \mid \Delta, x : \sigma_1, \Theta}{\Gamma \vdash \lambda x.\, e \Leftarrow \sigma_1 \to \sigma_2 \mid \Delta}\ \text{Abs}$$

$$\frac{\Gamma, \hat{\alpha}_1, \hat{\alpha}_2, x : \hat{\alpha}_1 \vdash e \Leftarrow \hat{\alpha}_2 \mid \Delta, x : \hat{\alpha}_1, \Theta}{\Gamma \vdash \lambda x.\, e \Rightarrow \hat{\alpha}_1 \to \hat{\alpha}_2 \mid \Delta}\ \text{AbsSyn}$$

$$\frac{\Gamma, x : \sigma_1 \vdash e \Leftarrow \sigma_2 \mid \Delta, x : \sigma_1, \Theta}{\Gamma \vdash \lambda x : \sigma_1.\, e \Leftarrow \sigma_1 \to \sigma_2 \mid \Delta}\ \text{AnnAbs}$$

$$\frac{\Gamma, \hat{\alpha}_2, x : \sigma_1 \vdash e \Leftarrow \hat{\alpha}_2 \mid \Delta, x : \sigma_1, \Theta}{\Gamma \vdash \lambda x : \sigma_1.\, e \Rightarrow \sigma_1 \to \hat{\alpha}_2 \mid \Delta}\ \text{AnnAbsSyn}$$

$$\frac{\Gamma \vdash e_1 \Rightarrow \sigma_1 \mid \Theta_1 \quad \Theta_1 \vdash [\Theta_1]\sigma_1 \leq \sigma_2 \to \sigma \mid \Theta_2 \quad \Theta_2 \vdash e_2 \Leftarrow [\Theta_2]\sigma_2 \mid \Delta}{\Gamma \vdash e_1\ e_2 \Rightarrow \sigma \mid \Delta}\ \text{App}$$

Subtyping:

$$\frac{\alpha \in \Gamma}{\Gamma \vdash \alpha \leq \alpha \mid \Gamma}$$

$$\frac{\hat{\alpha} \in \Gamma}{\Gamma \vdash \hat{\alpha} \leq \hat{\alpha} \mid \Gamma}$$

$$\frac{\Gamma \vdash \sigma_1 \leq \sigma_1' \mid \Theta \quad \Theta \vdash [\Theta]\sigma_2 \leq [\Theta]\sigma_2' \mid \Delta}{\Gamma \vdash \sigma_1 \to \sigma_2 \leq \sigma_1' \to \sigma_2' \mid \Delta}$$

$$\frac{\Gamma, \alpha \mapsto \hat{\alpha}, \hat{\alpha} \vdash \sigma_1[\alpha \leftarrow \hat{\alpha}] \leq \sigma_2 \mid \Delta, \alpha \mapsto \hat{\alpha}, \Theta}{\Gamma \vdash \forall \alpha.\, \sigma_1 \leq \sigma_2 \mid \Delta}$$

$$\frac{\Gamma, \alpha \vdash \sigma_1 \leq \sigma_2 \mid \Delta, \alpha, \Theta}{\Gamma \vdash \sigma_1 \leq \forall \alpha.\, \sigma_2 \mid \Delta}$$

$$\frac{\hat{\alpha}_1 \notin \mathit{ftv}(\sigma_2) \quad \hat{\alpha}_1 \in \Gamma \quad \Gamma \vdash \hat{\alpha}_1 \simeq \sigma_2 \mid \Delta}{\Gamma \vdash \hat{\alpha}_1 \leq \sigma_2 \mid \Delta}$$

$$\frac{\hat{\alpha}_2 \notin \mathit{ftv}(\sigma_1) \quad \hat{\alpha}_2 \in \Gamma \quad \Gamma \vdash \sigma_1 \simeq \hat{\alpha}_2 \mid \Delta}{\Gamma \vdash \sigma_1 \leq \hat{\alpha}_2 \mid \Delta}$$

Instantiation:

$$\frac{\Gamma_1 \vdash \tau}{\Gamma_1, \hat{\alpha}, \Gamma_2 \vdash \hat{\alpha} \simeq \tau \mid \Gamma_1, \hat{\alpha} = \tau, \Gamma_2}$$

$$\frac{\Gamma_1 \vdash \tau}{\Gamma_1, \hat{\alpha}, \Gamma_2 \vdash \tau \simeq \hat{\alpha} \mid \Gamma_1, \hat{\alpha} = \tau, \Gamma_2}$$

$$\frac{}{\Gamma_1, \hat{\alpha}_1, \Gamma_2, \hat{\alpha}_2, \Gamma_3 \vdash \hat{\alpha}_1 \simeq \hat{\alpha}_2 \mid \Gamma_1, \hat{\alpha}_1, \Gamma_2, \hat{\alpha}_2 = \hat{\alpha}_1, \Gamma_3}$$

$$\frac{\Gamma_1, \hat{\alpha}_3, \hat{\alpha}_2, \hat{\alpha}_1 = \hat{\alpha}_2 \to \hat{\alpha}_3, \Gamma_2 \vdash \sigma_2 \simeq \hat{\alpha}_2 \mid \Theta \quad \Theta \vdash \hat{\alpha}_3 \simeq [\Theta]\sigma_3 \mid \Delta}{\Gamma_1, \hat{\alpha}_1, \Gamma_2 \vdash \hat{\alpha}_1 \simeq \sigma_2 \to \sigma_3 \mid \Delta}$$

$$\frac{\Gamma_1, \hat{\alpha}_3, \hat{\alpha}_2, \hat{\alpha}_1 = \hat{\alpha}_2 \to \hat{\alpha}_3, \Gamma_2 \vdash \hat{\alpha}_2 \simeq \sigma_2 \mid \Theta \quad \Theta \vdash [\Theta]\sigma_3 \simeq \hat{\alpha}_3 \mid \Delta}{\Gamma_1, \hat{\alpha}_1, \Gamma_2 \vdash \sigma_2 \to \sigma_3 \simeq \hat{\alpha}_1 \mid \Delta}$$

$$\frac{\Gamma_1, \hat{\alpha_1}, \Gamma_2, \alpha_2 \vdash \hat{\alpha_1} \simeq \sigma_2 \mid \Delta, \alpha_2, \Theta}{\Gamma_1, \hat{\alpha_1}, \Gamma_2 \vdash \hat{\alpha_1} \simeq \forall \alpha_2 .\, \sigma_2 \mid \Delta}$$

$$\frac{\Gamma_1, \hat{\alpha_2}, \Gamma_2, \alpha_1 \mapsto \hat{\alpha_1}, \hat{\alpha_1} \vdash \sigma_1[\alpha_1 \leftarrow \hat{\alpha_1}] \simeq \hat{\alpha_2} \mid \Delta, \alpha_1 \mapsto \hat{\alpha_1}, \Theta}{\Gamma_1, \hat{\alpha_2}, \Gamma_2 \vdash \forall \alpha_1 .\, \sigma_1 \simeq \hat{\alpha_2} \mid \Delta}$$

Subsumption:

$$\frac{}{\Gamma \vdash \hat{\alpha_1} \preceq \hat{\alpha_2} \to \hat{\alpha_3} \mid \Gamma, \hat{\alpha_2}, \hat{\alpha_3}, \hat{\alpha_1} = \hat{\alpha_2} \to \hat{\alpha_3}}$$

$$\frac{}{\Gamma \vdash \sigma_1 \to \sigma_2 \preceq \sigma_1 \to \sigma_2 \mid \Gamma}$$

$$\frac{\alpha \notin \mathit{ftv}(\sigma_1)}{\Gamma \vdash \forall \alpha .\, \sigma_1 \to \sigma_2 \preceq \sigma_1 \to \forall \alpha .\, \sigma_2 \mid \Gamma}$$

$$\frac{\nexists \sigma_1' \to \sigma_2' = \sigma_1 .\, \alpha_1 \notin \mathit{ftv}(\sigma_1') \quad \Gamma, \hat{\alpha_1} \vdash \sigma_1[\alpha_1 \leftarrow \hat{\alpha_1}] \preceq \sigma_2 \mid \Delta}{\Gamma \vdash \forall \alpha_1 .\, \sigma_1 \preceq \sigma_2 \mid \Delta}$$

8.6 System-FC with Explicit Kind Equality

[WHE13]

第 9 章

Static Memory Management and Regions

第 10 章

Dynamic Memory Management and Gabage Collections

10.1 WIP: On-the-Fly GC: Concurrent Tri-color Mark and Sweep

[DLM+78]

10.2 Memory Allocator with BitMap Free List

[UOO11][UO16]

10.2.1 Heap Structure

定義 59 (ビット (bit)). ビットとは、$B \in \{\top, \bot\}$ のこと。ビットの集合を $\mathcal{B} = \{\top, \bot\}$ と表記する。 □

セグメントクラスは N_c 個あり、それぞれのクラス $i \in [N_c]$ はブロックサイズ $\mathrm{sizeOfClass}(i)$ を持ち、$\forall i_1 < i_2. \mathrm{sizeOfClass}(i_1) < \mathrm{sizeOfClass}(i_2)$ を満たす。また、クラスそれぞれでセグメントが持つブロック数 $\mathrm{blockCountOfClass}(i)$ が決まっている。

定義 60 (セグメント (segment)). セグメントとは、以下による組 $S = (i, M, L)$ のことである:

- セグメントクラス $i \in [N_c]$。$\mathrm{subheapClass}(S) = i$ と表記する。
- ビットマップ $M \in \mathcal{B}^{\mathrm{blockCountOfClass}(i)}$。$\mathrm{bitmap}(S) = M$ と表記する。
- ブロック配列 $L \in \mathrm{Blk}^{\mathrm{blockCountOfClass}(i)}$。$\mathrm{block}(S) = L$ と表記する。

セグメントのクラスを Seg と表記する。 □

サブヒープは、N_c 個のクラスによるヒープ分割領域である。

定義 61 (サブヒープ (sub-heap)). クラス i のサブヒープとは、以下による組 $V_i = (R)$ のことである:

- 空きセグメント番号の列 $R \in \mathbb{N}^*$。$\mathrm{free}(V_i) = R$ と表記する。

□

定義 62 (ヒープ (heap)). ヒープとは、以下による組 $H = (A, \{V_i\}_{i \in [N_c]}, F)$ のことである:

- セグメントの列 $A \in \mathrm{Seg}^*_\bot$。$\mathrm{segments}(H) = A$ と表記する。
- サブヒープの族 $\{V_i\}_{i \in [N_c]}$。$\mathrm{subheap}_i(H) = V_i$ と表記する。
- 空きセグメントの列 $F \in \mathbb{N}^*$。$\mathrm{free}(H) = F$ と表記する。

□

10.2.2 Initialize

Ensure: H
1: **for** $i \in [N_c]$ **do**
2: $V_i \leftarrow (\mathrm{sizeOfClass(i)}, \epsilon)$
3: **end for**
4: $H \leftarrow (\{V_i\}_{i \in [N_c]}, \epsilon)$

10.2.3 Allocation

Require: $H, size$

Ensure: H, blk

$cls = \mathrm{classOfSize}(size)$

if $cls = -1$ **then**

$blk \leftarrow (\mathrm{FreeSize}, \mathrm{allocFreeSize}(size))$

else

$V_{cls} \leftarrow \mathrm{subheap}_{cls}(H)$

if $|\mathrm{free}(V_{cls})| > 0$ **then**

$i_{\mathrm{seg}} \cdot F \leftarrow \mathrm{free}(V_{cls})$

$seg \leftarrow \mathrm{segments}(H)(i_{\mathrm{seg}})$

$i_{\mathrm{blk}} \leftarrow \mathrm{pick}(\{i \mid i \in [\mathrm{blockCountOfClass}(cls)], \mathrm{bitmap}(seg)(i) = \bot\})$

else if $|\mathrm{free}(H)| > 0$ **then**

$i_{\mathrm{seg}} \cdot F \leftarrow \mathrm{free}(H)$

$\mathrm{free}(H) \leftarrow F$

$\mathrm{segments}(H)(i_{\mathrm{seg}}) \leftarrow \mathrm{newSegment}(cls)$

$\mathrm{free}(V_{cls}) \leftarrow i_{\mathrm{seg}} \cdot \mathrm{free}(V_{cls})$

$seg \leftarrow \mathrm{segments}(H)(i_{\mathrm{seg}})$

$i_{\mathrm{blk}} \leftarrow 1$

else

$\mathrm{segments}(H) \leftarrow \mathrm{segments}(H) \cdot \bot$

$i_{\mathrm{seg}} \leftarrow |\mathrm{segments}(H)|$

$\mathrm{segments}(H)(i_{\mathrm{seg}}) \leftarrow \mathrm{newSegment}(cls)$

$\mathrm{free}(V_{cls}) \leftarrow i_{\mathrm{seg}} \cdot \mathrm{free}(V_{cls})$

$seg \leftarrow \mathrm{segments}(H)(i_{\mathrm{seg}})$

$i_{\mathrm{blk}} \leftarrow 1$

end if

$\mathrm{bitmap}(seg)(i_{\mathrm{blk}}) \leftarrow \top$

if $\forall i \in [\mathrm{blockCountOfClass}(cls)].\ \mathrm{bitmap}(seg)(i) = \top$ **then**

$i_{\mathrm{seg}} \cdot F \leftarrow \mathrm{free}(V_{cls})$

$\mathrm{free}(V_{cls}) \leftarrow F$

end if

$blk \leftarrow (\mathrm{OnSubHeap}, i_{\mathrm{seg}}, i_{\mathrm{blk}})$

end if

定義 63.

$$\mathrm{classOfSize}(s) = \begin{cases} -1 & (\forall i \in [N_c].\ \mathrm{sizeOfClass}(i) < s) \\ \max\{i \in [N_c] \mid s \leq \mathrm{sizeOfClass}(i)\} & (\text{otherwise}) \end{cases}$$

□

定義 64.

$$\mathrm{newSegment}(i) = (i, \bot^{\mathrm{blockCountOfClass}(i)}, \mathrm{newBlock}(\mathrm{sizeOfClass(i)})^{\mathrm{blockCountOfClass}(i)})$$

□

10.2.4 Free

Require: H, blk
Ensure: H

if $blk = (\text{FreeSize}, body)$ **then**
  $\text{freeFreeSize}(blk)$
else if $blk = (\text{OnSubHeap}, i_{seg}, i_{blk})$ **then**
  $seg \leftarrow \text{segments}(H)(i_{\text{seg}})$
  $cls \leftarrow \text{subheapClass}(seg)$
  $V_{cls} \leftarrow \text{subheap}_{cls}(H)$
  $\text{bitmap}(seg)(i_{\text{blk}}) \leftarrow \bot$
  if $\forall i \in [\text{blockCountOfClass}(cls)].\ \text{bitmap}(seg)(i) = \bot$ **then**
    $\text{free}(V_{cls}) \leftarrow \langle i \in \text{free}(V_{cls}) \mid i \neq i_{\text{seg}} \rangle$
    $\text{free}(H) \leftarrow i_{\text{seg}} \cdot \text{free}(H)$
  else if $i_{\text{seg}} \notin \text{free}(V_{cls})$ **then**
    $\text{free}(V_{cls}) \leftarrow i_{\text{seg}} \cdot \text{free}(V_{cls})$
  end if
end if

10.3 Concurrent Garbage Collector for Functional Programs

[UOO11][UO16][GD20]

10.3.1 Heap Structure

Heap $\mathcal{H} = (\mathcal{F}, (H_c, H_{c+1}, \dots, H_{c+n}), \mathcal{M})$

$\mathcal{F} \in \text{Seg}^*$ A pool of free segments.
$H_i \in \text{Seg}_i^* \times \text{Seg}_i \times \text{Seg}_i^*$ A sub-heap to allocate 2^i-bytes blocks.
$\mathcal{M}$ A special sub-heap for large objects.

Segment $S_i = (\mathcal{B}_i, P, \mathcal{C})$

$\mathcal{B}_i$ Allocation blocks of the same size.
P A pointer to the next block.
$\mathcal{C}$ A bitmap represented object liveness.

10.3.2 Allocation and GC

第 11 章

I/O Management and Concurrency

第 12 章

Code Generation and Virtual Machines

第 13 章

Program Stability and Compatibility

第 14 章

Program Separation and Linking

第 15 章

Syntax and Parsing

15.1 WIP: Parsing by LR Method

[Knu65]

15.2 Syntax and Semantics of PEG

[For02], [For04]

15.2.1 Syntax

$$
\begin{array}{rcll}
e & ::= & \epsilon & \text{(epsilon)} \\
 & | & \sigma & \text{(terminal)} \\
 & | & A & \text{(non-terminal)} \\
 & | & ee & \text{(sequence)} \\
 & | & e \mathbin{/} e & \text{(alternative)} \\
 & | & e^{*} & \text{(repetition)} \\
 & | & !\,e & \text{(not predicate)} \\
\sigma & \in & \Sigma & \\
A & \in & N &
\end{array}
$$

定義 65. PEG 文法とは，以下による組 $G = (\Sigma, N, R, e_0)$ のことである．

Σ　終端記号の集合．

N　非終端記号の集合．

R　$A \to e$ を満たす規則の集合．規則は，非終端記号に対して必ず一つ．

e_0　初期式．

□

15.2.2 Structured Semantics

$$\frac{}{\langle \epsilon, x\rangle \to \mathrm{s}(\epsilon)}$$

$$\frac{}{\langle \sigma, \sigma x\rangle \to \mathrm{s}(\sigma)} \qquad \frac{\sigma \neq \sigma'}{\langle \sigma, \sigma' x\rangle \to \mathrm{f}} \qquad \frac{}{\langle \sigma, \epsilon\rangle \to \mathrm{f}}$$

$$\frac{A \leftarrow e \in R \quad \langle e, x\rangle \to o}{\langle A, x\rangle \to o}$$

$$\frac{\langle e_1, x_1x_2y\rangle \to \mathrm{s}(x_1) \quad \langle e_2, x_2y\rangle \to \mathrm{s}(x_2)}{\langle e_1e_2, x_1x_2y\rangle \to \mathrm{s}(x_1x_2)} \qquad \frac{\langle e_1, x\rangle \to \mathrm{f}}{\langle e_1e_2, x\rangle \to \mathrm{f}} \qquad \frac{\langle e_1, x_1y\rangle \to \mathrm{s}(x_1) \quad \langle e_2, y\rangle \to \mathrm{f}}{\langle e_1e_2, x_1y\rangle \to \mathrm{f}}$$

$$\frac{\langle e_1, xy\rangle \to \mathrm{s}(x)}{\langle e_1 \mathbin{/} e_2, xy\rangle \to \mathrm{s}(x)} \qquad \frac{\langle e_1, x\rangle \to \mathrm{f} \quad \langle e_2, x\rangle \to o}{\langle e_1 \mathbin{/} e_2, x\rangle \to o}$$

$$\frac{\langle e, x_1x_2y\rangle \to \mathrm{s}(x_1) \quad \langle e^{*}, x_2y\rangle \to \mathrm{s}(x_2)}{\langle e^{*}, x_1x_2y\rangle \to \mathrm{s}(x_1x_2)} \qquad \frac{\langle e, x\rangle \to \mathrm{f}}{\langle e^{*}, x\rangle \to \mathrm{s}(\epsilon)}$$

$$\frac{\langle e, x\rangle \to \mathrm{f}}{\langle !e, x\rangle \to \mathrm{s}(\epsilon)} \qquad \frac{\langle e, xy\rangle \to \mathrm{s}(x)}{\langle !e, xy\rangle \to \mathrm{f}}$$

$$
\begin{aligned}
[\![(\Sigma, N, R, e_0)]\!] &= [\![e_0]\!] \\
[\![e]\!] &= \{x \in \Sigma^{*} \mid \langle e, x\rangle \to \mathrm{s}(x)\}
\end{aligned}
$$

15.2.3 Equivalence

Abbreviations

$$\begin{aligned} \& e &= !(!\, e) && \text{(and predicate)} \\ e^{+} &= ee^{*} && \text{(positive repetition)} \\ e^{?} &= e/\epsilon && \text{(optional)} \end{aligned}$$

Associativity

$$\frac{}{[\![e_1/(e_2/e_3)]\!] = [\![(e_1/e_2)/e_3]\!]}$$

$$\frac{}{[\![e_1(e_2e_3)]\!] = [\![(e_1e_2)e_3]\!]}$$

Epsilon

$$\frac{}{[\![\epsilon/e]\!] = [\![\epsilon]\!]}$$

$$\frac{}{[\![e\epsilon]\!] = [\![e]\!]} \qquad \frac{}{[\![\epsilon e]\!] = [\![e]\!]}$$

Repetition

$$M \quad ::= \quad eM \mid \epsilon$$

$$\frac{}{[\![e^{*}]\!] = [\![M]\!]}$$

15.2.4 Producing Analysis

$s ::= 0 \mid 1,\ \ o ::= \mathrm{s} \mid \mathrm{f}$

- $\epsilon \rightharpoonup 0$
- $\sigma \rightharpoonup 1$
- $\sigma \rightharpoonup \mathrm{f}$
- $A \leftarrow e \in R,\ \ e \rightharpoonup o$ ならば $A \rightharpoonup o$
- $e_1 \rightharpoonup 0,\ \ e_2 \rightharpoonup 0$ ならば $e_1e_2 \rightharpoonup 0$
- $e_1 \rightharpoonup 1,\ \ e_2 \rightharpoonup s$ ならば $e_1e_2 \rightharpoonup 1$
- $e_1 \rightharpoonup \mathrm{s},\ \ e_2 \rightharpoonup 1$ ならば $e_1e_2 \rightharpoonup 1$
- $e_1 \rightharpoonup \mathrm{f}$ ならば $e_1e_2 \rightharpoonup \mathrm{f}$
- $e_1 \rightharpoonup \mathrm{s},\ \ e_2 \rightharpoonup \mathrm{f}$ ならば $e_1e_2 \rightharpoonup \mathrm{f}$
- $e_1 \rightharpoonup \mathrm{s}$ ならば $e_1 \,/\, e_2 \rightharpoonup \mathrm{s}$
- $e_1 \rightharpoonup \mathrm{f},\ \ e_2 \rightharpoonup o$ ならば $e_1 \,/\, e_2 \rightharpoonup o$
- $e \rightharpoonup 1$ ならば $e^{*} \rightharpoonup 1$
- $e \rightharpoonup \mathrm{f}$ ならば $e^{*} \rightharpoonup \mathrm{f}$
- $e \rightharpoonup \mathrm{s}$ ならば $!\, e \rightharpoonup \mathrm{f}$

- $e \rightharpoonup \mathrm{f}$ ならば $!e \rightharpoonup 0$

定理 66.

- $\langle e, x\rangle \to \mathrm{s}(\epsilon)$ ならば，$e \rightharpoonup 0$
- $\langle e, xy\rangle \to \mathrm{s}(x)$，$x \neq \epsilon$ ならば，$e \rightharpoonup 1$
- $\langle e, x\rangle \to \mathrm{f}$ ならば，$e \rightharpoonup \mathrm{f}$

□

系 67. $e \not\rightharpoonup o$ ならば，$\langle e, xy\rangle \not\to \mathrm{s}(x)$ かつ $\langle e, xy\rangle \not\to \mathrm{f}$ □

15.3 Haskell Parsing with PEG

[Sim10]

15.3.1 Lexical Syntax

program ::= (lexeme | whitespace)*
lexeme ::= qvarid
| qconid
| qvarsym
| qconsym
| literal
| special
| reservedop
| reservedid
literal ::= integer
| float
| char
| string
special ::= "(" | ")" | "," | ";" | "[" | "]" | "`" | "{" | "}"
whitespace ::= whitestuff$^+$
whitestuff ::= whitechar | comment | ncomment

whitechar ::= newline | "\v" | " " | "\t" | (Unicode whitespace)
newline ::= "\r\n" | "\r" | "\n" | "\f"
comment ::= dashes (!symbol any*)$^?$ newline
dashes ::= "-" ("-")$^+$
opencom ::= "{-"
closecom ::= "-}"
ncomment ::= opencom ANYs (ncomment ANYs)* closecom
ANYs ::= !(ANY* (opencom | closecom) ANY*) ANY*
ANY ::= graphic | whitechar
any ::= graphic | " " | "\t"
graphic ::= small | large | symbol | digit | special | "\"" | "'"
small ::= "a" | "b" | ··· | "z" | (Unicode lowercase letter) | "_"
large ::= "A" | "B" | ··· | "Z" | (Unicode uppercase letter) | (Unicode titlecase letter)
symbol ::= "!" | "#" | "$" | "%" | "&" | "*" | "+" | "." | "/" | "<" | "=" | ">"
| "?" | "@" | "\\" | "^" | "|" | "-" | "~" | ":"
| !(symbol | "_" | "\"" | "'") uniSymbol
uniSymbol ::= (Unicode symbol) | (Unicode punctuation)
digit ::= "0" | "1" | ··· | "9" | (Unicode decimal digit)
octit ::= "0" | "1" | ··· | "7"
hexit ::= digit | "A" | ··· | "F" | "a" | ··· | "f"
varid ::= !(reservedid !other) small other*
conid ::= large other*
other ::= small | large | digit | "'"
reservedid ::= "case" | "class" | "data" | "default" | "deriving" | "do" | "else"
| "foreign" | "if" | "import" | "in" | "infix" | "infixl" | "infixr"
| "instance" | "let" | "module" | "newtype" | "of" | "then" | "type"
| "where" | "_"
varsym ::= !((reservedop | dashes) !symbol | ":") symbol$^+$
consym ::= !(reservedop !symbol) ":" symbol$^+$
reservedop ::= ".." | ":" | "::" | "=" | "\\" | "<-" | "->" | "@" | "~" | "=>"

$$
\begin{array}{rcl}
\mathit{modid} & ::= & (\mathit{conid}\ \texttt{"."})^{*}\ \mathit{conid} \\
\mathit{qvarid} & ::= & (\mathit{modid}\ \texttt{"."})^{?}\ \mathit{varid} \\
\mathit{qconid} & ::= & (\mathit{modid}\ \texttt{"."})^{?}\ \mathit{conid} \\
\mathit{qvarsym} & ::= & (\mathit{modid}\ \texttt{"."})^{?}\ \mathit{varsym} \\
\mathit{qconsym} & ::= & (\mathit{modid}\ \texttt{"."})^{?}\ \mathit{consym}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{decimal} & ::= & \mathit{digit}^{+} \\
\mathit{octal} & ::= & \mathit{octit}^{+} \\
\mathit{hexdecimal} & ::= & \mathit{hexit}^{+} \\
\mathit{integer} & ::= & \mathit{decimal} \\
 & | & \texttt{"0o"}\ \mathit{octal} \mid \texttt{"0O"}\ \mathit{octal} \\
 & | & \texttt{"0x"}\ \mathit{hexdecimal} \mid \texttt{"0X"}\ \mathit{hexdecimal} \\
\mathit{float} & ::= & \mathit{decimal}\ \texttt{"."}\ \mathit{decimal}\ \mathit{exponent}^{?} \\
 & | & \mathit{decimal}\ \mathit{exponent} \\
\mathit{exponent} & ::= & (\texttt{"e"} \mid \texttt{"E"})\ (\texttt{"+"} \mid \texttt{"-"})\ \mathit{decimal} \\
\mathit{char} & ::= & \texttt{"'"}\ (!(\texttt{"'"} \mid \texttt{"\textbackslash\textbackslash"})\ \mathit{graphic} \mid \texttt{" "} \mid !\texttt{"\textbackslash\textbackslash\&"}\ \mathit{escape})\ \texttt{"'"} \\
\mathit{string} & ::= & \texttt{"\textbackslash""}\ (!(\texttt{"\textbackslash""} \mid \texttt{"\textbackslash\textbackslash"})\ \mathit{graphic} \mid \texttt{" "} \mid \mathit{escape} \mid \mathit{gap})^{*}\ \texttt{"\textbackslash""} \\
\mathit{escape} & ::= & \texttt{"\textbackslash\textbackslash"}(\mathit{charesc} \mid \mathit{ascii} \mid \mathit{decimal} \mid \texttt{"o"}\ \mathit{octal} \mid \texttt{"x"}\ \mathit{hexdecimal}) \\
\mathit{charesc} & ::= & \texttt{"a"} \mid \texttt{"b"} \mid \texttt{"f"} \mid \texttt{"n"} \mid \texttt{"r"} \mid \texttt{"t"} \mid \texttt{"v"} \mid \texttt{"\textbackslash\textbackslash"} \mid \texttt{"\textbackslash""} \mid \texttt{"'"} \mid \texttt{"\&"} \\
\mathit{ascii} & ::= & \texttt{"\^{}"}\ \mathit{cntrl} \mid \texttt{"NUL"} \mid \texttt{"SOH"} \mid \texttt{"STX"} \mid \texttt{"ETX"} \mid \texttt{"EOT"} \mid \texttt{"ENQ"} \mid \texttt{"ACK"} \mid \texttt{"BEL"} \mid \texttt{"BS"} \\
 & | & \texttt{"HT"} \mid \texttt{"LF"} \mid \texttt{"VT"} \mid \texttt{"FF"} \mid \texttt{"CR"} \mid \texttt{"SO"} \mid \texttt{"SI"} \mid \texttt{"DLE"} \mid \texttt{"DC1"} \mid \texttt{"DC2"} \mid \texttt{"DC3"} \\
 & | & \texttt{"DC4"} \mid \texttt{"NAK"} \mid \texttt{"SYN"} \mid \texttt{"ETB"} \mid \texttt{"CAN"} \mid \texttt{"EM"} \mid \texttt{"SUB"} \mid \texttt{"ESC"} \mid \texttt{"FS"} \mid \texttt{"GS"} \mid \texttt{"RS"} \\
 & | & \texttt{"US"} \mid \texttt{"SP"} \mid \texttt{"DEL"} \\
\mathit{cntrl} & ::= & \texttt{"A"} \mid \texttt{"B"} \mid \cdots \mid \texttt{"Z"} \mid \texttt{"@"} \mid \texttt{"["} \mid \texttt{"\textbackslash\textbackslash"} \mid \texttt{"]"} \mid \texttt{"\^{}"} \mid \texttt{"\_"} \\
\mathit{gap} & ::= & \texttt{"\textbackslash\textbackslash"}\ \mathit{whitechar}^{+}\ \texttt{"\textbackslash\textbackslash"}
\end{array}
$$

15.3.2 Preprocess for Layout

$$
L(s) = \begin{cases}
L_1(r', s) & (s = t : s', \mathrm{pos}(t) = (r', c'), \mathrm{islft}(t)) \\
\{c'\} : \langle c' \rangle : L_1(r', s) & (s = t : s', \mathrm{pos}(t) = (r', c'), \not\mathrm{islft}(t)) \\
\{1\} : \epsilon & (s = \epsilon)
\end{cases}
$$

$$
L_1(r, s) = \begin{cases}
\langle c' \rangle : L_2(r', c', t, s') & (s = t : s', \mathrm{pos}(t) = (r', c'), r \neq r') \\
L_2(r', c', t, s') & (s = t : s', \mathrm{pos}(t) = (r', c'), r = r') \\
\epsilon & (s = \epsilon)
\end{cases}
$$

$$
L_2(r_1, c_1, t_1, s) = \begin{cases}
t_1 : t_2 : L_1(r_2, s') & (\mathrm{islt}(t_1), s = t_2 : s', \mathrm{pos}(t_2) = (r_2, c_2), t_2 = \texttt{"\{"}) \\
t_1 : \{c_2\} : \langle c_2 \rangle : t_2 : L_1(r_2, s') & (\mathrm{islt}(t_1), s = t_2 : s', \mathrm{pos}(t_2) = (r_2, c_2), t_2 \neq \texttt{"\{"}) \\
t_1 : \{1\} : \epsilon & (\mathrm{islt}(t_1), s = \epsilon) \\
t_1 : L_1(r_1, s) & (\not\mathrm{islt}(t_1))
\end{cases}
$$

$$
\mathrm{islft}(t) = \begin{cases}
\top & (t = \texttt{"module"}) \\
\bot & (\text{otherwise})
\end{cases}
$$

$$
\mathrm{islt}(t) = \begin{cases}
\top & (t = \texttt{"let"}) \\
\top & (t = \texttt{"where"}) \\
\top & (t = \texttt{"do"}) \\
\top & (t = \texttt{"of"}) \\
\bot & (\text{otherwise})
\end{cases}
$$

15.3.3 PEG with Layout Tokens

$$
\begin{array}{rcl}
\mathit{module} & ::= & \texttt{"module"}\ \mathit{modid}\ \mathit{exports}^{?}\ \texttt{"where"}\ \mathit{body} \\
 & | & \mathit{body} \\
\mathit{body} & ::= & \mathit{expbo}\ \mathit{bodyinl}\ \mathit{expbc} \\
 & | & \mathit{impbo}\ \mathit{bodyinl}\ \mathit{impbc} \\
\mathit{bodyinl} & ::= & \mathit{impdecls}\ \mathit{semi}^{+}\ \mathit{topdecls} \\
 & | & \mathit{impdecls} \\
 & | & \mathit{topdecls}
\end{array}
$$

```
impdecls  ::=  semi* (impdecl semi+)* impdecl
 exports  ::=  "(" (export ",")* export? ")"
  export  ::=  qvar
            |  qtycon ("(" (".." | (cname ",")* cname)? ")")?
            |  "module" modid
 impdecl  ::=  "import" "qualified"? modid ("as" modid)? impspec?
 impspec  ::=  "(" (import ",")* import? ")"
            |  "hiding" "(" (import ",")* import? ")"
  import  ::=  var
            |  tycon ("(" (".." | (cname ",")* cname)? ")")?
   cname  ::=  var | con
```

```
topdecls  ::=  (topdecl semi)* topdecl |
 topdecl  ::=  "type" simpletype "=" type
            |  "data" (context "=>")? simpletype ("=" constrs)? deriving?
            |  "newtype" (context "=>")? simpletype "=" newconstr deriving?
            |  "class" (scontext "=>")? tycon tyvar ("where" cdecls)?
            |  "instance" (scontext "=>")? qtycon inst ("where" idecls)?
            |  "default" "(" ((type ",")* type)? ")"
            |  "foreign" fdecl
            |  decl
```

```
     decls  ::=  expbo declsinl expbc
              |  impbo declsinl impbc
  declsinl  ::=  (decl semi)* decl |
      decl  ::=  (funlhs | pat) rhs
              |  gendecl
    cdecls  ::=  expbo cdeclsinl expbc
              |  impbo cdeclsinl impbc
 cdeclsinl  ::=  (cdecl semi)* cdecl |
     cdecl  ::=  (funlhs | var) rhs
              |  gendecl
    idecls  ::=  expbo ideclsinl expbc
              |  impbo ideclsinl impbc
 ideclsinl  ::=  (idecl semi)* idecl |
     idecl  ::=  (funlhs | var) rhs
              |
   gendecl  ::=  vars "::" (context "=>")? type
              |  fixity integer? ops
              |
       ops  ::=  (op ",")* op
      vars  ::=  (var ",")* var
    fixity  ::=  "infixl" | "infixr" | "infix"
```

```
  type  ::=  btype ("->" type)?
 btype  ::=  btype? atype
 atype  ::=  gtycon
          |  tyvar
          |  "(" (type ",")+ type ")"
          |  "[" type "]"
          |  "(" type ")"
gtycon  ::=  qtycon
          |  "(" ")"
          |  "[" "]"
          |  "(" "->" ")"
          |  "(" ","+ ")"
```

context ::= *class*
| "(" ((*class* ",")* *class*)$^?$ ")"
class ::= *qtycon tyvar*
| *qtycon* "(" *tyvar atype*$^+$ ")"
scontext ::= *simpleclass*
| "(" ((*simpleclass* ",")* *simpleclass*)$^?$ ")"
simpleclass ::= *qtycon tyvar*

simpletype ::= *tycon tyvar**
constrs ::= (*constr* "|")* *constr*
constr ::= *con expbo* ((*fielddecl* ",")* *fielddecl*)$^?$ *expbc*
| (*btype* | "!" *atype*) *conop* (*btype* | "!" *atype*)
| *con* ("!"$^?$ *atype*)*
newconstr ::= *con expbo var* "::" *type expbc*
| *con atype*
fielddecl ::= *vars* "::" (*type* | "!" *atype*)
deriving ::= "deriving" *dclass*
| "deriving" "(" ((*dclass* ",")* *dclass*)$^?$ ")"
dclass ::= *qtycon*
inst ::= *gtycon*
| "(" *gtycon tyvar** ")"
| "(" (*tyvar* ",")$^+$ *tyvar* ")"
| "[" *tyvar* "]"
| "(" *tyvar* "->" *tyvar* ")"

fdecl ::= "import" *callconv safety*$^?$ *impent var* "::" *ftype*
| "export" *callconv expent var* "::" *ftype*
callconv ::= "ccall" | "stdcall" | "cplusplus" | "jvm" | "dotnet"
| (system-specific calling conventions)
impent ::= *string*$^?$
expent ::= *string*$^?$
safety ::= "unsafe" | "safe"
ftype ::= *fatype* "->" *ftype*
| *frtype*
frtype ::= *fatype*
| "(" ")"
fatype ::= *qtycon atype**

funlhs ::= *var apat*$^+$
| *pat varop pat*
| "(" *funlhs* ")" *apat*$^+$
rhs ::= "=" *exp* ("where" *decls*)$^?$
| *gdrhs* ("where" *decls*)$^?$
gdrhs ::= *guards* "=" *exp gdrhs*$^?$
guards ::= "|" (*guard* ",")* *guard* |
guard ::= *pat* "<-" *infixexp*
| "let" *decls*
| *infixexp*

```
exp      ::= infixexp "::" (context "=>")? type
           | infixexp
infixexp ::= "-" infixexp
           | lexp qop infixexp
           | lexp
lexp     ::= "\\" apat+ "->" exp
           | "let" decls "in" exp
           | "if" exp semi? "then" exp semi? "else" exp
           | "case" exp "of" casealts
           | "do" dostmts
           | fexp
fexp     ::= aexp+
aexp     ::= qcon expbo ((fbind ",")* fbind)? expbc
           | aexp2 (expbo ((fbind ",")* fbind)? expbc)*
           | qvar
aexp2    ::= literal
           | "(" exp ")"
           | "(" (exp ",")+ exp ")"
           | "[" (exp ",")* exp "]"
           | "[" exp ("," exp)? ".." exp? "]"
           | "[" exp "|" (qual ",")* qual "]"
           | "(" infixexp qop ")"
           | "(" !("-" infixexp) qop infixexp ")"
           | gcon
```

```
qual     ::= pat "<-" exp
           | "let" decls
           | exp
casealts ::= expbo alts expbc
           | impbo alts impbc
alts     ::= (alt semi)* alt
alt      ::= pat "->" exp ("where" decls)?
           | pat gdpat ("where" decls)?
           |
gdpat    ::= guards "->" exp gdpat?
dostmts  ::= expbo stmts expbc
           | impbo stmts impbc
stmts    ::= stmt* exp semi?
stmt     ::= exp semi
           | pat "<-" exp semi
           | "let" decls semi
           | semi
fbind    ::= qvar "=" exp
```

```
pat      ::= lpat qconop pat
           | lpat
lpat     ::= "-" (integer | float)
           | gcon apat+
           | apat
apat     ::= var ("@" apat)?
           | literal
           | "_"
           | "(" pat ")"
           | "(" (pat ",")+ pat ")"
           | "[" (pat ",")* pat "]"
           | "~" apat
           | qcon expbo ((fpat ",")* fpat)? expbc
           | gcon
fpat     ::= qvar "=" pat
```

$$
\begin{array}{rcl}
\mathit{gcon} & ::= & \texttt{"("}\ \texttt{")"} \\
& | & \texttt{"["}\ \texttt{"]"} \\
& | & \texttt{"("}\ \texttt{","}^{+}\ \texttt{")"} \\
& | & \mathit{qcon} \\
\mathit{var} & ::= & \mathit{varid} \mid \texttt{"("}\ \mathit{varsym}\ \texttt{")"} \\
\mathit{qvar} & ::= & \mathit{qvarid} \mid \texttt{"("}\ \mathit{qvarsym}\ \texttt{")"} \\
\mathit{con} & ::= & \mathit{conid} \mid \texttt{"("}\ \mathit{consym}\ \texttt{")"} \\
\mathit{qcon} & ::= & \mathit{qconid} \mid \texttt{"("}\ \mathit{gconsym}\ \texttt{")"} \\
\mathit{varop} & ::= & \mathit{varsym} \mid \texttt{"`"}\ \mathit{varid}\ \texttt{"`"} \\
\mathit{qvarop} & ::= & \mathit{qvarsym} \mid \texttt{"`"}\ \mathit{qvarid}\ \texttt{"`"} \\
\mathit{conop} & ::= & \mathit{consym} \mid \texttt{"`"}\ \mathit{conid}\ \texttt{"`"} \\
\mathit{qconop} & ::= & \mathit{gconsym} \mid \texttt{"`"}\ \mathit{qconid}\ \texttt{"`"} \\
\mathit{op} & ::= & \mathit{varop} \mid \mathit{conop} \\
\mathit{qop} & ::= & \mathit{qvarop} \mid \mathit{qconop} \\
\mathit{gconsym} & ::= & \texttt{":"} \mid \mathit{qconsym} \\
\mathit{tyvar} & ::= & \mathit{varid} \\
\mathit{tycon} & ::= & \mathit{conid} \\
\mathit{qtycon} & ::= & \mathit{qconid}
\end{array}
$$

$$
\begin{array}{rclll}
\mathit{expbo} & ::= & [l] & \texttt{"\{"} & [0 : l] \\
\mathit{expbc} & ::= & [0 : l] & \texttt{"\}"} & [l] \\
\mathit{impbo} & ::= & [m : l] & \{n\} & [n : m : l \mid n > m] \\
& ::= & [m : l] & \{n\} & [(n+1) : m : l \mid n \leq m] \\
& | & [\epsilon] & \{n\} & [n : \epsilon \mid n > 0] \\
\mathit{impbc} & ::= & [m : l] & \epsilon & [l \mid m > 0] \\
\mathit{semi} & ::= & & \texttt{";"} & \\
& | & [m : l] & \langle n \rangle & [m : l \mid m = n]
\end{array}
$$

$$
\text{skip} \quad ::= \quad [m : l] \quad \langle n \rangle \quad [m : l \mid m < n]
$$

15.4 WIP: A Memory Optimization for PEG with Cut Operations

[MMY08][MMY10]

15.5 WIP: SRB: An Abstract Machine of PEG

第 16 章

Analysis and Optimizations

第 17 章

Meta-Programming and Multi-Stage Programming

第 18 章

Generic Programming

第 19 章

Advanced Calculus

第 20 章

Strik: A Language for Practical Programming

20.1 WIP: Implementation Note of PEG Parser

Normalizing

$$
\begin{array}{rcll}
e_{\mathrm{RHS}} & ::= & e_1 \,/\, \cdots \,/\, e_n \,/\, \epsilon & (n \in \mathbb{N}) \\
& | & e_1 \,/\, \cdots \,/\, e_n & (n \in \mathbb{N}_{\geq 1}) \\
e & ::= & !(u_1 \cdots u_n) & (n \in \mathbb{N}_{\geq 1}) \\
& | & \&(u_1 \cdots u_n) & (n \in \mathbb{N}_{\geq 1}) \\
& | & u_1 \cdots u_n & (n \in \mathbb{N}_{\geq 1}) \\
u & ::= & \sigma & \\
& | & A &
\end{array}
$$

$$
\begin{aligned}
\mathrm{norm}(N, []) &= (N, \emptyset) \\
\mathrm{norm}(N, [A \leftarrow e] + X) &= (N_2, \{A \leftarrow \mathrm{alt}(a)\} \cup X_1 \cup X_2) \\
&\qquad (\mathrm{norm}(N, e) = (a, N_1, X_1), \mathrm{norm}(N_1, X) = (N_2, X_2))
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{norm}(N, \epsilon) &= ([\epsilon], N, \emptyset) \\
\mathrm{norm}(N, \sigma) &= ([\sigma], N, \emptyset) \\
\mathrm{norm}(N, A) &= ([A], N, \emptyset) \\
\mathrm{norm}(N, e_1 e_2) &= (\mathrm{seq}(a_1, a_2), N_2, X_1 \cup X_2) && (\mathrm{norm}(N, e_1) = (a_1, N_1, X_1), \mathrm{norm}(N_1, e_2) = (a_2, N_2, X_2)) \\
\mathrm{norm}(N, e_1/e_2) &= (a_1 + a_2, N_2, X_1 \cup X_2) && (\mathrm{norm}(N, e_1) = (a_1, N_1, X_1), \mathrm{norm}(N_1, e_2) = (a_2, N_2, X_2)) \\
\mathrm{norm}(N, e^*) &= ([M], N' \uplus \{M\}, X \cup \{M \leftarrow AM/\epsilon\}) && (\mathrm{norm}(N \uplus \{A\}, [A \leftarrow e]) = (N', X)) \\
\mathrm{norm}(N, \&\, e) &= ([M], N' \uplus \{M\}, X \cup \{M \leftarrow \&\, A\}) && (\mathrm{norm}(N + \{A\}, [A \leftarrow e]) = (N', X)) \\
\mathrm{norm}(N, !\, e) &= ([M], N' \uplus \{M\}, X \cup \{M \leftarrow !\, A\}) && (\mathrm{norm}(N + \{A\}, [A \leftarrow e]) = (N', X))
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{seq}(a_1, a_2) &= [e_1 e_2 \mid e_1 \leftarrow a_1, e_2 \leftarrow a_2] \\
\mathrm{alt}([e_1, \ldots, e_n]) &= e_1 / \cdots / e_m && (\forall i < m.\, e_i \neq \epsilon, e_m = \epsilon) \\
\mathrm{alt}([e_1, \ldots, e_n]) &= e_1 / \cdots / e_n && (\forall i.\, e_i \neq \epsilon)
\end{aligned}
$$

$$
\begin{gathered}
\mathrm{norm}((\Sigma, N, R, e_0)) = (\Sigma, N', R', S) \\
(R = \{A_1 \leftarrow e_1, \ldots, A_n \leftarrow e_n\}, \mathrm{norm}(N \uplus \{S\}, [S \leftarrow e_0, A_1 \leftarrow e_1, \ldots, A_n \leftarrow e_n]) = (N', R'))
\end{gathered}
$$

Machine

State:

- a rule
- current position in rule

Transition:

- σ
- EOS
- otherwise

Output:

with backpoint　バックポイントを設置し，バックポイントに戻った時の次の遷移を指定する．fail した場合一番直近の backpoint まで入力状態とスタックを戻す．reduce 時取り除かれる．

enter　非終端記号を参照する．メモ化されている場合その値を使う．それ以外の場合，reduce 時戻ってくる状態を記録し，次の状態に遷移する．

goto　次の状態に遷移する．

shift　入力を 1 つ消費し，次の状態に遷移する．

reduce　規則に沿ってスタックから要素を取り出してまとめ，メモし，スタックに新たに入れた後，enter 時に記録された状態に遷移する．

Optimization

1. unify transitions.
2. look ahead backpoints.

Example

$$
\begin{array}{rcl}
E & ::= & CA \\
 & | & \epsilon \\
A & ::= & aB \\
 & | & a \\
B & ::= & bA \\
 & | & b \\
C & ::= & !\,abab \\
 & | & \&\,ab
\end{array}
$$

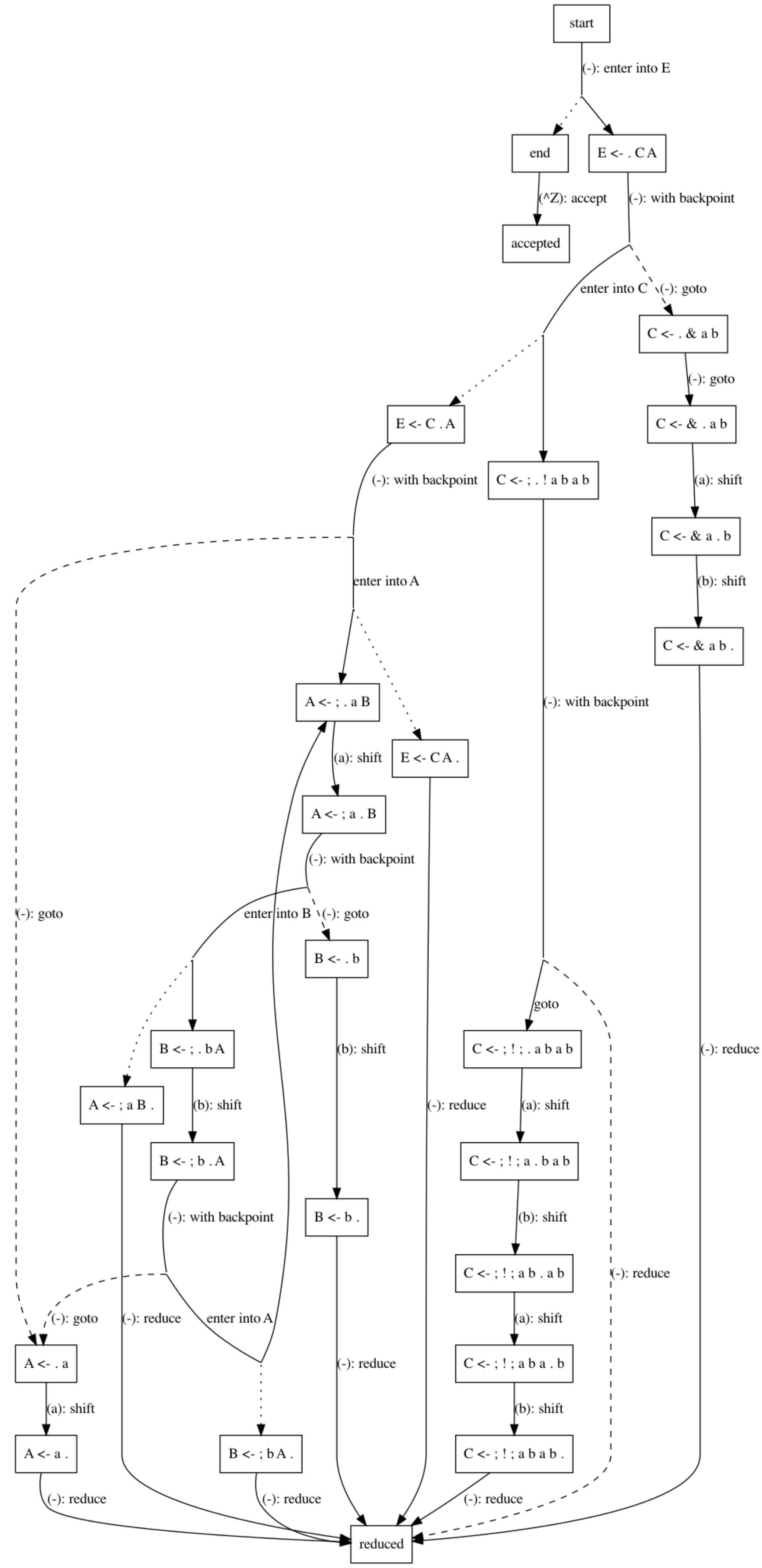
start
(-): enter into E
end
E <- . C A
(^Z): accept
(-): with backpoint
accepted
enter into C
(-): goto
C <- . & a b
(-): goto
E <- C . A
C <- & . a b
(-): with backpoint
C <- ; . ! a b a b
(a): shift
C <- & a . b
enter into A
(b): shift
C <- & a b .
A <- ; . a B
(-): with backpoint
E <- C A .
(a): shift
A <- ; a . B
(-): with backpoint
(-): goto
enter into B
(-): goto
B <- . b
goto
B <- ; . b A
(b): shift
C <- ; ! ; . a b a b
(-): reduce
A <- ; a B .
(b): shift
(-): reduce
(a): shift
B <- ; b . A
C <- ; ! ; a . b a b
(-): with backpoint
B <- b .
(b): shift
C <- ; ! ; a b . a b
(-): reduce
(-): goto
(-): reduce
enter into A
(a): shift
A <- . a
(-): reduce
C <- ; ! ; a b a . b
(a): shift
(b): shift
A <- a .
B <- ; b A .
C <- ; ! ; a b a b .
(-): reduce
(-): reduce
(-): reduce
reduced

図 20.1 状態遷移図

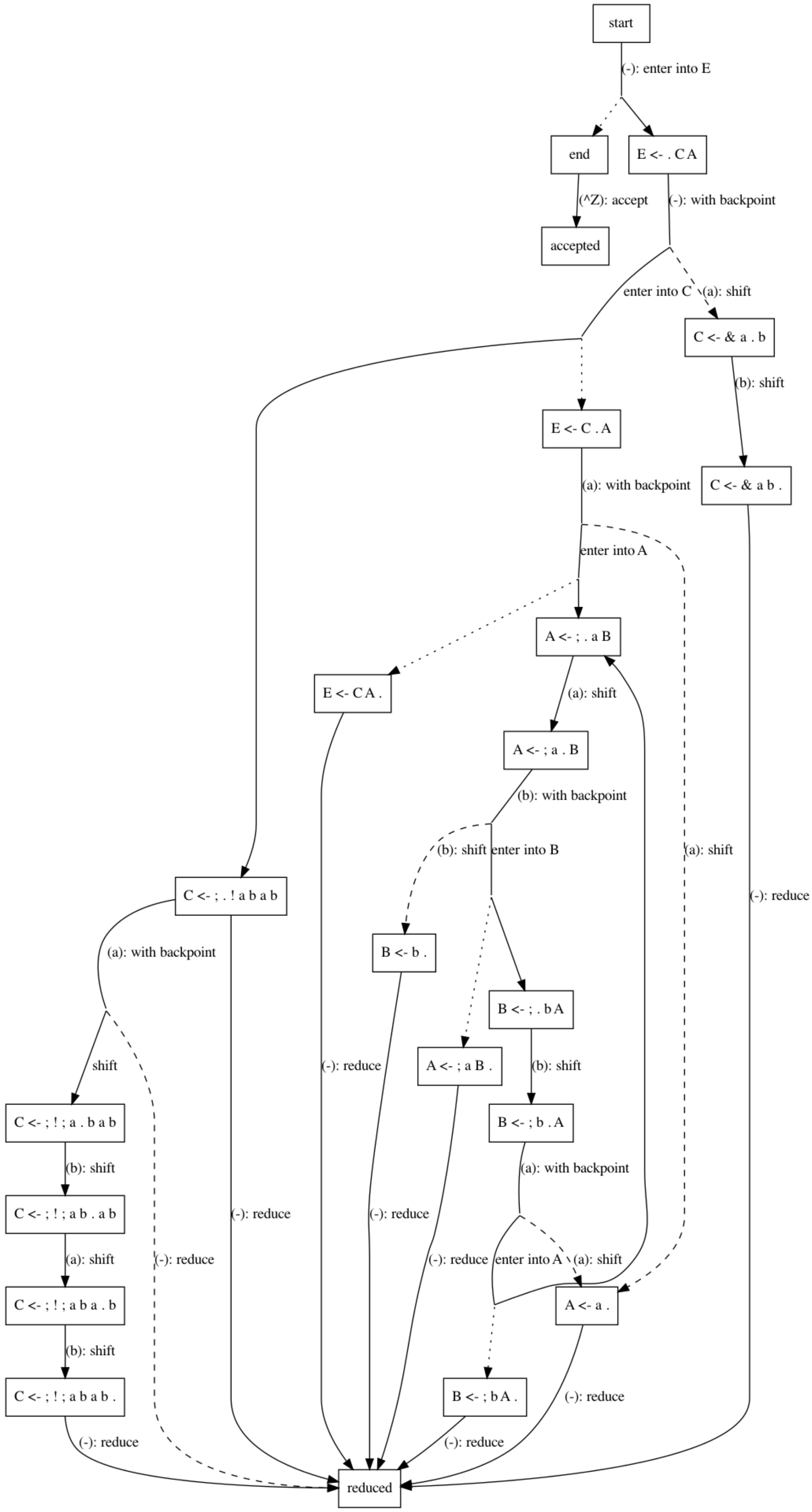

図 20.2　最適化された状態遷移図

20.2 Strik Syntax and Layout

20.2.1 Syntax

Program

$$program \ ::= \ \mathbf{program}(decl_1; \cdots ; decl_n) \quad \text{(program)}$$

Declaration

$$decl \ ::= \ \mathbf{fun}(f)(argItem_1; \cdots ; argItem_n)(expr)$$

Expression

$$
\begin{array}{rcll}
expr & ::= & x & \text{(variable)} \\
 & \mid & n & \text{(integer)} \\
 & \mid & (tupleItem_1; \cdots ; tupleItem_n) & \text{(tuple)} \\
 & \mid & \mathbf{block}(blockItem_1; \cdots ; blockItem_n) & \text{(block)} \\
 & \mid & \lambda(argItem_1; \cdots ; argItem_n)(expr) & \text{(abstraction)} \\
 & \mid & expr(tupleItem_1; \cdots ; tupleItem_n) & \text{(application)} \\
 & \mid & \mathbf{case}(expr_{1,1} \to expr_{1,2}; \cdots ; expr_{n,1} \to expr_{n,2}) & \text{(branch)} \\
 & \mid & expr : type & \text{(type annotation)} \\
argItem & ::= & x : type & \\
 & \mid & x & \\
 & \mid & \mathbf{prom}(x) : type & \\
 & \mid & \mathbf{prom}(x) & \\
tupleItem & ::= & x = expr & \\
 & \mid & \mathbf{prom}(x) = type & \\
blockItem & ::= & expr & \\
 & \mid & \mathbf{let}(x = expr) & \\
 & \mid & \mathbf{rec}(x = expr) &
\end{array}
$$

Type

$$
\begin{array}{rcll}
type & ::= & x & \text{(variable)} \\
 & \mid & n & \text{(integer)} \\
 & \mid & (typeTupleSigItem_1; \cdots ; typeTupleSigItem_n) & \text{(tuple type)} \\
 & \mid & (typeTupleItem_1; \cdots ; typeTupleItem_n) & \text{(type tuple)} \\
 & \mid & (typeTupleSigItem_1; \cdots ; typeTupleSigItem_n) \to type & \text{(function type)} \\
 & \mid & \mathbf{block}(typeBlockItem_1; \cdots ; typeBlockItem_n) & \text{(block)} \\
 & \mid & type(typeTupleItem_1; \cdots ; typeTupleItem_n) & \text{(application)} \\
 & \mid & type : type_0 & \text{(type annotation)} \\
typeTupleSigItem & ::= & x : type & \\
 & \mid & \mathbf{prom}(x) : type & \\
typeTupleItem & ::= & x = type & \\
 & \mid & \mathbf{prom}(x) = type & \\
typeBlockItem & ::= & type & \\
 & \mid & \mathbf{let}(x = type) &
\end{array}
$$

20.2.2 Layout

20.3 Strik Type System

20.3.1 Declarative

Context:

$$\begin{array}{rcll} \Gamma & ::= & \epsilon & \text{(empty)} \\ & | & x : \mathit{type} & \text{(variable)} \\ & | & x = \mathit{type} & \text{(synonym)} \\ & | & \Gamma_1; \Gamma_2 & \text{(concatenation)} \end{array}$$

Program:

$\boxed{\Gamma \vdash \mathit{program} \mid \Delta}$

$$\frac{\begin{array}{c} \mathit{decl}_1 \mid x_1 \quad \cdots \quad \mathit{decl}_n \mid x_n \\ \mathbf{fresh}(a_1, \ldots, a_n) \quad \Gamma_1 = \Gamma; x_1 : a_1; \cdots; x_n : a_n \\ \Delta_0 = \Gamma_1 \quad \Delta_0 \vdash \mathit{decl}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{decl}_n \mid \Delta_n \end{array}}{\Gamma \vdash \mathbf{program}(\mathit{decl}_1; \cdots; \mathit{decl}_n) \mid \Delta_n}$$

Declaration:

$\boxed{\mathit{decl} \mid x}$

$$\frac{}{\mathbf{fun}(f)(\mathit{argItem}_1; \cdots; \mathit{argItem}_n)(\mathit{expr}) \mid f}$$

$\boxed{\Gamma \vdash \mathit{decl} \mid \Delta}$

$$\frac{\begin{array}{c} \Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{argItem}_1 : \mathit{typeTupleSigItem}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{argItem}_n : \mathit{typeTupleSigItem}_n \mid \Delta_n \\ \Delta_n \vdash \mathit{expr} : \mathit{type} \quad \Delta = \Gamma_1; f : a; \Gamma_2; a = (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \end{array}}{\Gamma_1; f : a; \Gamma_2 \vdash \mathbf{fun}(f)(\mathit{argItem}_1; \cdots; \mathit{argItem}_n)(\mathit{expr}) \mid \Delta}$$

Expression:

$\boxed{\Gamma \vdash \mathit{expr} : \mathit{type}}$

$$\frac{\Gamma(x) = \mathit{type} \quad \Gamma \vdash \mathit{type} : \mathbf{Type}}{\Gamma \vdash x : \mathit{type}}$$

$$\frac{\Gamma \vdash \mathbf{Int} : \mathbf{Type}}{\Gamma \vdash n : \mathbf{Int}}$$

$$\frac{\Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{tupleItem}_1 : \mathit{typeTupleSigItem}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{tupleItem}_n : \mathit{typeTupleSigItem}_n \mid \Delta_n}{\Gamma \vdash (\mathit{tupleItem}_1; \cdots; \mathit{tupleItem}_n) : (\mathit{typeTupleSigItem}_1; \ldots; \mathit{typeTupleSigItem}_n)}$$

$$\frac{\Delta_0 = \Gamma \quad \mathit{type}_0 = () \quad \Delta_0 \vdash \mathit{blockItem}_1 : \mathit{type}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{blockItem}_n : \mathit{type}_n \mid \Delta_n}{\Gamma \vdash \mathbf{block}(\mathit{blockItem}_1; \cdots; \mathit{blockItem}_n) : \mathit{type}_n}$$

$$\frac{\begin{array}{c} \Delta_0 = \Gamma \\ \Delta_0 \vdash \mathit{argItem}_1 : \mathit{typeTupleSigItem}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{argItem}_n : \mathit{typeTupleSigItem}_n \mid \Delta_n \\ \Delta_n \vdash \mathit{expr} : \mathit{type} \end{array}}{\Gamma \vdash \lambda(\mathit{argItem}_1; \cdots; \mathit{argItem}_n)(\mathit{expr}) : (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type}}$$

$$\frac{\begin{array}{c} \Gamma \vdash \mathit{expr} : (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \\ \Gamma \vdash (\mathit{tupleItem}_1; \cdots; \mathit{tupleItem}_n) : (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \end{array}}{\Gamma \vdash \mathit{expr}(\mathit{tupleItem}_1; \cdots; \mathit{tupleItem}_n) : \mathit{type}}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathbf{Type} \quad \Gamma \vdash \mathit{expr}_{1,1} : \mathbf{Bool} \quad \Gamma \vdash \mathit{expr}_{1,2} : \mathit{type} \quad \cdots \quad \Gamma \vdash \mathit{expr}_{n,1} : \mathbf{Bool} \quad \Gamma \vdash \mathit{expr}_{n,2} : \mathit{type}}{\Gamma \vdash \mathbf{case}(\mathit{expr}_{1,1} \to \mathit{expr}_{1,2}; \cdots; \mathit{expr}_{n,1} \to \mathit{expr}_{n,2}) : \mathit{type}}$$

$$\frac{\Gamma \vdash \mathit{expr} : \mathit{type}}{\Gamma \vdash (\mathit{expr} : \mathit{type}) : \mathit{type}}$$

$$\frac{\Gamma \vdash \mathit{expr} : \mathit{type}_1 \quad \Gamma \vdash \mathit{type}_1 \preceq \mathit{type}_2}{\Gamma \vdash \mathit{expr} : \mathit{type}_2}$$

Tuple:

$\boxed{\Gamma \vdash \mathit{tupleItem} : \mathit{typeTupleSigItem} \mid \Delta}$

$$\frac{\Gamma \vdash \mathit{expr} : \mathit{type}}{\Gamma \vdash x = \mathit{expr} : (x : \mathit{type}) \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathit{type}_0}{\Gamma \vdash \mathbf{prom}(x) = \mathit{type} : (\mathbf{prom}(x) : \mathit{type}_0) \mid \Gamma; x : \mathit{type}_0; x = \mathit{type}}$$

Block:

$\boxed{\Gamma \vdash \mathit{blockItem} : \mathit{type} \mid \Delta}$

$$\frac{\Gamma \vdash \mathit{expr} : \mathit{type}}{\Gamma \vdash \mathit{expr} : \mathit{type} \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{expr} : \mathit{type}}{\Gamma \vdash \mathbf{let}(x = \mathit{expr}) : () \mid \Gamma; x : \mathit{type}}$$

$$\frac{\Gamma; x : \mathit{type} \vdash \mathit{expr} : \mathit{type}}{\Gamma \vdash \mathbf{rec}(x = \mathit{expr}) : () \mid \Gamma; x : \mathit{type}}$$

Argument:

$\boxed{\Gamma \vdash \mathit{argItem} : \mathit{typeTupleSigItem} \mid \Delta}$

$$\frac{\Gamma \vdash \mathit{type} : \mathbf{Type}}{\Gamma \vdash (x : \mathit{type}) : (x : \mathit{type}) \mid \Gamma; x : \mathit{type}}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathbf{Type}}{\Gamma \vdash x : (x : \mathit{type}) \mid \Gamma; x : \mathit{type}}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathit{type}_0}{\Gamma \vdash (\mathbf{prom}(x) : \mathit{type}) : (\mathbf{prom}(x) : \mathit{type}) \mid \Gamma; x : \mathit{type}}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathit{type}_0}{\Gamma \vdash \mathbf{prom}(x) : (\mathbf{prom}(x) : \mathit{type}) \mid \Gamma; x : \mathit{type}}$$

Type:

$\boxed{\Gamma \vdash \mathit{type} : \mathit{type}_0}$

$$\frac{\Gamma(x) = \mathit{type} \quad \Gamma \vdash \mathit{type} : \mathit{type}_0}{\Gamma \vdash x : \mathit{type}}$$

$$\frac{}{\Gamma \vdash \mathbf{Type} : \mathbf{Type}}$$

$$\frac{\Gamma \vdash \mathbf{Int} : \mathbf{Type}}{\Gamma \vdash n : \mathbf{Int}}$$

$$\frac{\Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{typeTupleSigItem}_1 : \mathit{type}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeTupleSigItem}_n : \mathit{type}_n \mid \Delta_n}{\Gamma \vdash (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) : \mathbf{Type}}$$

$$\frac{\Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{typeTupleItem}_1 : \mathit{typeTupleSigItem}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeTupleItem}_n : \mathit{typeTupleSigItem}_n \mid \Delta_n}{\Gamma \vdash (\mathit{typeTupleItem}_1; \cdots; \mathit{typeTupleItem}_n) : (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n)}$$

$$\frac{\begin{array}{c}\Delta_0 = \Gamma \\ \Delta_0 \vdash \mathit{typeTupleSigItem}_1 : \mathit{type}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeTupleSigItem}_n : \mathit{type}_n \mid \Delta_n \\ \Delta_n \vdash \mathit{type} : \mathbf{Type}\end{array}}{\Gamma \vdash (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \rightarrow \mathit{type} : \mathbf{Type}}$$

$$\frac{\Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{typeBlockItem}_1 : \mathit{type}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeBlockItem}_n : \mathit{type}_n \mid \Delta_n}{\Gamma \vdash \mathbf{block}(\mathit{typeBlockItem}_1; \cdots; \mathit{typeBlockItem}_n) : \mathit{type}_n}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{type} : (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type}_0 \\ \Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{typeTupleItem}_1 : \mathit{typeTupleSigItem}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeTupleItem}_n : \mathit{typeTupleSigItem}_n \mid \Delta_n\end{array}}{\Gamma \vdash \mathit{type}(\mathit{typeTupleItem}_1; \cdots; \mathit{typeTupleItem}_n) : \mathit{type}_0}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathit{type}_0}{\Gamma \vdash (\mathit{type} : \mathit{type}_0) : \mathit{type}_0}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathit{type}_1 \quad \Gamma \vdash \mathit{type}_1 \preceq \mathit{type}_2}{\Gamma \vdash \mathit{type} : \mathit{type}_2}$$

Tuple Type:

$$\boxed{\Gamma \vdash \mathit{typeTupleSigItem} : \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathbf{Type}}{\Gamma \vdash (x : \mathit{type}) : \mathbf{Type} \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathit{type}_0}{\Gamma \vdash (\mathbf{prom}(x) : \mathit{type}) : \mathit{type}_0 \mid \Gamma; x : \mathit{type}}$$

Type Tuple:

$$\boxed{\Gamma \vdash \mathit{typeTupleItem} : \mathit{typeTupleSigItem} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathit{type}_0}{\Gamma \vdash (x = \mathit{type}) : (x : \mathit{type}_0) \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathit{type}_0}{\Gamma \vdash (\mathbf{prom}(x) = \mathit{type}) : (\mathbf{prom}(x) : \mathit{type}_0) \mid \Gamma; x : \mathit{type}_0; x = \mathit{type}}$$

Type Block:

$$\boxed{\Gamma \vdash \mathit{typeBlockItem} : \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathit{type}_0}{\Gamma \vdash \mathit{type} : \mathit{type}_0 \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{type} : \mathit{type}_0}{\Gamma \vdash \mathbf{let}(x = \mathit{type}) : () \mid \Gamma; x : \mathit{type}_0; x = \mathit{type}}$$

Cast:

$$\boxed{\Gamma \vdash \mathit{type}_1 \preceq \mathit{type}_2}$$

$$\frac{\mathit{type}_1 = \mathit{type}_2}{\Gamma \vdash \mathit{type}_1 \preceq \mathit{type}_2}$$

$$\frac{\Gamma_1; x = \mathit{type}; \Gamma_2 \vdash \mathit{type}_1[x \leftarrow \mathit{type}] \preceq \mathit{type}_2[x \leftarrow \mathit{type}]}{\Gamma_1; x = \mathit{type}; \Gamma_2 \vdash \mathit{type}_1 \preceq \mathit{type}_2}$$

20.3.2 Bidirectional

Program:

$$\boxed{\Gamma \vdash \mathit{program} \mid \Delta}$$

$$\frac{\begin{array}{c}\mathit{decl}_1 \mid x_1 \quad \cdots \quad \mathit{decl}_n \mid x_n \\ \mathbf{fresh}(a_1, \ldots, a_n) \quad \Gamma_1 = \Gamma; x_1 : a_1; \cdots; x_n : a_n \\ \Delta_0 = \Gamma_1 \quad \Delta_0 \vdash \mathit{decl}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{decl}_n \mid \Delta_n\end{array}}{\Gamma \vdash \mathbf{program}(\mathit{decl}_1; \cdots; \mathit{decl}_n) \mid \Delta_n}$$

Declaration:

$\boxed{decl \mid x}$

$$\frac{}{\mathbf{fun}(f)(argItem_1; \cdots; argItem_n)(expr) \mid f}$$

$\boxed{\Gamma \vdash decl \mid \Delta}$

$$\frac{\begin{array}{c}\Delta_0 = \Gamma \quad \Delta_0 \vdash argItem_1 \Rightarrow typeTupleSigItem_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash argItem_n \Rightarrow typeTupleSigItem_n \mid \Delta_n \\ \Delta_n \vdash expr \Leftarrow type \quad \Delta = \Gamma_1; f : a; \Gamma_2; a = (typeTupleSigItem_1; \cdots; typeTupleSigItem_n) \to type\end{array}}{\Gamma_1; f : a; \Gamma_2 \vdash \mathbf{fun}(f)(argItem_1; \cdots; argItem_n)(expr) \mid \Delta}$$

Expression:

$\boxed{\Gamma \vdash expr \Rightarrow type}$ $\boxed{\Gamma \vdash expr \Leftarrow type}$

$$\frac{\Gamma(x) = type \quad \Gamma \vdash type \Leftarrow \mathbf{Type}}{\Gamma \vdash x \Rightarrow type}$$

$$\frac{\Gamma \vdash \mathbf{Int} \Leftarrow \mathbf{Type}}{\Gamma \vdash n \Rightarrow \mathbf{Int}}$$

$$\frac{\Gamma \vdash expr \Leftarrow type}{\Gamma \vdash (expr : type) \Rightarrow type}$$

$$\frac{\Gamma \vdash expr \Rightarrow type_1 \quad \Gamma \vdash type_1 \preceq type_2}{\Gamma \vdash expr \Leftarrow type_2}$$

$$\frac{\Delta_0 = \Gamma \quad \Delta_0 \vdash tupleItem_1 \Rightarrow typeTupleSigItem_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash tupleItem_n \Rightarrow typeTupleSigItem_n \mid \Delta_n}{\Gamma \vdash (tupleItem_1; \cdots; tupleItem_n) \Rightarrow (typeTupleSigItem_1; \ldots; typeTupleSigItem_n)}$$

$$\frac{\Delta_0 = \Gamma \quad type_0 = () \quad \Delta_0 \vdash blockItem_1 \Rightarrow type_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash blockItem_n \Rightarrow type_n \mid \Delta_n}{\Gamma \vdash \mathbf{block}(blockItem_1; \cdots; blockItem_n) \Rightarrow type_n}$$

$$\frac{\begin{array}{c}\Delta_0 = \Gamma \\ \Delta_0 \vdash argItem_1 \Rightarrow typeTupleSigItem_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash argItem_n \Rightarrow typeTupleSigItem_n \mid \Delta_n \\ \Delta_n \vdash expr \Leftarrow type\end{array}}{\Gamma \vdash \lambda(argItem_1; \cdots; argItem_n)(expr) \Rightarrow (typeTupleSigItem_1; \cdots; typeTupleSigItem_n) \to type}$$

$$\frac{\begin{array}{c}\Gamma \vdash expr \Rightarrow (typeTupleSigItem_1; \cdots; typeTupleSigItem_n) \to type \\ \Gamma \vdash (tupleItem_1; \cdots; tupleItem_n) \Leftarrow (typeTupleSigItem_1; \cdots; typeTupleSigItem_n)\end{array}}{\Gamma \vdash expr(tupleItem_1; \cdots; tupleItem_n) \Rightarrow type}$$

$$\frac{\begin{array}{c}n > 0 \quad \Gamma \vdash expr_{1,1} \Leftarrow \mathbf{Bool} \quad \Gamma \vdash expr_{1,2} \Rightarrow type \\ \Gamma \vdash expr_{2,1} \Leftarrow \mathbf{Bool} \quad \Gamma \vdash expr_{2,2} \Leftarrow type \quad \cdots \quad \Gamma \vdash expr_{n,1} \Leftarrow \mathbf{Bool} \quad \Gamma \vdash expr_{n,2} \Leftarrow type\end{array}}{\Gamma \vdash \mathbf{case}(expr_{1,1} \to expr_{1,2}; \cdots; expr_{n,1} \to expr_{n,2}) \Rightarrow type}$$

$$\frac{\Gamma \vdash type \Leftarrow \mathbf{Type}}{\Gamma \vdash \mathbf{case}() \Rightarrow type}$$

Tuple:

$\boxed{\Gamma \vdash tupleItem \Rightarrow typeTupleSigItem \mid \Delta}$

$$\frac{\Gamma \vdash expr \Rightarrow type}{\Gamma \vdash x = expr \Rightarrow (x : type) \mid \Gamma}$$

$$\frac{\Gamma \vdash type \Rightarrow type_0}{\Gamma \vdash \mathbf{prom}(x) = type \Rightarrow (\mathbf{prom}(x) : type_0) \mid \Gamma; x : type_0; x = type}$$

Block:

$\boxed{\Gamma \vdash blockItem \Rightarrow type \mid \Delta}$

$$\frac{\Gamma \vdash \mathit{expr} \Rightarrow \mathit{type}}{\Gamma \vdash \mathit{expr} \Rightarrow \mathit{type} \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{expr} \Rightarrow \mathit{type}}{\Gamma \vdash \mathbf{let}(x = \mathit{expr}) \Rightarrow () \mid \Gamma; x : \mathit{type}}$$

$$\frac{\Gamma; x : \mathit{type} \vdash \mathit{expr} \Rightarrow \mathit{type}}{\Gamma \vdash \mathbf{rec}(x = \mathit{expr}) \Rightarrow () \mid \Gamma; x : \mathit{type}}$$

Argument:

$$\boxed{\Gamma \vdash \mathit{argItem} \Rightarrow \mathit{typeTupleSigItem} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type} \Leftarrow \mathbf{Type}}{\Gamma \vdash (x : \mathit{type}) \Rightarrow (x : \mathit{type}) \mid \Gamma; x : \mathit{type}}$$

$$\frac{\Gamma \vdash \mathit{type} \Leftarrow \mathbf{Type}}{\Gamma \vdash x \Rightarrow (x : \mathit{type}) \mid \Gamma; x : \mathit{type}}$$

$$\frac{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0}{\Gamma \vdash (\mathbf{prom}(x) : \mathit{type}) \Rightarrow (\mathbf{prom}(x) : \mathit{type}) \mid \Gamma; x : \mathit{type}}$$

$$\frac{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0}{\Gamma \vdash \mathbf{prom}(x) \Rightarrow (\mathbf{prom}(x) : \mathit{type}) \mid \Gamma; x : \mathit{type}}$$

Type:

$$\boxed{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0} \boxed{\Gamma \vdash \mathit{type} \Leftarrow \mathit{type}_0}$$

$$\frac{\Gamma(x) = \mathit{type} \quad \Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0}{\Gamma \vdash x \Rightarrow \mathit{type}}$$

$$\frac{}{\Gamma \vdash \mathbf{Type} \Rightarrow \mathbf{Type}}$$

$$\frac{\Gamma \vdash \mathbf{Int} \Leftarrow \mathbf{Type}}{\Gamma \vdash n \Rightarrow \mathbf{Int}}$$

$$\frac{\Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{typeTupleSigItem}_1 \Rightarrow \mathit{type}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeTupleSigItem}_n \Rightarrow \mathit{type}_n \mid \Delta_n}{\Gamma \vdash (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \Rightarrow \mathbf{Type}}$$

$$\frac{\Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{typeTupleItem}_1 \Rightarrow \mathit{typeTupleSigItem}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeTupleItem}_n \Rightarrow \mathit{typeTupleSigItem}_n \mid \Delta_n}{\Gamma \vdash (\mathit{typeTupleItem}_1; \cdots; \mathit{typeTupleItem}_n) \Rightarrow (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n)}$$

$$\frac{\begin{array}{c}\Delta_0 = \Gamma \\ \Delta_0 \vdash \mathit{typeTupleSigItem}_1 \Rightarrow \mathit{type}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeTupleSigItem}_n \Rightarrow \mathit{type}_n \mid \Delta_n \\ \Delta_n \vdash \mathit{type} \Leftarrow \mathbf{Type}\end{array}}{\Gamma \vdash (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \Rightarrow \mathbf{Type}}$$

$$\frac{\Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{typeBlockItem}_1 \Rightarrow \mathit{type}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeBlockItem}_n \Rightarrow \mathit{type}_n \mid \Delta_n}{\Gamma \vdash \mathbf{block}(\mathit{typeBlockItem}_1; \cdots; \mathit{typeBlockItem}_n) \Rightarrow \mathit{type}_n}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{type} \Rightarrow (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type}_0 \\ \Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{typeTupleItem}_1 \Leftarrow \mathit{typeTupleSigItem}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeTupleItem}_n \Leftarrow \mathit{typeTupleSigItem}_n \mid \Delta_n\end{array}}{\Gamma \vdash \mathit{type}(\mathit{typeTupleItem}_1; \cdots; \mathit{typeTupleItem}_n) \Rightarrow \mathit{type}_0}$$

$$\frac{\Gamma \vdash \mathit{type} \Leftarrow \mathit{type}_0}{\Gamma \vdash (\mathit{type} : \mathit{type}_0) \Rightarrow \mathit{type}_0}$$

$$\frac{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_1 \quad \Gamma \vdash \mathit{type}_1 \preceq \mathit{type}_2}{\Gamma \vdash \mathit{type} \Leftarrow \mathit{type}_2}$$

Tuple Type:

$$\boxed{\Gamma \vdash \mathit{typeTupleSigItem} \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type} \Leftarrow \mathbf{Type}}{\Gamma \vdash (x : \mathit{type}) \Rightarrow \mathbf{Type} \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0}{\Gamma \vdash (\textbf{prom}(x) : \mathit{type}) \Rightarrow \mathit{type}_0 \mid \Gamma; x : \mathit{type}}$$

Type Tuple:

$\boxed{\Gamma \vdash \mathit{typeTupleItem} \Rightarrow \mathit{typeTupleSigItem} \mid \Delta}$ $\boxed{\Gamma \vdash \mathit{typeTupleItem} \Leftarrow \mathit{typeTupleSigItem} \mid \Delta}$

$$\frac{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0}{\Gamma \vdash (x = \mathit{type}) \Rightarrow (x : \mathit{type}_0) \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{type} \Leftarrow \mathit{type}_0}{\Gamma \vdash (x = \mathit{type}) \Leftarrow (x : \mathit{type}_0) \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0}{\Gamma \vdash (\textbf{prom}(x) = \mathit{type}) \Rightarrow (\textbf{prom}(x) : \mathit{type}_0) \mid \Gamma; x : \mathit{type}_0; x = \mathit{type}}$$

$$\frac{\Gamma \vdash \mathit{type} \Leftarrow \mathit{type}_0}{\Gamma \vdash (\textbf{prom}(x) = \mathit{type}) \Rightarrow (\textbf{prom}(x) : \mathit{type}_0) \mid \Gamma; x : \mathit{type}_0; x = \mathit{type}}$$

Type Block:

$\boxed{\Gamma \vdash \mathit{typeBlockItem} \Rightarrow \mathit{type} \mid \Delta}$

$$\frac{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0}{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0 \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0}{\Gamma \vdash \textbf{let}(x = \mathit{type}) \Rightarrow () \mid \Gamma; x : \mathit{type}_0; x = \mathit{type}}$$

20.3.3 Algorithmic Bidirectional

Context:

$$\begin{array}{rcll} \Gamma & ::= & \epsilon & \text{(empty)} \\ & \mid & x : \mathit{type} & \text{(variable)} \\ & \mid & x = \mathit{type} & \text{(synonym)} \\ & \mid & \hat{x} : \mathit{type} & \text{(generated variable)} \\ & \mid & \hat{x} = \mathit{type} & \text{(equation)} \\ & \mid & \Gamma_1; \Gamma_2 & \text{(concatenation)} \end{array}$$

Program:

TODO

Declaration:

TODO

Expression:

$\boxed{\Gamma \vdash \mathit{expr} \Rightarrow \mathit{type} \mid \Delta}$ $\boxed{\Gamma \vdash \mathit{expr} \Leftarrow \mathit{type} \mid \Delta}$

$$\frac{\Gamma(x) = \mathit{type} \quad \Gamma \vdash \mathit{type} \Leftarrow \textbf{Type} \mid \Delta}{\Gamma \vdash x \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{expr} \Leftarrow \mathit{type} \mid \Delta}{\Gamma \vdash (\mathit{expr} : \mathit{type}) \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{expr} \Rightarrow \mathit{type}_1 \mid \Delta_1 \quad \Delta_1 \vdash \mathit{type}_1 \preceq \mathit{type}_2 \mid \Delta}{\Gamma \vdash \mathit{expr} \Leftarrow \mathit{type}_2 \mid \Delta}$$

$$\frac{\Gamma \vdash \textbf{Int} \Leftarrow \textbf{Type} \mid \Delta}{\Gamma \vdash n \Rightarrow \textbf{Int} \mid \Delta}$$

$$\frac{\Gamma \vdash (\circ; \mathit{tupleItem}_1; \cdots; \mathit{tupleItem}_n) \Rightarrow (\circ; \mathit{typeTupleSigItem}_1; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta}{\Gamma \vdash (\mathit{tupleItem}_1; \cdots; \mathit{tupleItem}_n) \Rightarrow (\mathit{typeTupleSigItem}_1; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta}$$

$$\frac{\Gamma \vdash \textbf{block}(\circ; \mathit{blockItem}_1; \cdots; \mathit{blockItem}_n); () \Rightarrow \mathit{type} \mid \Delta}{\Gamma \vdash \textbf{block}(\mathit{blockItem}_1; \cdots; \mathit{blockItem}_n) \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \lambda(\circ; \mathit{argItem}_1; \cdots; \mathit{argItem}_n)(\mathit{expr}) \Rightarrow (\circ; \mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta}{\Gamma \vdash \lambda(\mathit{argItem}_1; \cdots; \mathit{argItem}_n)(\mathit{expr}) \Rightarrow (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{expr} \Rightarrow (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta_1 \\ \Delta_1 \vdash (\mathit{tupleItem}_1; \cdots; \mathit{tupleItem}_n) \Leftarrow (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \mid \Delta\end{array}}{\Gamma \vdash \mathit{expr}(\mathit{tupleItem}_1; \cdots; \mathit{tupleItem}_n) \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{\begin{array}{c} n > 0 \quad \Gamma \vdash \mathit{expr}_{1,1} \Leftarrow \textbf{Bool} \mid \Delta_{1,1} \quad \Delta_{1,1} \vdash \mathit{expr}_{1,2} \Rightarrow \mathit{type} \mid \Delta_{1,2} \\ \Delta_{1,2} \vdash \mathit{expr}_{2,1} \Leftarrow \textbf{Bool} \mid \Delta_{2,1} \quad \Delta_{2,1} \vdash \mathit{expr}_{2,2} \Leftarrow \mathit{type} \mid \Delta_{2,2} \\ \vdots \\ \Delta_{n-1,2} \vdash \mathit{expr}_{n,1} \Leftarrow \textbf{Bool} \mid \Delta_{n,1} \quad \Delta_{n,1} \vdash \mathit{expr}_{n,2} \Leftarrow \mathit{type} \mid \Delta_{n,2} \end{array}}{\Gamma \vdash \textbf{case}(\mathit{expr}_{1,1} \to \mathit{expr}_{1,2}; \cdots; \mathit{expr}_{n,1} \to \mathit{expr}_{n,2}) \Rightarrow \mathit{type} \mid \Delta_{n,2}}$$

$$\frac{}{\Gamma \vdash \textbf{case}() \Rightarrow \hat{\alpha} \mid \Gamma; \hat{\alpha} : \textbf{Type}}$$

Tuple:

$$\boxed{\Gamma \vdash (\circ; \mathit{tupleItem}_1; \cdots; \mathit{tupleItem}_n) \Rightarrow (\circ; \mathit{typeTupleSigItem}_1; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta}$$

$$\frac{}{\Gamma \vdash (\circ) \Rightarrow (\circ) \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{expr} \Rightarrow \mathit{type} \mid \Delta_1 \quad \Delta_1 \vdash (\circ; \mathit{tupleItem}_2; \cdots; \mathit{tupleItem}_n) \Rightarrow (\circ, \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta}{\Gamma \vdash (\circ; x = \mathit{expr}; \mathit{tupleItem}_2; \cdots; \mathit{tupleItem}_n) \Rightarrow (\circ; x : \mathit{type}; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0 \mid \Delta_1 \\ \Delta_1; x; x = \mathit{type} \vdash (\circ; \mathit{tupleItem}_2; \cdots; \mathit{tupleItem}_n) \Rightarrow (\circ, \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta; x; \Delta_2\end{array}}{\Gamma \vdash (\circ; \textbf{prom}(x) = \mathit{type}; \mathit{tupleItem}_2; \cdots; \mathit{tupleItem}_n) \Rightarrow (\circ; x : \mathit{type}; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta}$$

Block:

$$\boxed{\Gamma \vdash \textbf{block}(\circ; \mathit{blockItem}_1; \cdots; \mathit{blockItem}_n); \mathit{type}_0 \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{}{\Gamma \vdash \textbf{block}(\circ); \mathit{type} \Rightarrow \mathit{type} \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{expr}_1 \Rightarrow \mathit{type}_1 \mid \Delta_1 \quad \Delta_1 \vdash \textbf{block}(\circ; \mathit{blockItem}_2; \cdots; \mathit{blockItem}_n); \mathit{type}_1 \Rightarrow \mathit{type} \mid \Delta}{\Gamma \vdash \textbf{block}(\circ; \mathit{expr}_1; \mathit{blockItem}_2; \cdots; \mathit{blockItem}_n); \mathit{type}_0 \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{expr}_1 \Rightarrow \mathit{type}_1 \mid \Delta_1 \quad \Delta_1; x_1 : \mathit{type}_1 \vdash \textbf{block}(\circ; \mathit{blockItem}_2; \cdots; \mathit{blockItem}_n); () \Rightarrow \mathit{type} \mid \Delta; x_1 : \mathit{type}_1; \Delta_2}{\Gamma \vdash \textbf{block}(\circ; \textbf{let}(x_1 = \mathit{expr}_1); \mathit{blockItem}_2; \cdots; \mathit{blockItem}_n); \mathit{type}_0 \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{\Gamma; \hat{\alpha}_1; x_1 : \hat{\alpha}_1 \vdash \mathit{expr}_1 \Rightarrow \mathit{type}_1 \mid \Delta_1 \quad \Delta_1 \vdash \textbf{block}(\circ; \mathit{blockItem}_2; \cdots; \mathit{blockItem}_n); () \Rightarrow \mathit{type} \mid \Delta; x_1 : \hat{\alpha}_1; \Delta_2}{\Gamma \vdash \textbf{block}(\circ; \textbf{rec}(x_1 = \mathit{expr}_1); \mathit{blockItem}_2; \cdots; \mathit{blockItem}_n); \mathit{type}_0 \Rightarrow \mathit{type} \mid \Delta}$$

Abstraction:

$$\boxed{\Gamma \vdash \lambda(\circ; \mathit{argItem}_1; \cdots; \mathit{argItem}_n)(\mathit{expr}) \Rightarrow (\circ; \mathit{typeTupleSigItem}_1; \ldots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{expr} \Rightarrow \mathit{type} \mid \Delta}{\Gamma \vdash \lambda(\circ)(\mathit{expr}) \Rightarrow (\circ) \to \mathit{type} \mid \Delta}$$

$$\frac{\begin{array}{l}\Gamma; x_1 : \mathit{type}_1 \vdash \lambda(\circ; \mathit{argItem}_2; \cdots; \mathit{argItem}_n)(\mathit{expr}) \\ \qquad \Rightarrow (\circ; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta; x_1 : \mathit{type}_1; \Delta_1\end{array}}{\begin{array}{l}\Gamma \vdash \lambda(\circ; x_1 : \mathit{type}_1; \mathit{argItem}_2; \cdots; \mathit{argItem}_n)(\mathit{expr}) \\ \qquad \Rightarrow (\circ; x_1 : \mathit{type}_1; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta\end{array}}$$

$$\frac{\begin{array}{l}\Gamma; \hat{\alpha}_1; x_1 : \hat{\alpha}_1 \vdash \lambda(\circ; \mathit{argItem}_2; \cdots; \mathit{argItem}_n)(\mathit{expr}) \\ \qquad \Rightarrow (\circ; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta; x_1 : \hat{\alpha}_1; \Delta_1\end{array}}{\begin{array}{l}\Gamma \vdash \lambda(\circ; x_1; \mathit{argItem}_2; \cdots; \mathit{argItem}_n)(\mathit{expr}) \\ \qquad \Rightarrow (\circ; x_1 : \hat{\alpha}_1; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta\end{array}}$$

$$\frac{\begin{array}{l}\Gamma; x_1 : \mathit{type}_1 \vdash \lambda(\circ; \mathit{argItem}_2; \cdots; \mathit{argItem}_n)(\mathit{expr})\\ \quad \Rightarrow (\circ; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta; x_1 : \mathit{type}_1; \Delta_1\end{array}}{\begin{array}{l}\Gamma \vdash \lambda(\circ; \mathbf{prom}(x_1) : \mathit{type}_1; \mathit{argItem}_2; \cdots; \mathit{argItem}_n)(\mathit{expr})\\ \quad \Rightarrow (\circ; \mathbf{prom}(x_1) : \mathit{type}_1; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta\end{array}}$$

$$\frac{\begin{array}{l}\Gamma; \hat{\alpha}_1; x_1 : \hat{\alpha}_1 \vdash \lambda(\circ; \mathit{argItem}_2; \cdots; \mathit{argItem}_n)(\mathit{expr})\\ \quad \Rightarrow (\circ; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta; x_1 : \hat{\alpha}_1; \Delta_1\end{array}}{\begin{array}{l}\Gamma \vdash \lambda(\circ; \mathbf{prom}(x_1); \mathit{argItem}_2; \cdots; \mathit{argItem}_n)(\mathit{expr})\\ \quad \Rightarrow (\circ; \mathbf{prom}(x_1) : \hat{\alpha}_1; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta\end{array}}$$

Type:

$\boxed{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0 \mid \Delta}$ $\boxed{\Gamma \vdash \mathit{type} \Leftarrow \mathit{type}_0 \mid \Delta}$

$$\frac{\Gamma(x) = \mathit{type} \quad \Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0 \mid \Delta}{\Gamma \vdash x \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type} \Leftarrow \mathit{type}_0 \mid \Delta}{\Gamma \vdash (\mathit{type} : \mathit{type}_0) \Rightarrow \mathit{type}_0 \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_1 \mid \Delta_1 \quad \Delta_1 \vdash \mathit{type}_1 \leq \mathit{type}_2 \mid \Delta}{\Gamma \vdash \mathit{type} \Leftarrow \mathit{type}_2 \mid \Delta}$$

$$\frac{}{\Gamma \vdash \mathbf{Type} \Rightarrow \mathbf{Type} \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathbf{Int} \Leftarrow \mathbf{Type} \mid \Delta}{\Gamma \vdash n \Rightarrow \mathbf{Int} \mid \Delta}$$

$$\frac{\Gamma \vdash (\circ; \mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \mid \Delta}{\Gamma \vdash (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \Rightarrow \mathbf{Type} \mid \Delta}$$

$$\frac{\Delta_0 = \Gamma \quad \Delta_0 \vdash \mathit{typeTupleItem}_1 \Rightarrow \mathit{typeTupleSigItem}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeTupleItem}_n \Rightarrow \mathit{typeTupleSigItem}_n \mid \Delta_n}{\Gamma \vdash (\mathit{typeTupleItem}_1; \cdots; \mathit{typeTupleItem}_n) \Rightarrow (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n)}$$

$$\frac{\Gamma \vdash (\circ; \mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta}{\Gamma \vdash (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \Rightarrow \mathbf{Type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathbf{block}(\circ; \mathit{typeBlockItem}_1; \cdots; \mathit{typeBlockItem}_n); () \Rightarrow \mathit{type} \mid \Delta}{\Gamma \vdash \mathbf{block}(\mathit{typeBlockItem}_1; \cdots; \mathit{typeBlockItem}_n) \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{type} \Rightarrow (\mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type}_0 \mid \Delta_0\\ \Delta_0 \vdash \mathit{typeTupleItem}_1 \Leftarrow \mathit{typeTupleSigItem}_1 \mid \Delta_1 \quad \cdots \quad \Delta_{n-1} \vdash \mathit{typeTupleItem}_n \Leftarrow \mathit{typeTupleSigItem}_n \mid \Delta_n\end{array}}{\Gamma \vdash \mathit{type}(\mathit{typeTupleItem}_1; \cdots; \mathit{typeTupleItem}_n) \Rightarrow \mathit{type}_0 \mid \Delta_n}$$

Tuple Type:

$\boxed{\Gamma \vdash (\circ; \mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \mid \Delta}$

$$\frac{}{\Gamma \vdash (\circ) \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{type}_1 \Leftarrow \mathbf{Type} \mid \Delta_1 \quad \Delta_1; x_1 : \mathit{type}_1 \vdash (\circ; \mathit{typeTupleSigItem}_2; \cdots; \mathit{typeTupleSigItem}_n) \mid \Delta; x_1 : \mathit{type}_1; \Delta_2}{\Gamma \vdash (\circ; x_1 : \mathit{type}_1; \mathit{typeTupleSigItem}_2; \cdots; \mathit{typeTupleSigItem}_n) \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type}_1 \Rightarrow \mathit{type} \mid \Delta_1 \quad \Delta_1; x_1 : \mathit{type}_1 \vdash (\circ; \mathit{typeTupleSigItem}_2; \cdots; \mathit{typeTupleSigItem}_n) \mid \Delta; x_1 : \mathit{type}_1; \Delta_2}{\Gamma \vdash (\circ; \mathbf{prom}(x_1) : \mathit{type}_1; \mathit{typeTupleSigItem}_2; \cdots; \mathit{typeTupleSigItem}_n) \mid \Delta}$$

Type Tuple:

$\boxed{\Gamma \vdash (\circ; \mathit{typeTupleItem}_1; \cdots; \mathit{typeTupleItem}_n) \Rightarrow (\circ; \mathit{typeTupleSigItem}_1; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta}$

$$\frac{}{\Gamma \vdash (\circ) \Rightarrow (\circ) \mid \Gamma}$$

$$\frac{\begin{array}{l}\Gamma \vdash \mathit{type}_1 \Rightarrow \mathit{type} \mid \Delta_1\\ \Delta_1; x_1 : \mathit{type} \vdash (\circ; \mathit{typeTupleItem}_2; \cdots; \mathit{typeTupleItem}_n)\\ \quad \Rightarrow (\circ, \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta; x_1 : \mathit{type}; \Delta_2\end{array}}{\begin{array}{l}\Gamma \vdash (\circ; x_1 = \mathit{type}_1; \mathit{typeTupleItem}_2; \cdots; \mathit{typeTupleItem}_n)\\ \quad \Rightarrow (\circ; x_1 : \mathit{type}; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Gamma\end{array}}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{type}_1 \Leftarrow \mathit{type} \mid \Delta_1 \\ \Delta_1; x_1 : \mathit{type} \vdash (\circ; \mathit{typeTupleItem}_2; \cdots; \mathit{typeTupleItem}_n) \\ \Leftarrow (\circ, \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta; x_1 : \mathit{type}; \Delta_2\end{array}}{\begin{array}{c}\Gamma \vdash (\circ; x_1 = \mathit{type}_1; \mathit{typeTupleItem}_2; \cdots; \mathit{typeTupleItem}_n) \\ \Leftarrow (\circ; x_1 : \mathit{type}; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Gamma\end{array}}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{type}_1 \Rightarrow \mathit{type} \mid \Delta_1 \\ \Delta_1; x_1 : \mathit{type}; x_1 = \mathit{type}_1 \vdash (\circ; \mathit{typeTupleItem}_2; \cdots; \mathit{typeTupleItem}_n) \\ \Rightarrow (\circ, \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta; x_1 : \mathit{type}; \Delta_2\end{array}}{\begin{array}{c}\Gamma \vdash (\circ; \mathbf{prom}(x_1) = \mathit{type}_1; \mathit{typeTupleItem}_2; \cdots; \mathit{typeTupleItem}_n) \\ \Rightarrow (\circ; \mathbf{prom}(x_1) : \mathit{type}; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Gamma\end{array}}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{type}_1 \Leftarrow \mathit{type} \mid \Delta_1 \\ \Delta_1; x_1 : \mathit{type}; x_1 = \mathit{type}_1 \vdash (\circ; \mathit{typeTupleItem}_2; \cdots; \mathit{typeTupleItem}_n) \\ \Leftarrow (\circ, \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Delta; x_1 : \mathit{type}; \Delta_2\end{array}}{\begin{array}{c}\Gamma \vdash (\circ; \mathbf{prom}(x_1) = \mathit{type}_1; \mathit{typeTupleItem}_2; \cdots; \mathit{typeTupleItem}_n) \\ \Leftarrow (\circ; \mathbf{prom}(x_1) : \mathit{type}; \mathit{typeTupleSigItem}_2; \ldots; \mathit{typeTupleSigItem}_n) \mid \Gamma\end{array}}$$

Type Block:

$$\boxed{\Gamma \vdash \mathbf{block}(\circ; \mathit{typeBlockItem}_1; \cdots; \mathit{typeBlockItem}_n); \mathit{type}_0 \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{}{\Gamma \vdash \mathbf{block}(\circ); \mathit{type} \Rightarrow \mathit{type} \mid \Gamma}$$

$$\frac{\Gamma \vdash \mathit{type}_1 \Rightarrow \mathit{type}'_1 \mid \Delta_1 \quad \Delta_1 \vdash \mathbf{block}(\circ; \mathit{typeBlockItem}_2; \cdots; \mathit{typeBlockItem}_n); \mathit{type}'_1 \Rightarrow \mathit{type} \mid \Delta}{\Gamma \vdash \mathbf{block}(\circ; \mathit{type}_1; \mathit{typeBlockItem}_2; \cdots; \mathit{typeBlockItem}_n); \mathit{type}_0 \Rightarrow \mathit{type} \mid \Delta}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{type}_1 \Rightarrow \mathit{type}'_1 \mid \Delta_1 \\ \Delta_1; x_1 : \mathit{type}'_1 \vdash \mathbf{block}(\circ; \mathit{typeBlockItem}_2; \cdots; \mathit{typeBlockItem}_n); () \Rightarrow \mathit{type} \mid \Delta; x_1 : \mathit{type}'_1; x_1 = \mathit{type}_1; \Delta_2\end{array}}{\Gamma \vdash \mathbf{block}(\circ; \mathbf{let}(x_1 = \mathit{type}_1); \mathit{typeBlockItem}_2; \cdots; \mathit{typeBlockItem}_n); \mathit{type}_0 \Rightarrow \mathit{type} \mid \Delta}$$

Abstraction Type:

$$\boxed{\Gamma \vdash (\circ; \mathit{typeTupleSigItem}_1; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type} \Rightarrow \mathit{type}_0 \mid \Delta}{\Gamma \vdash (\circ) \to \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type}_1 \Leftarrow \mathbf{Type} \mid \Delta_1 \quad \Delta_1; x_1 : \mathit{type}_1 \vdash (\circ; \mathit{typeTupleSigItem}_2; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta; x_1 : \mathit{type}_1; \Delta_2}{\Gamma \vdash (\circ; x_1 : \mathit{type}_1; \mathit{typeTupleSigItem}_2; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta}$$

$$\frac{\Gamma \vdash \mathit{type}_1 \Rightarrow \mathit{type}'_1 \mid \Delta_1 \quad \Delta_1; x_1 : \mathit{type}_1 \vdash (\circ; \mathit{typeTupleSigItem}_2; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta; x_1 : \mathit{type}_1; \Delta_2}{\Gamma \vdash (\circ; \mathbf{prom}(x_1) : \mathit{type}_1; \mathit{typeTupleSigItem}_2; \cdots; \mathit{typeTupleSigItem}_n) \to \mathit{type} \mid \Delta}$$

20.4 Strik Module System

20.4.1 Syntax

$$
\begin{array}{rcl}
e & ::= & \cdots \\
 & | & \mathrm{letrec}\{B\}\ \mathrm{in}\ e \\
 & | & P \\
\tau & ::= & \cdots \\
 & | & P \\
P & ::= & M \\
M & ::= & x \\
 & | & \{B\} \\
 & | & M.x \\
 & | & \mathrm{fun}\, x : S.\, M \\
 & | & x\ x \\
 & | & x : S \\
B & ::= & x = e \\
 & | & \mathrm{type}\, t = T \\
 & | & \mathrm{module}\, x = M \\
 & | & \mathrm{use}\, B \\
 & | & \epsilon \\
 & | & B; B \\
T & ::= & \lambda x.\, T \\
 & | & \tau \\
S & ::= & P \\
 & | & \{D\} \\
 & | & (x : S) \to S \\
 & | &
\end{array}
$$

参考文献

[DK13] Jana Danfield and Neelakantan R. Krishnaswami. Complete and easy bidirectional typechecking for higher-rank polymorphism. pages 429–442. ACM, 9 2013.

[DLM+78] Edsger W. Dijkstra, Leslie Lamport, A. J. Martin, C. S. Scholten, and E. F. M. Steffens. On-the-fly garbage collection: an exercise in cooperation. *Communications of the ACM*, 21(11):966–975, November 1978.

[EL04] Van Eekelen and Van Leer. The essence of ml type inference. In *Advanced Topics in Types and Programming Languages*, chapter 10. The MIT Press, 2004.

[For02] Bryan Ford. Packrat Parsing : a Practical Linear-Time Algorithm with Backtracking. Master's thesis, Massachusetts Institute of Technology, 2002.

[For04] Bryan Ford. Parsing Expression Grammars: A Recognition-Based Syntactic Foundation. *ACM SIGPLAN Notices*, 39(1):111–122, jan 2004.

[GD20] Ben Gamari and Laura Dietz. Alligator collector: a latency-optimized garbage collector for functional programming languages. In *Proceedings of the 2020 ACM SIGPLAN International Symposium on Memory Management*, pages 87–99, London UK, June 2020. ACM.

[GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. Cambridge University Press, apr 1989.

[JVWS07] Simon Peyton Jones, Dimitrios Vytiniotis, Stephanie Weirich, and Mark Shields. Practical type inference for arbitrary-rank types. *Journal of Functional Programming*, 17:1–82, 1 2007.

[Knu65] Donald E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6):607–639, dec 1965.

[LY98] Oukseh Lee and Kwangkeun Yi. Proofs about a folklore let-polymorphic type inference algorithm. *ACM Transactions on Programming Languages and Systems*, 20:707–723, 7 1998.

[MM82] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4:258–282, 4 1982.

[MMY08] Kota Mizushima, Atusi Maeda, and Yoshinori Yamaguchi. Improvement technique of memory efficiency of packrat parsing. 情報処理学会論文誌 *(IPSJ Journal)*, 49:117–126, 2008.

[MMY10] Kota Mizushima, Atusi Maeda, and Yoshinori Yamaguchi. Packrat parsers can handle practical grammars in mostly constant space. In *Proceedings of the 9th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering - PASTE '10*, page 29. ACM Press, 2010.

[OSW99] Martin Odersky, Martin Sulzmann, and Martin Wehr. Type inference with constrained types. *Theory and Practice of Object Systems*, 5:35–55, 1 1999.

[Roc05] Jérôme Rocheteau. Lambda-Mu-Calculus and Duality: Call-by-Name and Call-by-Value. In Jürgen Giesl, editor, *Term Rewriting and Applications*, volume 3467, pages 204–218. Springer, Berlin, Heidelberg, 2005.

[RRD14] Andreas Rossberg, Claudio Russo, and Derek Dreyer. F-ing modules. *Journal of Functional Programming*, 24(5):529–607, sep 2014.

[Sel01] Peter Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, apr 2001.

[Sim10] Simon Marlow. Haskell 2010 Language Report, 2010.

[UO16] Katsuhiro Ueno and Atsushi Ohori. A fully concurrent garbage collector for functional programs on multi-core processors. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Program-*

ming, pages 421–433, Nara Japan, September 2016. ACM.

[UOO11] Katsuhiro Ueno, Atsushi Ohori, and Toshiaki Otomo. An efficient non-moving garbage collector for functional languages. In *Proceedings of the 16th ACM SIGPLAN international conference on Functional programming*, pages 196–208, Tokyo Japan, September 2011. ACM.

[VJSS11] Dimitrios Vytiniotis, Simon Peyton Jones, Tom Schrijvers, and Martin Sulzmann. Outsidein(x): Modular type inference with local assumptions. *Journal of Functional Programming*, 21:333–412, 9 2011.

[Wel99] J.B. Wells. Typability and type checking in system f are equivalent and undecidable. *Annals of Pure and Applied Logic*, 98:111–156, 6 1999.

[WHE13] Stephanie Weirich, Justin Hsu, and Richard A. Eisenberg. System FC with explicit kind equality. In *Proceedings of the 18th ACM SIGPLAN international conference on Functional programming*, pages 275–286, Boston Massachusetts USA, September 2013. ACM.