

# プログラミング言語周りノート

-

2021 年 12 月 30 日

# 目次

第 1 章	Preliminaries	3
1.1	基本的な表記 . . . . .	4
第 2 章	Basic Calculus	7
2.1	WIP: (Untyped) $\lambda$ -Calculus . . . . .	8
2.2	Simply Typed $\lambda$ -Calculus . . . . .	9
2.3	WIP: System-T . . . . .	12
2.4	WIP: PCF . . . . .	13
2.5	System-F . . . . .	14
2.6	System-F $\omega$ . . . . .	19
2.7	$\lambda \mu$ -Calculus . . . . .	26
2.8	WIP: Lambda Bar Mu Mu Tilde Calculus . . . . .	30
2.9	WIP: $\pi$ -Calculus . . . . .	31
第 3 章	Modules and Phase Distinction	33
3.1	Light-Weight F-ing modules . . . . .	34
3.2	F-ing modules . . . . .	40
第 4 章	Control Operators	49
第 5 章	Coherent Implicit Parameter	51
第 6 章	Polymorphic Record Type	53
第 7 章	Type Checking and Inference	55
第 8 章	Static Memory Management and Regions	57
第 9 章	Dynamic Memory Management and Gabage Collection	59
第 10 章	I/O Management and Concurrency	61
第 11 章	Code Generation and Virtual Machines	63
第 12 章	Program Stability and Compatibility	65
第 13 章	Program Separation and Linking	67
第 14 章	Syntax and Parsing	69
14.1	WIP: Parsing by LR Method . . . . .	70
14.2	Syntax and Semantics of PEG . . . . .	71
14.3	Haskell Parsing with PEG . . . . .	74

---

第 15 章	Analysis and Optimizations	81
第 16 章	Meta-Programming and Multi-Stage Programming	83
第 17 章	Generic Programming	85
第 18 章	Advanced Calculus	87
第 19 章	Some Notes of Quell Ideas	89
19.1	WIP: Implementation Note of PEG Parser . . . . .	90
19.2	Quell Syntax and Identations . . . . .	94
19.3	Quell Modules . . . . .	95
参考文献		97

## 第 1 章

## Preliminaries

## 1.1 基本的な表記

量子化 (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, 量子子の束縛において, *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  $A$  の冪集合を  $\mathcal{P}(A) = \{X \mid X \subseteq A\}$ , 有限冪集合を  $\mathcal{P}_{fin}(A) = \{X \in \mathcal{P}(A) \mid X \text{ は有限集合}\}$  と表記する。
- 集合  $A_1, \dots, A_n$  の直積 (*cartesian product*) を  $A_1 \times \dots \times A_n = \{(a_1, \dots, a_n) \mid a_1 \in A_1, \dots, a_n \in A_n\}$  と表記する。集合  $A$  の  $n$  直積を  $A^n = \underbrace{A \times \dots \times A}_{n \text{ 項}}$  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  $A_1 \uplus \dots \uplus A_n = (A_1 \times \{1\}) \cup \dots \cup (A_n \times \{n\})$  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,  $a \in A_1 \uplus \dots \uplus A_n$  と表記する。
- 集合  $A$  の  $B$  との差集合を  $A \setminus B = \{a \in A \mid a \notin B\}$  と表記する。

集合  $\Sigma$  について,  $\bigcup_{n \in \mathbb{N}} \Sigma^n$  を  $\Sigma^*$  と表記する。この時,  $\alpha \in \Sigma^*$  を  $\Sigma$  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  $\sigma_1 \dots \sigma_n$  と表記する。
- 列  $\alpha = \sigma_1 \dots \sigma_n \in \Sigma^*$  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,

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

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

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

という表記を導入し, 部分関数  $f : A \rightarrow B$  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  $\text{dom}(R) = \{a \mid \exists b. (a, b) \in R\}$ , 値域 (*range*) を  $\text{cod}(R) = \{b \mid \exists a. (a, b) \in R\}$  と表記する。
- 部分関数  $f : A \rightarrow B$  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  $R \upharpoonright_X = \{(a, b) \in R \mid a \in X\}$  と表記する. 特に関数  $f : A \rightarrow B$  の  $X \subseteq A$  による制限は, 関数  $f \upharpoonright_X : X \rightarrow B$  になる.
- $a \in A$ ,  $b \in B$  について, その組を  $a \mapsto b = (a, b)$ , 関数  $f : A \rightarrow B$  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.

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



## 第 2 章

## Basic Calculus



## 2.1 WIP: (Untyped) $\lambda$ -Calculus

## 2.2 Simply Typed $\lambda$ -Calculus

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

### 2.2.1 Syntax

$e$	::=	$x$	(variable)
		$e e$	(application)
		$\lambda x : \tau. e$	(abstraction)
		$c_A$	(constant)
$\tau$	::=	$A$	(atomic type)
		$\tau \rightarrow \tau$	(function type)
$\Gamma$	::=	$\cdot$	(empty)
		$\Gamma, x : \tau$	(cons)

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:

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

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

Free Variable:

$$\boxed{fv(e) = \{\bar{x}'\}}$$

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

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.

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

$$\frac{[\bar{x}' \leftarrow \bar{e}'](x) = e}{x[\bar{x}' \leftarrow \bar{e}'] = e} \quad \frac{x \notin \text{dom}([\bar{x}' \leftarrow \bar{e}'])}{x[\bar{x}' \leftarrow \bar{e}'] = x}$$

$$\frac{e_1[\bar{x}' \leftarrow \bar{e}'] = e_1'' \quad e_2[\bar{x}' \leftarrow \bar{e}'] = e_2''}{(e_1 e_2)[\bar{x}' \leftarrow \bar{e}'] = e_1'' e_2''} \quad \frac{e([\bar{x}' \leftarrow \bar{e}'] \upharpoonright_{\text{dom}([\bar{x}' \leftarrow \bar{e}']) \setminus \{x\}}) = e''}{(\lambda x : \tau. e)[\bar{x}' \leftarrow \bar{e}'] = \lambda x : \tau. e''} \quad \frac{}{c_A[\bar{x}' \leftarrow \bar{e}'] = c_A}$$

$\alpha$ -Equality:

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

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

**定理 1 (Correctness of Substitution).** 式  $e$ , 置換  $[\bar{x}' \leftarrow \bar{e}']$  について,  $X = \text{dom}([\bar{x}' \leftarrow \bar{e}'])$  とした時,

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

□

**定理 2 ( $\alpha$ -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

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

$$\begin{array}{c} \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \text{ T-Var} \\ \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1. e : \tau_1 \rightarrow \tau_2} \text{ T-Abs} \\ \frac{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1 e_2 : \tau} \text{ T-App} \\ \frac{}{\Gamma \vdash c_A : A} \text{ T-Const} \end{array}$$

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

### 2.2.3 Evaluation Semantics (Call-By-Value)

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

Small Step:

$$\boxed{e \Rightarrow e'}$$

$$\begin{array}{c} \overline{(\lambda x : \tau. e) v \Rightarrow e[x \leftarrow v]} \\ \frac{e \Rightarrow e'}{C[e] \Rightarrow C[e']} \end{array}$$

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

□

**定理 4 (Type Soundness).**  $e : \tau$  の時,  $e \Rightarrow^* v$ ,  $e \Downarrow v$  となる  $v = \text{nf}(\Rightarrow, e)$  が存在し,

- $\tau = \tau_1 \rightarrow \tau_2$  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

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

$$\begin{array}{c}
\frac{\Gamma, x : \tau \vdash e_1 : \tau_2 \rightarrow \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\lambda x : \tau. e_1) e_2 \equiv e_1[x \leftarrow e_2] : \tau} \text{Eq-}\beta\text{-Lam} \quad \frac{x \notin \text{fv}(e) \quad \Gamma \vdash e : \tau_1 \rightarrow \tau_2}{\Gamma \vdash (\lambda x : \tau_1. e x) \equiv e : \tau_1 \rightarrow \tau_2} \text{Eq-}\eta\text{-Lam} \\
\frac{e_1 \equiv_{\alpha} e_2 \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau} \text{Eq-}\alpha\text{-Refl} \\
\frac{\Gamma \vdash e_2 \equiv e_1 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau} \text{Eq-Sym} \quad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau \quad \Gamma \vdash e_2 \equiv e_3 : \tau}{\Gamma \vdash e_1 \equiv e_3 : \tau} \text{Eq-Trans} \\
\frac{\Gamma, x : \tau \vdash e_1 \equiv e_2 : \tau'}{\Gamma \vdash \lambda x : \tau. e_1 \equiv \lambda x : \tau. e_2 : \tau \rightarrow \tau'} \text{Eq-Cong-Abs} \quad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau' \rightarrow \tau \quad \Gamma \vdash e'_1 \equiv e'_2 : \tau'}{\Gamma \vdash e_1 e'_1 \equiv e_2 e'_2 : \tau} \text{Eq-Cong-App}
\end{array}$$

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

**定理 5 (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 6 (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 7.**  $e_1 \equiv e_2 : \tau$  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 6 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 6 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

$e ::=$	$x$	(variable)
	$  \lambda x : \tau. e$	(abstraction)
	$  e e$	(application)
	$  \Lambda t. e$	(universal abstraction)
	$  e \tau$	(universal application)
$\tau ::=$	$t$	(type variable)
	$  \tau \rightarrow \tau$	(function type)
	$  \forall t. \tau$	(polymorphic type)
$\Gamma ::=$	$\cdot$	(empty)
	$  \Gamma, x : \tau$	(variable cons)
	$  \Gamma, t : \Omega$	(type variable cons)

Convention:

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

Environment Reference:

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

$$\begin{array}{ccc} \frac{x = x'}{(\Gamma, x' : \tau)(x) = \tau} & \frac{x \neq x' \quad \Gamma(x) = \tau}{(\Gamma, x' : \tau')(x) = \tau} & \frac{\Gamma(x) = \tau}{(\Gamma, t : \Omega)(x) = \tau} \\ \frac{t = t'}{(\Gamma, t' : \Omega)(t) = \Omega} & \frac{t \neq t' \quad \Gamma(t) = \Omega}{(\Gamma, t' : \Omega')(t) = \Omega} & \frac{\Gamma(t) = \Omega}{(\Gamma, x : \tau)(t) = \Omega} \end{array}$$

Free Variable:

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

$$\begin{array}{ccccc} \frac{}{fv(x) = \{x\}} & \frac{fv(e_1) = X_1 \quad fv(e_2) = X_2}{fv(e_1 e_2) = X_1 \cup X_2} & \frac{fv(e) = X}{fv(\lambda x : \tau. e) = X \setminus \{x\}} & \frac{fv(e) = X}{fv(e \tau) = X} & \frac{fv(e) = X}{fv(\Lambda t. e) = X} \end{array}$$

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.

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

$$\begin{array}{ccc} \frac{[\bar{x}' \leftarrow \bar{e}'](x) = e}{x[\bar{x}' \leftarrow \bar{e}'] = e} & \frac{x \notin \text{dom}([\bar{x}' \leftarrow \bar{e}'])}{x[\bar{x}' \leftarrow \bar{e}'] = x} & \\ \frac{e_1[\bar{x}' \leftarrow \bar{e}'] = e_1'' \quad e_2[\bar{x}' \leftarrow \bar{e}'] = e_2''}{(e_1 e_2)[\bar{x}' \leftarrow \bar{e}'] = e_1'' e_2''} & \frac{e([\bar{x}' \leftarrow \bar{e}'] \upharpoonright_{\text{dom}([\bar{x}' \leftarrow \bar{e}']) \setminus \{x\}}) = e''}{(\lambda x : \tau. e)[\bar{x}' \leftarrow \bar{e}'] = \lambda x : \tau. e''} & \\ \frac{e[\bar{x}' \leftarrow \bar{e}'] = e''}{(e \tau)[\bar{x}' \leftarrow \bar{e}'] = e'' \tau} & \frac{e[\bar{x}' \leftarrow \bar{e}'] = e''}{(\Lambda t. e)[\bar{x}' \leftarrow \bar{e}'] = \Lambda t. e''} & \end{array}$$

Type Free Variable:

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

$$\begin{array}{c} \overline{tyfv(x) = \emptyset} \quad \frac{tyfv(e_1) = T_1 \quad tyfv(e_2) = T_2}{tyfv(e_1 e_2) = T_1 \cup T_2} \quad \frac{tyfv(\tau) = T_1 \quad tyfv(e) = T_2}{tyfv(\lambda x : \tau. e) = T_1 \cup T_2} \\ \frac{tyfv(e) = T_1 \quad tyfv(\tau) = T_2}{tyfv(e \tau) = T_1 \cup T_2} \quad \frac{tyfv(e) = T}{tyfv(\Lambda t. e) = T \setminus \{t\}} \\ \overline{tyfv(t) = \{t\}} \quad \frac{tyfv(\tau_1) = T_1 \quad tyfv(\tau_2) = T_2}{tyfv(\tau_1 \rightarrow \tau_2) = T_1 \cup T_2} \quad \frac{tyfv(\tau) = T}{tyfv(\forall t. \tau) = T \setminus \{t\}} \end{array}$$

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  $[t_1, \dots, t_n \leftarrow \tau_1, \dots, \tau_n]$  と表記する.

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

$$\begin{array}{c} \overline{x[t' \leftarrow \bar{\tau}'] = x} \quad \frac{e_1[t' \leftarrow \bar{\tau}'] = e_1'' \quad e_2[t' \leftarrow \bar{\tau}'] = e_2''}{(e_1 e_2)[t' \leftarrow \bar{\tau}'] = e_1'' e_2''} \quad \frac{\tau[t' \leftarrow \bar{\tau}'] = \tau'' \quad e[t' \leftarrow \bar{\tau}'] = e''}{(\lambda x : \tau. e)[t' \leftarrow \bar{\tau}'] = \lambda x : \tau''. e''} \\ \frac{e[t' \leftarrow \bar{\tau}'] = e'' \quad \tau[t' \leftarrow \bar{\tau}'] = \tau''}{(e \tau)[t' \leftarrow \bar{\tau}'] = e'' \tau''} \quad \frac{e([t' \leftarrow \bar{\tau}'] \upharpoonright_{\text{dom}([t' \leftarrow \bar{\tau}']) \setminus \{t\}}) = e''}{(\Lambda t. e)[t' \leftarrow \bar{\tau}'] = \Lambda t. e''} \end{array}$$

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

$$\begin{array}{c} \frac{[\bar{t}' \leftarrow \bar{\tau}'](t) = \tau}{t[\bar{t}' \leftarrow \bar{\tau}'] = \tau} \quad \frac{t \notin \text{dom}([\bar{t}' \leftarrow \bar{\tau}'])}{t[\bar{t}' \leftarrow \bar{\tau}'] = t} \quad \frac{\tau_1[\bar{t}' \leftarrow \bar{\tau}'] = \tau_1'' \quad \tau_2[\bar{t}' \leftarrow \bar{\tau}'] = \tau_2''}{(\tau_1 \rightarrow \tau_2)[\bar{t}' \leftarrow \bar{\tau}'] = \tau_1'' \rightarrow \tau_2''} \quad \frac{\tau([\bar{t}' \leftarrow \bar{\tau}'] \upharpoonright_{\text{dom}([\bar{t}' \leftarrow \bar{\tau}']) \setminus \{t\}}) = \tau''}{(\forall t. \tau)[\bar{t}' \leftarrow \bar{\tau}'] = \forall t. \tau''} \end{array}$$

$\alpha$ -Equality:

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

$$\begin{array}{c} \frac{x_1 = x_2}{x_1 \equiv_{\alpha} x_2} \quad \frac{e_1 \equiv_{\alpha} e_2 \quad e_1' \equiv_{\alpha} e_2'}{e_1 e_1' \equiv_{\alpha} e_2 e_2'} \quad \frac{\tau_1 \equiv_{\alpha} \tau_2 \quad x' \notin f\upsilon(e_1) \cup f\upsilon(e_2) \quad e_1[x_1 \leftarrow x'] \equiv_{\alpha} e_2[x_2 \leftarrow x']}{\lambda x_1 : \tau_1. e_1 \equiv_{\alpha} \lambda x_2 : \tau_2. e_2} \\ \frac{e_1 \equiv_{\alpha} e_2 \quad \tau_1 \equiv_{\alpha} \tau_2}{e_1 \tau_1 \equiv_{\alpha} e_2 \tau_2} \quad \frac{t' \notin tyfv(e_1) \cup tyfv(e_2) \quad e_1[t_1 \leftarrow t'] \equiv_{\alpha} e_2[t_2 \leftarrow t']}{\Lambda t_1. e_1 \equiv_{\alpha} \Lambda t_2. e_2} \end{array}$$

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

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

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

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

□

**定理 9 (Correctness of Type Substitution).** 式  $e$ , 型  $\tau$ , 型置換  $[\bar{t}' \leftarrow \bar{\tau}']$  について,  $T = \text{dom}([\bar{t}' \leftarrow \bar{\tau}'])$  とした時,

$$\begin{array}{c} tyfv(e[\bar{t}' \leftarrow \bar{\tau}']) = (tyfv(e) \setminus T) \cup \bigcup_{t \in tyfv(e) \cap T} tyfv([\bar{t}' \leftarrow \bar{\tau}'])(t) \\ tyfv(\tau[\bar{t}' \leftarrow \bar{\tau}']) = (tyfv(\tau) \setminus T) \cup \bigcup_{t \in tyfv(\tau) \cap T} tyfv([\bar{t}' \leftarrow \bar{\tau}'])(t). \end{array}$$

□

**定理 10 ( $\alpha$ -Equality Does Not Touch Free Variables).**



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

□

### 2.5.2 Typing Semantics

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

$$\begin{array}{c}
 \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \text{ T-Var} \\
 \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1. e : \tau_1 \rightarrow \tau_2} \text{ T-Abs} \\
 \frac{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1 e_2 : \tau} \text{ T-App} \\
 \frac{\Gamma, t : \Omega \vdash e : \tau}{\Gamma \vdash \Lambda t. e : \forall t. \tau} \text{ T-UnivAbs} \\
 \frac{\Gamma \vdash e : \forall t. \tau_1}{\Gamma \vdash e \tau_2 : \tau_1[t \leftarrow \tau_2]} \text{ T-UnivApp} \\
 \frac{\Gamma \vdash \tau \equiv_{\alpha} \tau' : \Omega \quad \Gamma \vdash e : \tau'}{\Gamma \vdash e : \tau} \text{ T-}\alpha\text{-Equiv}
 \end{array}$$

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

### 2.5.3 Evaluation Semantics (Call-By-Value)

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

Small Step:

$$\boxed{e \Rightarrow e'}$$

$$\begin{array}{c}
 \overline{(\lambda x : \tau. e) v \Rightarrow e[x \leftarrow v]} \\
 \overline{(\Lambda t. e) \tau \Rightarrow e[t \leftarrow \tau]} \\
 \frac{e \Rightarrow e'}{C[e] \Rightarrow C[e']}
 \end{array}$$

Big Step:

$$\boxed{e \Downarrow v}$$

$$\begin{array}{c}
 \frac{e_1 \Downarrow \lambda x : \tau. e'_1 \quad e_2 \Downarrow v_2 \quad e'_1[x \leftarrow v_2] \Downarrow v}{e_1 e_2 \Downarrow v} \\
 \frac{e \Downarrow \Lambda t. e'_1 \quad e'_1[t \leftarrow \tau] \Downarrow v}{e \tau \Downarrow v}
 \end{array}$$

**定理 11 (Adequacy of Small Step and Big Step).**  $e \Rightarrow^* v$  iff  $e \Downarrow v$ .

□

**定理 12 (Type Soundness).**  $e : \tau$  の時,  $e \Rightarrow^* v$ ,  $e \Downarrow v$  となる  $v = \text{nf}(\Rightarrow, e)$  が存在し,

- $\tau = \tau_1 \rightarrow \tau_2$  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

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

$$\begin{array}{c}
\frac{\Gamma, x : \tau_2 \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\lambda x : \tau_2. e_1) e_2 \equiv e_1[x \leftarrow e_2] : \tau} \text{Eq-}\beta\text{-Lam} \quad \frac{x \notin \text{fv}(e) \quad \Gamma \vdash e : \tau_1 \rightarrow \tau_2}{\Gamma \vdash (\lambda x : \tau_1. e) x \equiv e : \tau_1 \rightarrow \tau_2} \text{Eq-}\eta\text{-Lam} \\
\frac{\Gamma, t : \Omega \vdash e : \tau}{\Gamma \vdash (\Lambda t. e) \tau_2 \equiv e[t \leftarrow \tau_2] : \tau[t \leftarrow \tau_2]} \text{Eq-}\beta\text{-UnivLam} \quad \frac{t \notin \text{tyfv}(e) \quad \Gamma \vdash e : \forall t'. \tau}{\Gamma \vdash (\Lambda t. e) t \equiv e : \forall t'. \tau} \text{Eq-}\eta\text{-UnivLam} \\
\frac{e_1 \equiv_\alpha e_2 \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau} \text{Eq-}\alpha\text{-Refl} \quad \frac{\tau \equiv_\alpha \tau' \quad \Gamma \vdash e_1 \equiv e_2 : \tau'}{\Gamma \vdash e_1 \equiv e_2 : \tau} \text{Eq-}\alpha\text{-Type} \\
\frac{\Gamma \vdash e_2 \equiv e_1 : \tau}{\Gamma \vdash e_1 \equiv e_2 : \tau} \text{Eq-Sym} \quad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau \quad \Gamma \vdash e_2 \equiv e_3 : \tau}{\Gamma \vdash e_1 \equiv e_3 : \tau} \text{Eq-Trans} \\
\frac{\Gamma, x : \tau \vdash e_1 \equiv e_2 : \tau'}{\Gamma \vdash \lambda x : \tau. e_1 \equiv \lambda x : \tau. e_2 : \tau \rightarrow \tau'} \text{Eq-Cong-Abs} \quad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau' \rightarrow \tau \quad \Gamma \vdash e'_1 \equiv e'_2 : \tau'}{\Gamma \vdash e_1 e'_1 \equiv e_2 e'_2 : \tau} \text{Eq-Cong-App} \\
\frac{\Gamma, t : \Omega \vdash e_1 \equiv e_2 : \tau}{\Gamma \vdash \Lambda t. e_1 \equiv \Lambda t. e_2 : \forall(t. \tau)} \text{Eq-Cong-UnivAbs} \quad \frac{\Gamma \vdash e_1 \equiv e_2 : \forall t. \tau}{\Gamma \vdash e_1 \tau' \equiv e_2 \tau' : \tau[t \leftarrow \tau']} \text{Eq-Cong-UnivApp}
\end{array}$$

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

**定理 13 (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 14 (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 15.**  $e_1 \equiv e_2 : \tau$  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 14 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 14 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  $\tau_1$  and  $\tau_2$ :

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

Admissible typing rule:

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

$$\begin{array}{c}
\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2} \text{T-Product} \quad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \pi_1 e : \tau_1} \text{T-Prodj-1} \quad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \pi_2 e : \tau_2} \text{T-Prodj-2}
\end{array}$$

Admissible equality:

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

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

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

Existential Type

Existence of  $\exists t. \tau$ :

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

Admissible typing rule:

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

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

Admissible equality:

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

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

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

## 2.6 System-F $\omega$

Alias: F  $\omega$ ,  $\lambda\omega$  [RRD14]

### 2.6.1 Syntax

$e ::=$	$x$	(variable)
	$  \lambda x : \tau. e$	(abstraction)
	$  e e$	(application)
	$  \Lambda t : \kappa. e$	(universal abstraction)
	$  e \tau$	(universal application)
$\tau ::=$	$t$	(type variable)
	$  \tau \rightarrow \tau$	(function type)
	$  \forall t : \kappa. \tau$	(polymorphic type)
	$  \lambda t : \kappa. \tau$	(type abstraction)
	$  \tau \tau$	(type application)
$\kappa ::=$	$\Omega$	(type kind)
	$  \kappa \rightarrow \kappa$	(arrow kind)
$\Gamma ::=$	$\cdot$	(empty)
	$  \Gamma, x : \tau$	(variable cons)
	$  \Gamma, t : \kappa$	(type variable cons)

Convention:

$$\tau_1 \rightarrow \tau_2 \rightarrow \dots \rightarrow \tau_n \stackrel{\text{def}}{=} \tau_1 \rightarrow (\tau_2 \rightarrow (\dots \rightarrow \tau_n) \dots)$$

$$e_1 e_2 \dots e_n \stackrel{\text{def}}{=} (\dots (e_1 e_2) \dots) e_n$$

Environment Reference:

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

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

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

Free Variable:

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

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

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.

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

$$\frac{\frac{\bar{x}' \leftarrow \bar{e}'](x) = e}{x[\bar{x}' \leftarrow \bar{e}'] = e} \quad \frac{x \notin \text{dom}([\bar{x}' \leftarrow \bar{e}'])}{x[\bar{x}' \leftarrow \bar{e}'] = x}}{e([\bar{x}' \leftarrow \bar{e}'] \upharpoonright_{\text{dom}([\bar{x}' \leftarrow \bar{e}']) \setminus \{x\}}) = e''} \quad \frac{e_1[\bar{x}' \leftarrow \bar{e}'] = e'_1 \quad e_2[\bar{x}' \leftarrow \bar{e}'] = e'_2}{(e_1 e_2)[\bar{x}' \leftarrow \bar{e}'] = e'_1 e'_2}$$

$$\frac{e([\bar{x}' \leftarrow \bar{e}'] \upharpoonright_{\text{dom}([\bar{x}' \leftarrow \bar{e}']) \setminus \{x\}}) = e''}{(\lambda x : \tau. e)[\bar{x}' \leftarrow \bar{e}'] = \lambda x : \tau. e''}$$

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

Type Free Variable:

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

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

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

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

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

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

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  $[t_1, \dots, t_n \leftarrow \tau_1, \dots, \tau_n]$  と表記する.

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

$$\frac{}{x[\bar{t}' \leftarrow \bar{\tau}'] = x} \quad \frac{e_1[\bar{t}' \leftarrow \bar{\tau}'] = e'_1 \quad e_2[\bar{t}' \leftarrow \bar{\tau}'] = e'_2}{(e_1 e_2)[\bar{t}' \leftarrow \bar{\tau}'] = e'_1 e'_2} \quad \frac{\tau[\bar{t}' \leftarrow \bar{\tau}'] = \tau'' \quad e[\bar{t}' \leftarrow \bar{\tau}'] = e''}{(\lambda x : \tau. e)[\bar{t}' \leftarrow \bar{\tau}'] = \lambda x : \tau''. e''}$$

$$\frac{e[\bar{t}' \leftarrow \bar{\tau}'] = e'' \quad \tau[\bar{t}' \leftarrow \bar{\tau}'] = \tau''}{(e \tau)[\bar{t}' \leftarrow \bar{\tau}'] = e'' \tau''} \quad \frac{e([\bar{t}' \leftarrow \bar{\tau}'] \upharpoonright_{\text{dom}([\bar{t}' \leftarrow \bar{\tau}'] \setminus \{t\})}) = e''}{(\Lambda t : \kappa. e)[\bar{t}' \leftarrow \bar{\tau}'] = \Lambda t : \kappa. e''}$$

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

$$\frac{[\bar{t}' \leftarrow \bar{\tau}'](t) = \tau}{t[\bar{t}' \leftarrow \bar{\tau}'] = \tau} \quad \frac{t \notin \text{dom}([\bar{t}' \leftarrow \bar{\tau}'])}{t[\bar{t}' \leftarrow \bar{\tau}'] = t}$$

$$\frac{\tau_1[\bar{t}' \leftarrow \bar{\tau}'] = \tau_1'' \quad \tau_2[\bar{t}' \leftarrow \bar{\tau}'] = \tau_2''}{(\tau_1 \rightarrow \tau_2)[\bar{t}' \leftarrow \bar{\tau}'] = \tau_1'' \rightarrow \tau_2''} \quad \frac{\tau([\bar{t}' \leftarrow \bar{\tau}'] \upharpoonright_{\text{dom}([\bar{t}' \leftarrow \bar{\tau}'] \setminus \{t\})}) = \tau''}{(\forall t : \kappa. \tau)[\bar{t}' \leftarrow \bar{\tau}'] = \forall t : \kappa. \tau''}$$

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

$\alpha$ -Equality:

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

$$\frac{x_1 = x_2}{x_1 \equiv_{\alpha} x_2} \quad \frac{e_1 \equiv_{\alpha} e_2 \quad e'_1 \equiv_{\alpha} e'_2}{e_1 e'_1 \equiv_{\alpha} e_2 e'_2} \quad \frac{\tau_1 \equiv_{\alpha} \tau_2 \quad x' \notin fv(e_1) \cup fv(e_2) \quad e_1[x_1 \leftarrow x'] \equiv_{\alpha} e_2[x_2 \leftarrow x']}{\lambda x_1 : \tau_1. e_1 \equiv_{\alpha} \lambda x_2 : \tau_2. e_2}$$

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

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

$$\frac{t_1 = t_2}{t_1 \equiv_{\alpha} t_2} \quad \frac{\tau_1 \equiv_{\alpha} \tau_2 \quad \tau'_1 \equiv_{\alpha} \tau'_2}{\tau_1 \rightarrow \tau'_1 \equiv_{\alpha} \tau_2 \rightarrow \tau'_2} \quad \frac{t' \notin tyfv(\tau_1) \cup tyfv(\tau_2) \quad \tau_1[t_1 \leftarrow t'] \equiv_{\alpha} \tau_2[t_2 \leftarrow t']}{\forall t_1 : \kappa. \tau_1 \equiv_{\alpha} \forall t_2 : \kappa. \tau_2}$$

$$\frac{t' \notin tyfv(\tau_1) \cup tyfv(\tau_2) \quad \tau_1[t_1 \leftarrow t'] \equiv_{\alpha} \tau_2[t_2 \leftarrow t']}{\lambda t_1 : \kappa. \tau_1 \equiv_{\alpha} \lambda t_2 : \kappa. \tau_2} \quad \frac{\tau_1 \equiv_{\alpha} \tau_2 \quad \tau'_1 \equiv_{\alpha} \tau'_2}{\tau_1 \tau'_1 \equiv_{\alpha} \tau_2 \tau'_2}$$

**定理 16 (Correctness of Substitution).** 置換  $[\bar{x}' \leftarrow \bar{e}']$  について,  $X = \text{dom}([\bar{x}' \leftarrow \bar{e}'])$  とした時,

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

□

**定理 17 (Correctness of Type Substitution).** 式  $e$ , 型  $\tau$ , 型置換  $[\bar{t}' \leftarrow \bar{\tau}']$  について,  $T = \text{dom}([\bar{t}' \leftarrow \bar{\tau}'])$  とした時,

$$\begin{aligned} tyfv(e[\bar{t}' \leftarrow \bar{\tau}']) &= (tyfv(e) \setminus T) \cup \bigcup_{t \in tyfv(e) \cap T} tyfv([\bar{t}' \leftarrow \bar{\tau}'](t)) \\ tyfv(\tau[\bar{t}' \leftarrow \bar{\tau}']) &= (tyfv(\tau) \setminus T) \cup \bigcup_{t \in tyfv(\tau) \cap T} tyfv([\bar{t}' \leftarrow \bar{\tau}'](t)). \end{aligned}$$

□

**定理 18 ( $\alpha$ -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:

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

$$\begin{aligned} & \frac{\Gamma(t) = \kappa}{\Gamma \vdash t : \kappa} \text{K-Var} \\ & \frac{\Gamma \vdash \tau_1 : \Omega \quad \Gamma \vdash \tau_2 : \Omega}{\Gamma \vdash \tau_1 \rightarrow \tau_2 : \Omega} \text{K-Arrow} \\ & \frac{\Gamma, t : \kappa \vdash \tau : \Omega}{\Gamma \vdash \forall t : \kappa. \tau : \Omega} \text{K-Forall} \\ & \frac{\Gamma, t : \kappa_1 \vdash \tau : \kappa_2}{\Gamma \vdash \lambda t : \kappa_1. \tau : \kappa_1 \rightarrow \kappa_2} \text{K-Abs} \\ & \frac{\Gamma \vdash \tau_1 : \kappa_2 \rightarrow \kappa \quad \Gamma \vdash \tau_2 : \kappa_2}{\Gamma \vdash \tau_1 \tau_2 : \kappa} \text{K-App} \end{aligned}$$

Type equivalence:

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

$$\begin{aligned} & \frac{\Gamma, t : \kappa_2 \vdash \tau_1 : \kappa \quad \Gamma \vdash \tau_2 : \kappa_2}{\Gamma \vdash (\lambda t : \kappa_2. \tau_1) \tau_2 \equiv \tau_1[t \leftarrow \tau_2] : \kappa} \text{T-Eq-}\beta\text{-Lam} & \frac{t \notin tyfv(\tau) \quad \Gamma \vdash \tau : \kappa_1 \rightarrow \kappa_2}{\Gamma \vdash (\lambda t : \kappa_1. \tau) t \equiv \tau : \kappa_1 \rightarrow \kappa_2} \text{T-Eq-}\eta\text{-Lam} \\ & \frac{\tau_1 \equiv_{\alpha} \tau_2 \quad \Gamma \vdash \tau_1 : \kappa \quad \Gamma \vdash \tau_2 : \kappa}{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa} \text{T-Eq-}\alpha\text{-Refl} \\ & \frac{\Gamma \vdash \tau_2 \equiv \tau_1 : \kappa}{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa} \text{T-Eq-Sym} & \frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa \quad \Gamma \vdash \tau_2 \equiv \tau_3 : \kappa}{\Gamma \vdash \tau_1 \equiv \tau_3 : \kappa} \text{T-Eq-Trans} \\ & \frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \Omega \quad \Gamma \vdash \tau'_1 \equiv \tau'_2 : \Omega}{\Gamma \vdash \tau_1 \rightarrow \tau'_1 \equiv \tau_2 \rightarrow \tau'_2 : \Omega} \text{T-Eq-Cong-Arrow} & \frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau_2 : \Omega}{\Gamma \vdash \forall t : \kappa. \tau_1 \equiv \forall t : \kappa. \tau_2 : \Omega} \text{Eq-Cong-Forall} \\ & \frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau_2 : \kappa'}{\Gamma \vdash \lambda t : \kappa. \tau_1 \equiv \lambda t : \kappa. \tau_2 : \kappa \rightarrow \kappa'} \text{T-Eq-Cong-Abs} & \frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa' \rightarrow \kappa \quad \Gamma \vdash \tau'_1 \equiv \tau'_2 : \kappa'}{\Gamma \vdash \tau_1 \tau'_1 \equiv \tau_2 \tau'_2 : \kappa} \text{Eq-Cong-App} \end{aligned}$$

**定理 19 (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:

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

$$\begin{array}{c} \frac{\Gamma \vdash \tau : \Omega \quad \Gamma(x) = \tau}{\Gamma \vdash x : \tau} \text{ T-Var} \\ \frac{\Gamma \vdash \tau_1 : \Omega \quad \Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1. e : \tau_1 \rightarrow \tau_2} \text{ T-Abs} \\ \frac{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1 e_2 : \tau} \text{ T-App} \\ \frac{\Gamma, t : \kappa \vdash e : \tau}{\Gamma \vdash \Lambda t : \kappa. e : \forall t : \kappa. \tau} \text{ T-UnivAbs} \\ \frac{\Gamma \vdash e : \forall t : \kappa. \tau_1 \quad \Gamma \vdash \tau_2 : \kappa}{\Gamma \vdash e \tau_2 : \tau_1[t \leftarrow \tau_2]} \text{ T-UnivApp} \\ \frac{\Gamma \vdash \tau \equiv \tau' : \Omega \quad \Gamma \vdash e : \tau'}{\Gamma \vdash e : \tau} \text{ T-Equiv} \end{array}$$

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

**定理 20 (Respect Type Kind).**  $\Gamma \vdash e : \tau$  ならば,  $\Gamma \vdash \tau : \Omega$ .

□

### 2.6.3 Evaluation Semantics (Call-By-Value)

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

Small Step:

$$\boxed{e \Rightarrow e'}$$

$$\begin{array}{c} \overline{(\lambda x : \tau. e) v \Rightarrow e[x \leftarrow v]} \\ \overline{(\Lambda t : \kappa. e) \tau \Rightarrow e[t \leftarrow \tau]} \\ \frac{e \Rightarrow e'}{C[e] \Rightarrow C[e']} \end{array}$$

Big Step:

$$\boxed{e \Downarrow v}$$

$$\begin{array}{c} \frac{e_1 \Downarrow \lambda x : \tau. e'_1 \quad e_2 \Downarrow v_2 \quad e'_1[x \leftarrow v_2] \Downarrow v}{e_1 e_2 \Downarrow v} \\ \frac{e \Downarrow \Lambda t : \kappa. e'_1 \quad e'_1[t \leftarrow \tau] \Downarrow v}{e \tau \Downarrow v} \end{array}$$

**定理 21 (Adequacy of Small Step and Big Step).**  $e \Rightarrow^* v$  iff  $e \Downarrow v$ .

□

**定理 22 (Type Soundness).**  $e : \tau$  の時,  $e \Rightarrow^* v$ ,  $e \Downarrow v$  となる  $v = \text{nf}(\Rightarrow, e)$  が存在し,

- $\tau = \tau_1 \rightarrow \tau_2$  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

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

$$\begin{array}{c}
\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} \quad \frac{x \notin \text{fv}(e) \quad \Gamma \vdash e : \tau_1 \rightarrow \tau_2}{\Gamma \vdash (\lambda x : \tau_1. e) x \equiv e : \tau_1 \rightarrow \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} \quad \frac{t \notin \text{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} \quad \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} \quad \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 \rightarrow \tau'} \text{Eq-Cong-Abs} \quad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau' \rightarrow \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}
\end{array}$$

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

**定理 23 (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 24 (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 25.**  $e_1 \equiv e_2 : \tau$  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 14 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 14 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  $\tau_1$  and  $\tau_2$ :

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

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:

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

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

Admissible equality:

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

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

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

Existential Type

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

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

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:

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

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

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

Admissible equality:

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

$$\begin{array}{c}
\frac{\Gamma \vdash \tau_t : \kappa \quad \Gamma \vdash e_1 : \tau_1[t \leftarrow \tau_t] \quad \Gamma, t : \kappa, x : \tau_1 \vdash e_2 : \tau_2 \quad t \notin \text{tyfv}(\tau_2)}{\Gamma \vdash \text{unpack}\langle t : \kappa, x : \tau_1 \rangle = \text{pack}\langle \tau_t, e_1 \rangle_{\exists t : \kappa. \tau_1}. \tau_2. e_2 \equiv e_2[t \leftarrow \tau_t][x \leftarrow e_1] : \tau_2} \text{Eq-}\beta\text{-Exist} \\
\frac{\Gamma \vdash e : (\exists t : \kappa. \tau) \quad \tau' \equiv \exists t : \kappa. \tau}{\Gamma \vdash \text{unpack}\langle t : \kappa, x : \tau \rangle = e. \tau'. \text{pack}\langle t, x \rangle_{\exists t : \kappa. \tau} \equiv e : (\exists t : \kappa. \tau)} \text{Eq-}\eta\text{-Exist}
\end{array}$$

## 2.7 $\lambda \mu$ -Calculus

Alias:  $\lambda \mu$  [Sel01][Roc05]

### 2.7.1 Syntax

$\tau ::=$	$t$	(type variable)
	$\top$	(top type)
	$\tau \times \tau$	(product type)
	$\tau \rightarrow \tau$	(function type)
	$\perp$	(bottom type)
$e ::=$	$x$	(variable)
	$\langle \rangle$	(top value)
	$\langle e, e \rangle$	(product)
	$\pi_1 e$	(left projection)
	$\pi_2 e$	(right projection)
	$\lambda x : \tau. e$	(abstraction)
	$e e$	(application)
	$[\alpha]e$	(naming)
	$\mu \alpha : \tau. e$	(un-naming)
$\Gamma ::=$	$\cdot$	
	$\Gamma, x : \tau$	
$\Delta ::=$	$\cdot$	
	$\alpha : \tau, \Delta$	

Environment Reference:

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

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

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

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

### 2.7.2 Typing Semantics

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

$$\begin{array}{c} \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau \mid \Delta} \text{ T-Var} \\ \frac{}{\Gamma \vdash \langle \rangle : \top \mid \Delta} \text{ T-Top} \\ \frac{\Gamma \vdash e_1 : \tau_1 \mid \Delta \quad \Gamma \vdash e_2 : \tau_2 \mid \Delta}{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2 \mid \Delta} \text{ T-Product} \\ \frac{\Gamma \vdash e : \tau_1 \times \tau_2 \mid \Delta}{\Gamma \vdash \pi_1 e : \tau_1 \mid \Delta} \text{ T-Proj-1} \\ \frac{\Gamma \vdash e : \tau_1 \times \tau_2 \mid \Delta}{\Gamma \vdash \pi_2 e : \tau_2 \mid \Delta} \text{ T-Proj-2} \\ \frac{\Gamma, x : \tau_1 \vdash e : \tau_2 \mid \Delta}{\Gamma \vdash \lambda x : \tau_1. e : \tau_1 \rightarrow \tau_2 \mid \Delta} \text{ T-Abs} \end{array}$$

$$\begin{array}{c}
\frac{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau \mid \Delta \quad \Gamma \vdash e_2 : \tau_2 \mid \Delta}{\Gamma \vdash e_1 e_2 : \tau \mid \Delta} \text{ T-App} \\
\frac{\Delta(\alpha) = \tau \quad \Gamma \vdash e : \tau \mid \Delta}{\Gamma \vdash [\alpha]e : \perp \mid \Delta} \text{ T-Name} \\
\frac{\Gamma \vdash e : \perp \mid \alpha : \tau, \Delta}{\Gamma \vdash (\mu\alpha : \tau. e) : \tau \mid \Delta} \text{ T-Unname}
\end{array}$$

### 2.7.3 Equivalence

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

$$\begin{array}{c}
\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 \rightarrow \tau_2 \mid \Delta}{\Gamma \vdash (\lambda x : \tau_1. e x) \equiv e : \tau_1 \rightarrow \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 : \perp \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 : \perp \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 : \perp \mid \alpha_2 : \tau_\alpha, \Delta}{\Gamma \vdash [\alpha_1](\mu\alpha_2 : \tau_\alpha. e) \equiv e[\alpha_2 \leftarrow \alpha_1] : \perp \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 : \perp \mid \alpha : \tau_1 \rightarrow \tau_2, \Delta \quad \Gamma \vdash e_2 : \tau_2 \mid \Delta}{\Gamma \vdash (\mu\alpha : \tau_1 \rightarrow \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}
\end{array}$$

### 2.7.4 Elaboration (Call-By-Value)

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

$$\begin{array}{c}
\frac{\Gamma(x_{x_0}) = V_\tau}{\Gamma \vdash x_0 : \tau \rightsquigarrow \lambda x_k : K_\tau. x_k x_{x_0}} \\
\frac{}{\Gamma \vdash \langle \rangle : \top \rightsquigarrow \lambda x_k : K_\top. x_k \langle \rangle} \\
\frac{\Gamma \vdash e_1 : \tau_1 \rightsquigarrow e'_1 \quad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow e'_2}{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2 \rightsquigarrow \lambda x_k : K_{\tau_1 \times \tau_2}. e'_1 (\lambda x_1 : V_{\tau_1}. e'_2 (\lambda x_2 : V_{\tau_2}. x_k \langle x_1, x_2 \rangle))} \\
\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \rightsquigarrow e'}{\Gamma \vdash \pi_1 e : \tau_1 \rightsquigarrow \lambda x_k : K_{\tau_1}. e' (\lambda x : V_{\tau_1} \times V_{\tau_2}. x_k (\pi_1 x))} \\
\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \rightsquigarrow e'}{\Gamma \vdash \pi_2 e : \tau_2 \rightsquigarrow \lambda x_k : K_{\tau_2}. e' (\lambda x : V_{\tau_1} \times V_{\tau_2}. x_k (\pi_2 x))} \\
\frac{\Gamma, x_{x_0} : V_{\tau_1} \vdash e : \tau_2 \rightsquigarrow e'}{\Gamma \vdash (\lambda x_0 : \tau_1. e) : \tau_1 \rightarrow \tau_2 \rightsquigarrow \lambda x_k : K_{\tau_1 \rightarrow \tau_2}. x_k (\lambda x : V_{\tau_1} \times K_{\tau_2}. (\lambda x_{x_0} : V_{\tau_1}. e') (\pi_1 x) (\pi_2 x))}
\end{array}$$

$$\begin{array}{c}
\frac{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau \rightsquigarrow e'_1 \quad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow e'_2}{\Gamma \vdash e_1 e_2 : \tau \rightsquigarrow \lambda x_k : K_\tau. e'_1 (\lambda x_1 : V_{\tau_2 \rightarrow \tau}. e'_2 (\lambda x_2 : V_{\tau_2}. x_1 \langle x_2, x_k \rangle))} \\
\frac{\Gamma, x_\alpha : K_\tau \vdash e : \perp \rightsquigarrow e'}{\Gamma \vdash (\mu \alpha : \tau. e) : \tau \rightsquigarrow \lambda x_\alpha : K_\tau. e' (\lambda x : \perp. \text{case } x \{ \})} \\
\frac{\Gamma(x_\alpha) = K_\tau \quad \Gamma \vdash e : \tau \rightsquigarrow e'}{\Gamma \vdash [\alpha]e : \tau \rightsquigarrow \lambda x_k : K_\perp. e' x_\alpha}
\end{array}$$

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

$$\begin{array}{c}
\overline{V_\tau = \tau} \\
\frac{V_{\tau_1} = \tau'_1 \quad V_{\tau_2} = \tau'_2}{V_{\tau_1 \times \tau_2} = V_{\tau'_1} \times V_{\tau'_2}} \\
\frac{V_{\tau_1} = \tau'_1 \quad K_{\tau_2} = \tau'_2}{V_{\tau_1 \rightarrow \tau_2} = \tau'_1 \times \tau'_2 \rightarrow R} \\
\overline{V_\perp = \perp}
\end{array}$$

Abbreviation:

$$\begin{array}{l}
K_\tau \stackrel{\text{def}}{=} V_\tau \rightarrow R \\
C_\tau \stackrel{\text{def}}{=} K_\tau \rightarrow R
\end{array}$$

定理 26.  $\Gamma \vdash e : \tau \rightsquigarrow e'$  ならば,  $\Gamma \vdash e' : C_\tau$ . □

定理 27.  $\Gamma \vdash e : \tau \mid \Delta \iff V(\Gamma), K(\Delta) \vdash e : \tau \rightsquigarrow e'$ . ただし,

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

□

### 2.7.5 Elaboration (Call-By-Name)

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

$$\begin{array}{c}
\frac{\Gamma(x_{x_0}) = C_\tau}{\Gamma \vdash x_0 : \tau \rightsquigarrow \lambda x_k : K_\tau. x_{x_0} x_k} \\
\frac{\Gamma \vdash \langle \rangle : \top \rightsquigarrow \lambda x_k : \perp. \text{case } x_k \{ \}}{\Gamma \vdash e_1 : \tau_1 \rightsquigarrow e'_1 \quad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow e'_2} \\
\frac{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2 \rightsquigarrow \lambda x_k : K_{\tau_1} + K_{\tau_2}. \text{case } x_k \{ x_{k_1}. e'_1 x_{k_1} \mid x_{k_2}. e'_2 x_{k_2} \}}{\Gamma \vdash e : \tau_1 \times \tau_2 \rightsquigarrow e'} \\
\frac{\Gamma \vdash \pi_1 e : \tau_1 \rightsquigarrow \lambda x_k : K_{\tau_1}. e' (i_1 x_k)}{\Gamma, x_{x_1} : C_{\tau_1} \vdash e : \tau_2 \rightsquigarrow e'} \\
\frac{\Gamma \vdash (\lambda x_1 : \tau_1. e) : \tau_1 \rightarrow \tau_2 \rightsquigarrow \lambda x_k : C_{\tau_1} \times K_{\tau_2}. e' [x_{x_1} \leftarrow \pi_1 x_k] (\pi_2 x_k)}{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau \rightsquigarrow e'_1 \quad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow e'_2} \\
\frac{\Gamma \vdash e_1 e_2 : \tau \rightsquigarrow \lambda x_k : K_\tau. e'_1 \langle e'_2, x_k \rangle}{\Gamma(x_\alpha) = K_\tau \quad \Gamma \vdash e : \tau \rightsquigarrow e'} \\
\frac{\Gamma \vdash [\alpha]e : \perp \rightsquigarrow \lambda x_k : K_\perp. e' x_\alpha}{\Gamma, x_\alpha : K_\tau \vdash e : \perp \rightsquigarrow e'} \\
\frac{\Gamma, x_\alpha : K_\tau \vdash e : \perp \rightsquigarrow e'}{\Gamma \vdash (\mu \alpha : \tau. e) : \tau \rightsquigarrow \lambda x_\alpha : K_\tau. e' \langle \rangle}
\end{array}$$

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

$$\begin{array}{c} \overline{K_\top = \perp} \\ \frac{K_{\tau_1} = \tau'_1 \quad K_{\tau_2} = \tau'_2}{K_{\tau_1 \times \tau_2} = \tau'_1 + \tau'_2} \\ \frac{C_{\tau_1} = \tau'_1 \quad K_{\tau_2} = \tau'_2}{K_{\tau_1 \rightarrow \tau_2} = \tau'_1 \times \tau'_2} \\ \overline{K_\perp = \top} \end{array}$$

Abbreviation:

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

定理 28.  $\Gamma \vdash e : \tau \rightsquigarrow e'$  ならば,  $\Gamma \vdash e' : C_\tau$ . □

定理 29.  $\Gamma \vdash e : \tau \mid \Delta \iff C(\Gamma), K(\Delta) \vdash e : \tau \rightsquigarrow e'$ . ただし,

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

□

## 2.8 WIP: Lambda Bar Mu Mu Tilde Calculus

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

## 2.9 WIP: $\pi$ -Calculus





## 第 3 章

# Modules and Phase Distinction

### 3.1 Light-Weight F-ing modules

[RRD14]

#### 3.1.1 Internal Language

Having same power as System F  $\omega$

Syntax:

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

Abbreviation:

$$\begin{aligned}
 \Sigma.l &\stackrel{\text{def}}{=} \begin{cases} (\Sigma.l).\bar{l}' & (\bar{l} = l \bar{l}') \\ \Sigma & (\bar{l} = \epsilon) \end{cases} \\
 \bar{\tau}_1 \rightarrow \bar{\tau}_2 &\stackrel{\text{def}}{=} \begin{cases} \tau_1 \rightarrow (\bar{\tau}_1' \rightarrow \tau_2) & (\bar{\tau}_1 = \tau_1 \bar{\tau}_1') \\ \tau_2 & (\bar{\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 \bar{e}_1 &\stackrel{\text{def}}{=} \begin{cases} e_0 e_1 \bar{e}_1' & (\bar{e}_1 = e_1 \bar{e}_1') \\ e_0 & (\bar{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 \bar{\tau} &\stackrel{\text{def}}{=} \begin{cases} e \tau \bar{\tau}' & (\bar{\tau} = \tau \bar{\tau}') \\ e & (\bar{\tau} = \epsilon) \end{cases} \\
 \text{let } \overline{x : \tau} = \bar{e}_1 \overline{t : \kappa} = \bar{\tau} \text{ in } e_2 &\stackrel{\text{def}}{=} (\lambda \overline{x : \tau}. \Lambda \overline{t : \kappa}. e_2) \bar{e}_1 \bar{\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} \\
 \text{pack}\langle \bar{\tau}, e \rangle_{\exists \overline{t : \kappa}. \tau_0} &\stackrel{\text{def}}{=} \begin{cases} \text{pack}\langle \tau, \text{pack}\langle \bar{\tau}', e \rangle_{\exists \overline{t' : \kappa'}. \tau_0} \rangle_{\exists \overline{t : \kappa}. \tau_0} & (\bar{\tau} = \tau \bar{\tau}', \overline{t : \kappa} = t : \kappa \overline{t' : \kappa'}) \\ e & (\bar{\tau} = \epsilon, \overline{t : \kappa} = \epsilon) \end{cases} \\
 (\text{unpack}\langle \overline{t : \kappa}, x : \tau \rangle = e_1 \text{ in } e_2) &\stackrel{\text{def}}{=} \begin{cases} \text{unpack}\langle t : \kappa, x_1 : \exists \overline{t' : \kappa'}. \tau \rangle = e_1 \text{ in } & (\overline{t : \kappa} = t : \kappa \overline{t' : \kappa'}) \\ \text{unpack}\langle \overline{t' : \kappa'}, x_2 : \tau \rangle = x_1 \text{ in } e_2 & \\ \text{let } x : \tau = e_1 \text{ in } e_2 & (\overline{t : \kappa} = \epsilon) \end{cases}
 \end{aligned}$$

Kinding:

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

$$\begin{array}{c}
 \frac{\Gamma(t) = \kappa}{\Gamma \vdash t : \kappa} \quad \frac{\Gamma \vdash \tau_1 : \Omega \quad \Gamma \vdash \tau_2 : \Omega}{\Gamma \vdash \tau_1 \rightarrow \tau_2 : \Omega} \quad \frac{\bigwedge_l \Gamma \vdash \tau_l : \Omega}{\Gamma \vdash \{\bar{l} : \tau_l\} : \Omega} \\
 \frac{\Gamma, t : \kappa \vdash \tau : \Omega}{\Gamma \vdash \forall t : \kappa. \tau : \Omega} \quad \frac{\Gamma, t : \kappa \vdash \tau : \Omega}{\Gamma \vdash \exists t : \kappa. \tau : \Omega} \quad \frac{\Gamma, t : \kappa_1 \vdash \tau : \kappa_2}{\Gamma \vdash \lambda t : \kappa_1. \tau : \kappa_1 \rightarrow \kappa_2} \quad \frac{\Gamma \vdash \tau_1 : \kappa_2 \rightarrow \kappa \quad \Gamma \vdash \tau_2 : \kappa_2}{\Gamma \vdash \tau_1 \tau_2 : \kappa}
 \end{array}$$

Type equivalence:

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

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

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

$$\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \Omega \quad \Gamma \vdash \tau'_1 \equiv \tau'_2 : \Omega}{\Gamma \vdash \tau_1 \rightarrow \tau'_1 \equiv \tau_2 \rightarrow \tau'_2 : \Omega} \quad \frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau_2 : \Omega}{\Gamma \vdash \forall t : \kappa. \tau_1 \equiv \forall t : \kappa. \tau_2 : \Omega}$$

$$\frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau_2 : \kappa'}{\Gamma \vdash \lambda t : \kappa. \tau_1 \equiv \lambda t : \kappa. \tau_2 : \kappa \rightarrow \kappa'} \quad \frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa' \rightarrow \kappa \quad \Gamma \vdash \tau'_1 \equiv \tau'_2 : \kappa'}{\Gamma \vdash \tau_1 \tau'_1 \equiv \tau_2 \tau'_2 : \kappa}$$

Typing:

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

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

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

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

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

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

Reduction:

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

$$\boxed{e \Rightarrow e'}$$

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

$$\frac{}{\text{unpack}\langle t : \kappa, x : \tau \rangle = \text{pack}\langle \tau_t, v \rangle_{\tau_{\exists}} \text{ in } e \Rightarrow e[t \leftarrow \tau_t][x \leftarrow v]} \quad \frac{e \Rightarrow e'}{C[e] \Rightarrow C[e']}$$

Equivalence:

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

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

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

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

$$\frac{\Gamma, t : \kappa \vdash \tau_1 \equiv \tau'_1 : \Omega \quad \Gamma \vdash \tau_t : \kappa \quad \Gamma \vdash e_1 : \tau_1[t \leftarrow \tau_t] \quad \Gamma, t : \kappa, x : \tau_1 \vdash e_2 : \tau}{\Gamma \vdash \text{unpack}\langle t : \kappa, x : \tau'_1 \rangle = \text{pack}\langle \tau_t, e_1 \rangle_{\exists t : \kappa. \tau_1} \text{ in } e_2 \equiv e_2[t \leftarrow \tau_t][x \leftarrow e_1] : \tau}$$

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

$$\begin{array}{c}
\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} \quad \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} \quad \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 \rightarrow \tau'} \quad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau' \rightarrow \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 \{l = e_{l,1}\} \equiv \{l = e_{l,2}\} : \{\overline{l} : \tau_l\}} \quad \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)} \quad \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 \text{pack}\langle \tau'_1, e_1 \rangle_{\exists t : \kappa. \tau_1} \equiv \text{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 \text{unpack}\langle t : \kappa, x : \tau'_1 \rangle = e'_1 \text{ in } e_1 \equiv \text{unpack}\langle t : \kappa, x : \tau'_2 \rangle = e'_2 \text{ in } e_2 : \tau}
\end{array}$$

### 3.1.2 Syntax

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

### 3.1.3 Signature

$\Sigma ::=$	$[\tau]$	(anonymous value declaration)
	$\mid [= \tau : \kappa]$	(anonymous type declaration)
	$\mid [= \Sigma]$	(anonymous signature declaration)
	$\mid \{\overline{l}_X : \Sigma\}$	(structural signature)

Atomic Signature:

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

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

NotAtomic( $\Sigma$ )

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

Admissible kinding:

$\Gamma \vdash \tau : \kappa$

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

Admissible type equivalence:

$\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa$

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

Admissible typing:

$\Gamma \vdash e : \tau$

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

Admissible equivalence:

$\Gamma \vdash e_1 \equiv e_2 : \tau$

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

## 3.1.4 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:

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

Path:

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

Use M-Dot.

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

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

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

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



## 3.2 F-ing modules

[RRD14]

### 3.2.1 Internal Language

See 第 3.1.1 小節.

### 3.2.2 Syntax

$X$	$::=$	$\dots$	(identifier)
$K$	$::=$	$\dots$	(kind)
$T$	$::=$	$\dots \mid P$	(type)
$E$	$::=$	$\dots \mid P$	(expression)
$P$	$::=$	$M$	(path)
$M$	$::=$	$X$	(identifier)
		$\mid \{B\}$	(bindings)
		$\mid M.X$	(projection)
		$\mid \text{fun } X : S \Rightarrow M$	(functor)
		$\mid X X$	(functor application)
		$\mid X :> S$	(sealing)
$B$	$::=$	$\text{val } X = E$	(value binding)
		$\mid \text{type } X = T$	(type binding)
		$\mid \text{module } X = M$	(module binding)
		$\mid \text{signature } X = S$	(signature binding)
		$\mid \text{include } M$	(module including)
		$\mid \epsilon$	(empty binding)
		$\mid B; B$	(binding concatenation)
$S$	$::=$	$P$	(signature path)
		$\mid \{D\}$	(declarations)
		$\mid (X : S) \rightarrow S$	((generative) functor signature)
		$\mid S \text{ where type } \bar{X} = T$	(bounded signature)
$D$	$::=$	$\text{val } X : T$	(value declaration)
		$\mid \text{type } X = T$	(type binding)
		$\mid \text{type } X : K$	(type declaration)
		$\mid \text{module } X : S$	(module declaration)
		$\mid \text{signature } X = S$	(signature binding)
		$\mid \text{include } S$	(signature including)
		$\mid \epsilon$	(empty declaration)
		$\mid D; D$	(declaration concatenation)

### 3.2.3 Signature

$\Xi$	$::=$	$\overline{\exists t : \kappa. \Sigma}$	(abstract signature)
$\Sigma$	$::=$	$[\tau]$	(atomic value declaration)
		$\mid [= \tau : \kappa]$	(atomic type declaration)
		$\mid [= \Xi]$	(atomic signature declaration)
		$\mid \{\overline{l_X : \Sigma}\}$	(structure signature)
		$\mid \forall t : \kappa. \Sigma \rightarrow \Xi$	(functor signature)

Atomic Signature:

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

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

$\text{NotAtomic}(\Sigma)$

$$\frac{}{\text{NotAtomic}(\{l_X : \Sigma\})} \quad \frac{}{\text{NotAtomic}(\forall t : \kappa. \Sigma \rightarrow \Xi)}$$

Admissible kinding:

$\Gamma \vdash \tau : \kappa$

$$\begin{aligned}
&\frac{\Gamma \vdash \tau : \Omega}{\Gamma \vdash [\tau] : \Omega} \text{ K-A-Val} \\
&\frac{\Gamma \vdash \tau : \kappa}{\Gamma \vdash [= \tau : \kappa] : \Omega} \text{ K-A-Typ} \\
&\frac{\Gamma \vdash \Xi : \Omega}{\Gamma \vdash [= \Xi] : \Omega} \text{ K-A-Sig}
\end{aligned}$$

Admissible type equivalence:

$\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa$

$$\begin{aligned}
&\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \Omega}{\Gamma \vdash [\tau_1] \equiv [\tau_2] : \Omega} \text{ T-Eq-Cong-A-Val} \\
&\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}{\Gamma \vdash [= \tau_1 : \kappa] \equiv [= \tau_2 : \kappa] : \Omega} \text{ T-Eq-Cong-A-Typ} \\
&\frac{\Gamma \vdash \Xi_1 \equiv \Xi_2 : \Omega}{\Gamma \vdash [= \Xi_1] \equiv [= \Xi_2] : \Omega} \text{ T-Eq-Cong-A-Sig}
\end{aligned}$$

Admissible typing:

$\Gamma \vdash e : \tau$

$$\begin{aligned}
&\frac{\Gamma \vdash e : \tau}{\Gamma \vdash [e] : [\tau]} \text{ T-A-Val} \\
&\frac{\Gamma \vdash \tau : \kappa}{\Gamma \vdash [\tau : \kappa] : [= \tau : \kappa]} \text{ T-A-Typ} \\
&\frac{\Gamma \vdash \Xi : \Omega}{\Gamma \vdash [\Xi] : [= \Xi]} \text{ T-A-Sig}
\end{aligned}$$

Admissible equivalence:

$\Gamma \vdash e_1 \equiv e_2 : \tau$

$$\begin{aligned}
&\frac{\Gamma \vdash e : \tau}{\Gamma \vdash [e]. \text{val} \equiv e : \tau} \text{ Eq-}\beta\text{-A-Val} \quad \frac{\Gamma \vdash e : [\tau]}{\Gamma \vdash [e. \text{val}] \equiv e : [\tau]} \text{ Eq-}\eta\text{-A-Val} \quad \frac{\Gamma \vdash e_1 \equiv e_2 : \tau}{\Gamma \vdash [e_1] \equiv [e_2] : [\tau]} \text{ Eq-Cong-A-Val} \\
&\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \kappa}{\Gamma \vdash [\tau_1 : \kappa] \equiv [\tau_2 : \kappa] : [= \tau_1 : \kappa]} \text{ Eq-Cong-A-Typ} \\
&\frac{\Gamma \vdash \Xi_1 \equiv \Xi_2 : \Omega}{\Gamma \vdash [\Xi_1] \equiv [\Xi_2] : [= \Xi_1]} \text{ Eq-Cong-A-Sig}
\end{aligned}$$

## 3.2.4 (Generative) Elaboration

Signature:

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

$$\begin{array}{c}
\frac{\Gamma \vdash P : [= \Xi] \rightsquigarrow e}{\Gamma \vdash P \rightsquigarrow \Xi} \text{ S-Path} \\
\frac{\Gamma \vdash D \rightsquigarrow \Xi}{\Gamma \vdash \{D\} \rightsquigarrow \Xi} \text{ S-Struct} \\
\frac{\Gamma \vdash S_1 \rightsquigarrow \exists \overline{t} : \overline{\kappa}. \Sigma \quad \Gamma, \overline{t} : \overline{\kappa}, x_X : \Sigma \vdash S_2 \rightsquigarrow \Xi}{\Gamma \vdash (X : S_1) \rightarrow S_2 \rightsquigarrow \forall \overline{t} : \overline{\kappa}. \Sigma \rightarrow \Xi} \text{ S-Funct} \\
\frac{\Gamma \vdash S \rightsquigarrow \exists \overline{t}_1 : \overline{\kappa}_1 \overline{t} : \overline{\kappa}_2 \overline{t}_2 : \overline{\kappa}_2. \Sigma \quad \Sigma. \overline{l}_X = [= \overline{t} : \overline{\kappa}] \quad \Gamma \vdash T : \overline{\kappa} \rightsquigarrow \tau}{\Gamma \vdash S \text{ where type } \overline{X} = T \rightsquigarrow \exists \overline{t}_1 : \overline{\kappa}_1 \overline{t}_2 : \overline{\kappa}_2. \Sigma [\overline{t} \leftarrow \tau]} \text{ S-Where-Typ}
\end{array}$$

Declarations:

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

$$\begin{array}{c}
\frac{\Gamma \vdash T : \Omega \rightsquigarrow \tau}{\Gamma \vdash \text{val } X : T \rightsquigarrow \{l_X : [\tau]\}} \text{ D-Val} \\
\frac{\Gamma \vdash T : \overline{\kappa} \rightsquigarrow \tau}{\Gamma \vdash \text{type } X = T \rightsquigarrow \{l_X : [= \tau : \overline{\kappa}]\}} \text{ D-Typ-Eq} \\
\frac{\Gamma \vdash K \rightsquigarrow \overline{\kappa}}{\Gamma \vdash \text{type } X : K \rightsquigarrow \exists \overline{t} : \overline{\kappa}. \{l_X : [= \overline{t} : \overline{\kappa}]\}} \text{ D-Typ} \\
\frac{\Gamma \vdash S \rightsquigarrow \exists \overline{t} : \overline{\kappa}. \Sigma}{\Gamma \vdash \text{module } X : S \rightsquigarrow \exists \overline{t} : \overline{\kappa}. \{l_X : \Sigma\}} \text{ D-Mod} \\
\frac{\Gamma \vdash S \rightsquigarrow \Xi}{\Gamma \vdash \text{signature } X = S \rightsquigarrow \{l_X : [= \Xi]\}} \text{ D-Sig-Eq} \\
\frac{\Gamma \vdash S \rightsquigarrow \exists \overline{t} : \overline{\kappa}. \{\overline{l}_X : \Sigma\}}{\Gamma \vdash \text{include } S \rightsquigarrow \exists \overline{t} : \overline{\kappa}. \{\overline{l}_X : \Sigma\}} \text{ D-Incl} \\
\frac{}{\Gamma \vdash \epsilon \rightsquigarrow \{\}} \text{ D-Emt} \\
\frac{\{\overline{l}_{X_1}\} \cap \{\overline{l}_{X_2}\} = \emptyset \quad \Gamma \vdash D_1 \rightsquigarrow \exists \overline{t}_1 : \overline{\kappa}_1. \{\overline{l}_{X_1} : \Sigma_1\} \quad \Gamma, \overline{t}_1 : \overline{\kappa}_1, x_{X_1} : \Sigma_1 \vdash D_2 \rightsquigarrow \exists \overline{t}_2 : \overline{\kappa}_2. \{\overline{l}_{X_2} : \Sigma_2\}}{\Gamma \vdash D_1; D_2 \rightsquigarrow \exists \overline{t}_1 : \overline{\kappa}_1 \overline{t}_2 : \overline{\kappa}_2. \{\overline{l}_{X_1} : \Sigma_1 \overline{l}_{X_2} : \Sigma_2\}} \text{ D-Seq}
\end{array}$$

Matching:

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

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

Subtyping:

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

$$\begin{array}{c}
\frac{\Gamma \vdash \tau_1 \leq \tau_2 \rightsquigarrow e}{\Gamma \vdash [\tau_1] \leq [\tau_2] \rightsquigarrow \lambda x : [\tau_1]. [e (x. \text{val})]} \text{ U-Val} \\
\frac{\Gamma \vdash \tau_1 \equiv \tau_2 : \overline{\kappa}}{\Gamma \vdash [= \tau_1 : \overline{\kappa}] \leq [= \tau_2 : \overline{\kappa}] \rightsquigarrow \lambda x : [= \tau_1 : \overline{\kappa}]. x} \text{ U-Typ} \\
\frac{\Gamma \vdash \Xi_1 \leq \Xi_2 \rightsquigarrow e_1 \quad \Gamma \vdash \Xi_2 \leq \Xi_1 \rightsquigarrow e_2}{\Gamma \vdash [= \Xi_1] \leq [= \Xi_2] \rightsquigarrow \lambda x : [= \Xi_1]. [\Xi_2]} \text{ U-Sig}
\end{array}$$

$$\begin{array}{c}
\frac{\bigwedge_l \Gamma \vdash \Sigma_{l_1} \leq \Sigma_{l_2} \rightsquigarrow e_l}{\Gamma \vdash \{\overline{l} : \Sigma_{l_1}, \overline{l'} : \Sigma'\} \leq \{\overline{l} : \Sigma_{l_2}\} \rightsquigarrow \lambda x : \{\overline{l} : \Sigma_{l_1}, \overline{l'} : \Sigma'\}. \{\overline{l} = e_l(x.l)\}} \text{U-Struct} \\
\frac{\Gamma, \overline{t_2} : \overline{\kappa_2} \vdash \Sigma_2 \leq \exists \overline{t_1} : \overline{\kappa_1}. \Sigma_1 \uparrow \overline{\tau} \rightsquigarrow e_1 \quad \Gamma, \overline{t_2} : \overline{\kappa_2} \vdash \Sigma_1[\overline{t_1} \leftarrow \overline{\tau}] \leq \Sigma_2 \rightsquigarrow e_2}{\Gamma \vdash \forall \overline{t_1} : \overline{\kappa_1}. \Sigma_1 \rightarrow \Sigma_2 \leq \exists \overline{t_2} : \overline{\kappa_2}. \Sigma_2 \rightarrow \Sigma_2 \rightsquigarrow \lambda x_1 : (\forall \overline{t_1} : \overline{\kappa_1}. \Sigma_1 \rightarrow \Sigma_2). \lambda x_2 : \Sigma_2. e_2(x_1 \overline{\tau}(e_1 x_2))} \text{U-Funct} \\
\frac{\Gamma, \overline{t_1} : \overline{\kappa_1} \vdash \Sigma_1 \leq \exists \overline{t_2} : \overline{\kappa_2}. \Sigma_2 \uparrow \overline{\tau} \rightsquigarrow e}{\Gamma \vdash \exists \overline{t_1} : \overline{\kappa_1}. \Sigma_1 \leq \exists \overline{t_2} : \overline{\kappa_2}. \Sigma_2 \rightsquigarrow \lambda x_1 : (\exists \overline{t_1} : \overline{\kappa_1}. \Sigma_1). \text{unpack}(\overline{t_1} : \overline{\kappa_1}, x_1' : \Sigma_1) = x_1 \text{ in pack}(\overline{\tau}, e x_1')_{\exists \overline{t_2} : \overline{\kappa_2}. \Sigma_2}} \text{U-Abs}
\end{array}$$

Module:

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

$$\begin{array}{c}
\frac{\Gamma(x_X) = \Sigma}{\Gamma \vdash X : \Sigma \rightsquigarrow x_X} \text{M-Var} \\
\frac{\Gamma \vdash B : \Xi \rightsquigarrow e}{\Gamma \vdash \{B\} : \Xi \rightsquigarrow e} \text{M-Struct} \\
\frac{\Gamma \vdash M : \exists \overline{t} : \overline{\kappa}. \{\overline{l_X} : \Sigma, \overline{l_{X'}} : \Sigma'\} \rightsquigarrow e}{\Gamma \vdash M.X : \exists \overline{t} : \overline{\kappa}. \Sigma \rightsquigarrow \text{unpack}(\overline{t} : \overline{\kappa}, x : \{\overline{l_X} : \Sigma, \overline{l_{X'}} : \Sigma'\}) = e \text{ in pack}(\overline{t}, x.l_X)_{\exists \overline{t} : \overline{\kappa}. \Sigma}} \text{M-Dot} \\
\frac{\Sigma \vdash S \rightsquigarrow \exists \overline{t} : \overline{\kappa}. \Sigma \quad \Gamma, \overline{t} : \overline{\kappa}, x_X : \Sigma \vdash M : \Xi \rightsquigarrow e}{\Gamma \vdash \text{fun } X : S \Rightarrow M : \forall \overline{t} : \overline{\kappa}. \Sigma \rightarrow \Xi \rightsquigarrow \Lambda \overline{t} : \overline{\kappa}. \lambda x_X : \Sigma. e} \text{M-Funct} \\
\frac{\Gamma(x_{X_1}) = \forall \overline{t} : \overline{\kappa}. \Sigma' \rightarrow \Xi \quad \Gamma(x_{X_2}) = \Sigma \quad \Gamma \vdash \Sigma \leq \exists \overline{t} : \overline{\kappa}. \Sigma' \uparrow \overline{\tau} \rightsquigarrow e}{\Gamma \vdash X_1 X_2 : \Xi[\overline{t} \leftarrow \overline{\tau}] \rightsquigarrow x_{X_1} \overline{\tau}(e x_{X_2})} \text{M-App} \\
\frac{\Gamma(x_X) = \Sigma \quad \Gamma \vdash S \rightsquigarrow \exists \overline{t} : \overline{\kappa}. \Sigma' \quad \Gamma \vdash \Sigma \leq \exists \overline{t} : \overline{\kappa}. \Sigma' \uparrow \overline{\tau} \rightsquigarrow e}{\Gamma \vdash X : > S : \exists \overline{t} : \overline{\kappa}. \Sigma' \rightsquigarrow \text{pack}(\overline{\tau}, e x_X)_{\exists \overline{t} : \overline{\kappa}. \Sigma'}} \text{M-Seal}
\end{array}$$

Bindings:

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

$$\begin{array}{c}
\frac{\Gamma \vdash E : \tau \rightsquigarrow e}{\Gamma \vdash \text{val } X = E : \{\overline{l_X} : [\tau]\} \rightsquigarrow \{\overline{l_X} = [e]\}} \text{B-Val} \\
\frac{\Gamma \vdash T : \kappa \rightsquigarrow \tau}{\Gamma \vdash \text{type } X = T : \{\overline{l_X} : [= \tau : \kappa]\} \rightsquigarrow \{\overline{l_X} = [\tau : \kappa]\}} \text{B-Typ} \\
\frac{\Gamma \vdash M : \exists \overline{t} : \overline{\kappa}. \Sigma \rightsquigarrow e \quad \text{NotAtomic}(\Sigma)}{\Gamma \vdash \text{module } X = M : \exists \overline{t} : \overline{\kappa}. \{\overline{l_X} : \Sigma\} \rightsquigarrow \text{unpack}(\overline{t} : \overline{\kappa}, x : \Sigma) = e \text{ in pack}(\overline{t}, \{\overline{l_X} = x\})_{\exists \overline{t} : \overline{\kappa}. \{\overline{l_X} : \Sigma\}}} \text{B-Mod} \\
\frac{\Gamma \vdash S \rightsquigarrow \Xi}{\Gamma \vdash \text{signature } X = S : \{\overline{l_X} : [= \Xi]\} \rightsquigarrow \{\overline{l_X} = [\Xi]\}} \text{B-Sig} \\
\frac{\Gamma \vdash M : \exists \overline{t} : \overline{\kappa}. \{\overline{l_X} : \Sigma\} \rightsquigarrow e}{\Gamma \vdash \text{include } M : \exists \overline{t} : \overline{\kappa}. \{\overline{l_X} : \Sigma\} \rightsquigarrow e} \text{B-Incl} \\
\frac{}{\Gamma \vdash \epsilon : \{\} \rightsquigarrow \{\}} \text{B-Emt} \\
\frac{\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 \quad \Gamma \vdash B_1 : \exists \overline{t_1} : \overline{\kappa_1}. \{\overline{l_{X_1}} : \Sigma_1\} \rightsquigarrow e_1 \quad \Sigma = \{\overline{l'_{X_1}} : \Sigma'_1, \overline{l_{X_2}} : \Sigma_2\} \quad \Gamma, \overline{t_1} : \overline{\kappa_1}, x_{X_1} : \Sigma_1 \vdash B_2 : \exists \overline{t_2} : \overline{\kappa_2}. \{\overline{l_{X_2}} : \Sigma_2\} \rightsquigarrow e_2}{\Gamma \vdash B_1; B_2 : \exists \overline{t_1} : \overline{\kappa_1} \overline{t_2} : \overline{\kappa_2}. \Sigma \rightsquigarrow \text{unpack}(\overline{t_1} : \overline{\kappa_1}, x_1) = e_1 \text{ in } \text{unpack}(\overline{t_2} : \overline{\kappa_2}, x_2) = (\text{let } x_{X_1} : \Sigma_1 = x_1.l_{X_1} \text{ in } e_2) \text{ in } \text{pack}(\overline{t_1} \overline{t_2}, \{\overline{l'_{X_1}} = x_1.l'_{X_1}, \overline{l_{X_2}} = x_2.l_{X_2}\})_{\exists \overline{t_1} : \overline{\kappa_1} \overline{t_2} : \overline{\kappa_2}. \Sigma}} \text{B-Seq}
\end{array}$$

Path:

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

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

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

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

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

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

### 3.2.5 Modules as First-Class Values

$$\begin{aligned} T &::= \dots \mid \text{pack } S \\ E &::= \dots \mid \text{pack } M : S \\ M &::= \dots \mid \text{unpack } E : S \end{aligned}$$

Rootedness:

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

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

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:

$$\begin{aligned} &\frac{\text{norm}_0(\tau) = \tau'}{\text{norm}([\tau]) = [\tau']} \\ &\frac{}{\text{norm}([= \tau : \kappa]) = [= \tau : \kappa]} \\ &\frac{\text{norm}(\Xi) = \Xi'}{\text{norm}([= \Xi]) = [= \Xi']} \\ &\frac{\bigwedge_X \text{norm}(\Sigma_X) = \Sigma'_X}{\text{norm}(\{\overline{l}_X : \Sigma_X\}) = \{\overline{l}_X : \Sigma'_X\}} \\ &\frac{\text{sort}_{\leq_{\Sigma'}}(\overline{t : \kappa}) = \overline{t' : \kappa'} \quad \text{norm}(\Sigma) = \Sigma' \quad \text{norm}(\Xi) = \Xi'}{\text{norm}(\forall \overline{t : \kappa}. \Sigma \rightarrow \Xi) = \forall \overline{t' : \kappa'}. \Sigma' \rightarrow \Xi'} \\ &\frac{\text{sort}_{\leq_{\Sigma'}}(\overline{t : \kappa}) = \overline{t' : \kappa'} \quad \text{norm}(\Sigma) = \Sigma'}{\text{norm}(\exists \overline{t : \kappa}. \Sigma) = \exists \overline{t' : \kappa'}. \Sigma'} \end{aligned}$$

Type:

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

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

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:

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

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

### 3.2.6 Elaboration with Applicative Functor

$$\begin{aligned} S &::= \dots \\ &| (X : S) \Rightarrow S \quad (\text{applicative functor signature}) \end{aligned}$$

$$\begin{aligned} \varphi &::= I && (\text{impure effect}) \\ &| P && (\text{pure effect}) \\ \Sigma &::= \dots \\ &| \overline{\{l_X : \Sigma\}} \\ &| \forall t : \kappa. \Sigma \rightarrow_I \Xi \quad (\text{generative functor signature}) \\ &| \forall t : \kappa. \Sigma \rightarrow_P \Sigma \quad (\text{applicative functor signature}) \end{aligned}$$

Abbreviation:

$$\begin{aligned} \tau_1 \rightarrow_\varphi \tau_2 &\stackrel{\text{def}}{=} \tau_1 \rightarrow \{l_\varphi : \tau_2\} \\ \lambda_\varphi x : \tau. e &\stackrel{\text{def}}{=} \lambda x : \tau. \{l_\varphi = e\} \\ (e_1 e_2)_\varphi &\stackrel{\text{def}}{=} (e_1 e_2).l_\varphi \\ \Gamma^\varphi &\stackrel{\text{def}}{=} \begin{cases} \cdot & (\varphi = I) \\ \Gamma & (\varphi = P) \end{cases} \\ \text{tyenv}(\Gamma) &\stackrel{\text{def}}{=} \begin{cases} \text{tyenv}(\Gamma') \, t : \kappa & (\Gamma = \Gamma', t : \kappa) \\ \text{tyenv}(\Gamma') & (\Gamma = \Gamma', x : \tau) \\ \epsilon & (\Gamma = \cdot) \end{cases} \\ \forall_P \Gamma. \tau_0 &\stackrel{\text{def}}{=} \begin{cases} \forall_P \Gamma'. \forall t : \kappa. \tau_0 & (\Gamma = \Gamma', t : \kappa) \\ \forall_P \Gamma'. \tau \rightarrow_P \tau_0 & (\Gamma = \Gamma', x : \tau) \\ \tau_0 & (\Gamma = \cdot) \end{cases} \\ \Lambda_P \Gamma. e &\stackrel{\text{def}}{=} \begin{cases} \Lambda_P \Gamma'. \Lambda t : \kappa. e & (\Gamma = \Gamma', t : \kappa) \\ \Lambda_P \Gamma'. \lambda_P x : \tau. e & (\Gamma = \Gamma', x : \tau) \\ e & (\Gamma = \cdot) \end{cases} \\ (e \, \Gamma)_P &\stackrel{\text{def}}{=} \begin{cases} (e \, \Gamma')_P \, t & (\Gamma = \Gamma', t : \kappa) \\ ((e \, \Gamma')_P \, x)_P & (\Gamma = \Gamma', x : \tau) \\ e & (\Gamma = \cdot) \end{cases} \end{aligned}$$

Effect combining:

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

$$\overline{\varphi \vee \varphi} = \overline{\varphi} \quad \overline{I \vee P} = \overline{I} \quad \overline{P \vee I} = \overline{I}$$

Subeffects:

$$\varphi_1 \leq \varphi_2$$

$$\overline{\varphi \leq \varphi} \text{ F-Refl} \quad \overline{P \leq I} \text{ F-Sub}$$

Signature:

$$\Gamma \vdash S \rightsquigarrow \Xi$$

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

$$\frac{\Gamma \vdash S_1 \rightsquigarrow \exists \overline{t_1 : \kappa_1}. \Sigma_1 \quad \Gamma, \overline{t_1 : \kappa_1}, x_X : \Sigma_1 \vdash S_2 \rightsquigarrow \exists \overline{t_2 : \kappa_2}. \Sigma_2}{\Gamma \vdash (X : S_1) \Rightarrow S_2 \rightsquigarrow \exists \overline{t'_2 : \kappa_1} \rightarrow \kappa_2. \forall \overline{t_1 : \kappa_1}. \Sigma_1 \rightarrow_P \Sigma_2[t_2 \leftarrow t'_2 \overline{t_1}]} \text{ S-Funct-P}$$

Subtyping:

$$\Gamma \vdash \Xi_1 \leq \Xi_2 \rightsquigarrow e$$

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

Module:

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

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

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

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

$$\frac{\Sigma \vdash S \rightsquigarrow \exists \overline{t : \kappa}. \Sigma \quad \Gamma, \overline{t : \kappa}, x_X : \Sigma \vdash M :_I \Xi \rightsquigarrow e}{\Gamma \vdash \text{fun } X : S \Rightarrow M :_P \forall \overline{t : \kappa}. \Sigma \rightarrow_I \Xi \rightsquigarrow \Lambda_P \Gamma. \Lambda \overline{t : \kappa}. \lambda_1 x_X : \Sigma. e} \text{ M-Funct-I}$$

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

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

$$\frac{\overline{t_{\Gamma} : \kappa_{\Gamma}} = \text{tyenv}(\Gamma) \quad \Gamma(x_X) = \Sigma \quad \Gamma \vdash S \rightsquigarrow \exists \overline{t : \kappa}. \Sigma' \quad \Gamma \vdash \Sigma \leq \exists \overline{t : \kappa}. \Sigma' \uparrow \overline{\tau} \rightsquigarrow e}{\Gamma \vdash X :> S :_P \exists \overline{t' : \overline{t_{\Gamma} : \kappa_{\Gamma}}} \rightarrow \kappa. \Sigma'[t \leftarrow t' \overline{t_{\Gamma}}] \rightsquigarrow \text{pack} \langle \lambda \overline{t_{\Gamma} : \kappa_{\Gamma}}. \tau, \Lambda_P \Gamma. e x_X \rangle} \text{ M-Seal}$$

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

定理 30 (Typing for module elaboration).

- $\Gamma \vdash M :_I \Xi \rightsquigarrow e$  ならば,  $\Gamma \vdash e : \Xi$ .
- $\Gamma \vdash M :_P \exists \overline{t : \kappa}. \Sigma \rightsquigarrow e$  ならば,  $\cdot \vdash e : \exists \overline{t : \kappa}. \forall_P \Gamma. \Sigma$ .

□

Bindings:

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

$$\begin{array}{c}
\frac{\Gamma \vdash E : \tau \rightsquigarrow e}{\Gamma \vdash \text{val } X = E :_{\mathbf{P}} \{l_X : [\tau]\} \rightsquigarrow \Lambda_{\mathbf{P}}\Gamma.\{l_X = e\}} \text{ B-Val} \\
\frac{\Gamma \vdash T : \kappa \rightsquigarrow \tau}{\Gamma \vdash \text{type } X = T :_{\mathbf{P}} \{l_X : [\tau : \kappa]\} \rightsquigarrow \Lambda_{\mathbf{P}}\Gamma.\{l_X = [\tau : \kappa]\}} \text{ B-Typ} \\
\frac{\Gamma \vdash M :_{\varphi} \exists \overline{t} : \overline{\kappa}. \Sigma \rightsquigarrow e \quad \text{NotAtomic}(\Sigma)}{\Gamma \vdash \text{module } X = M :_{\varphi} \exists \overline{t} : \overline{\kappa}. \{l_X : \Sigma\} \rightsquigarrow \text{unpack}\langle \overline{t} : \overline{\kappa}, x \rangle = e \text{ in pack}\langle \overline{t}, \Lambda_{\mathbf{P}}\Gamma^{\varphi}.\{l_X = x \Gamma^{\varphi}\} \rangle} \text{ B-Mod} \\
\frac{\Gamma \vdash S \rightsquigarrow \Xi}{\Gamma \vdash \text{signature } X = S :_{\mathbf{P}} \{l_X : [\Xi]\} \rightsquigarrow \Lambda_{\mathbf{P}}\Gamma.\{l_X = [\Xi]\}} \text{ B-Sig} \\
\frac{\Gamma \vdash M :_{\varphi} \exists \overline{t} : \overline{\kappa}. \{l_X : \Sigma\} \rightsquigarrow e}{\Gamma \vdash \text{include } M :_{\varphi} \exists \overline{t} : \overline{\kappa}. \{l_X : \Sigma\} \rightsquigarrow e} \text{ B-Incl} \\
\frac{}{\Gamma \vdash \epsilon :_{\mathbf{P}} \{\} \rightsquigarrow \Lambda_{\mathbf{P}}\Gamma.\{\}} \text{ B-Emt} \\
\frac{\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 \quad \Gamma \vdash B_1 :_{\varphi_1} \exists \overline{t_1} : \overline{\kappa_1}. \{l_{X_1} : \Sigma_1\} \rightsquigarrow e_1 \quad \Sigma = \{\overline{l'_{X_1}} : \Sigma'_1, \overline{l_{X_2}} : \Sigma_2\} \quad \Gamma, \overline{t_1} : \overline{\kappa_1}, \overline{x_{X_1}} : \overline{\Sigma_1} \vdash B_2 :_{\varphi_2} \exists \overline{t_2} : \overline{\kappa_2}. \{l_{X_2} : \Sigma_2\} \rightsquigarrow e_2}{\Gamma \vdash B_1; B_2 :_{\varphi_1 \vee \varphi_2} \exists \overline{t_1} : \overline{\kappa_1} \overline{t_2} : \overline{\kappa_2}. \Sigma \rightsquigarrow \text{unpack}\langle \overline{t_1} : \overline{\kappa_1}, x_1 \rangle = e_1 \text{ in } \text{unpack}\langle \overline{t_2} : \overline{\kappa_2}, x_2 \rangle = (\text{let } x_{X_1} = \Lambda_{\mathbf{P}}\Gamma^{\varphi_1 \vee \varphi_2}. (x_1 \Gamma^{\varphi_1})_{\mathbf{P}}. l_{X_1} \text{ in } e_2) \text{ in } \text{pack}\langle \overline{t_1} \overline{t_2}, \Lambda_{\mathbf{P}}\Gamma^{\varphi_1 \vee \varphi_2}. \text{let } x_{X_1} = (x_1 \Gamma^{\varphi_1})_{\mathbf{P}}. l_{X_1} \text{ in } \{l'_{X_1} = (x_1 \Gamma^{\varphi_1})_{\mathbf{P}}. l'_{X_1}, l_{X_2} = (x_2 (\Gamma, \overline{t_1} : \overline{\kappa_1}, \overline{x_{X_1}} : \overline{\Sigma_1})^{\varphi_2})_{\mathbf{P}}. l_{X_2}\} \rangle} \text{ B-Seq}
\end{array}$$

Path:

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

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

Expression:

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

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





## 第 4 章

# Control Operators



## 第 5 章

# Coherent Implicit Parameter



## 第 6 章

# Polymorphic Record Type



## 第 7 章

# Type Checking and Inference





## 第 8 章

# Static Memory Management and Regions



## 第 9 章

# Dynamic Memory Management and Gabage Collection



## 第 10 章

# I/O Management and Concurrency



## 第 11 章

# Code Generation and Virtual Machines





## 第 12 章

# Program Stability and Compatibility



## 第 13 章

# Program Separation and Linking



## 第 14 章

# Syntax and Parsing

## 14.1 WIP: Parsing by LR Method

[Knu65]

## 14.2 Syntax and Semantics of PEG

[For02], [For04]

### 14.2.1 Syntax

$e ::=$	$\epsilon$	(epsilon)
	$  \sigma$	(terminal)
	$  A$	(non-terminal)
	$  ee$	(sequence)
	$  e / e$	(alternative)
	$  e^*$	(repetition)
	$  !e$	(not predicate)
$\sigma \in$	$\Sigma$	
$A \in$	$N$	

**定義 31.** PEG 文法とは、以下による組  $G = (\Sigma, N, R, e_0)$  のことである。

$\Sigma$  終端記号の集合.

$N$  非終端記号の集合.

$R$   $A \rightarrow e$  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.

□

### 14.2.2 Structured Semantics

$$\begin{array}{c}
 \overline{\langle \epsilon, x \rangle \rightarrow s(\epsilon)} \\
 \overline{\langle \sigma, \sigma x \rangle \rightarrow s(\sigma)} \quad \overline{\langle \sigma, \sigma' x \rangle \rightarrow f} \quad \overline{\langle \sigma, \epsilon \rangle \rightarrow f} \quad \overline{\langle A, x \rangle \rightarrow o} \\
 \overline{\langle e, x \rangle \rightarrow o} \\
 \overline{\langle e_1, x_1 x_2 y \rangle \rightarrow s(x_1)} \quad \overline{\langle e_2, x_2 y \rangle \rightarrow s(x_2)} \quad \overline{\langle e_1, x \rangle \rightarrow f} \quad \overline{\langle e_1, x_1 y \rangle \rightarrow s(x_1)} \quad \overline{\langle e_2, y \rangle \rightarrow f} \\
 \overline{\langle e_1 e_2, x_1 x_2 y \rangle \rightarrow s(x_1 x_2)} \quad \overline{\langle e_1 e_2, x \rangle \rightarrow f} \quad \overline{\langle e_1 e_2, x_1 y \rangle \rightarrow f} \\
 \overline{\langle e_1, xy \rangle \rightarrow s(x)} \quad \overline{\langle e_1, x \rangle \rightarrow f} \quad \overline{\langle e_2, x \rangle \rightarrow o} \\
 \overline{\langle e_1 / e_2, xy \rangle \rightarrow s(x)} \quad \overline{\langle e_1 / e_2, x \rangle \rightarrow o} \\
 \overline{\langle e, x_1 x_2 y \rangle \rightarrow s(x_1)} \quad \overline{\langle e^*, x_2 y \rangle \rightarrow s(x_2)} \quad \overline{\langle e, x \rangle \rightarrow f} \\
 \overline{\langle e^*, x_1 x_2 y \rangle \rightarrow s(x_1 x_2)} \quad \overline{\langle e^*, x \rangle \rightarrow s(\epsilon)} \\
 \overline{\langle e, x \rangle \rightarrow f} \quad \overline{\langle e, xy \rangle \rightarrow s(x)} \\
 \overline{\langle !e, x \rangle \rightarrow s(\epsilon)} \quad \overline{\langle !e, xy \rangle \rightarrow f}
 \end{array}$$

$$\llbracket (\Sigma, N, R, e_0) \rrbracket = \llbracket e_0 \rrbracket$$

$$\llbracket e \rrbracket = \{x \in \Sigma^* \mid \langle e, x \rangle \rightarrow s(x)\}$$



### 14.2.3 Equivalence

#### Abbreviations

$\& e = !(e)$	(and predicate)
$e^+ = ee^*$	(positive repetition)
$e^? = e/\epsilon$	(optional)

#### Associativity

$$\overline{\llbracket e_1/(e_2/e_3) \rrbracket} = \overline{\llbracket (e_1/e_2)/e_3 \rrbracket}$$

$$\overline{\llbracket e_1(e_2e_3) \rrbracket} = \overline{\llbracket (e_1e_2)e_3 \rrbracket}$$

#### Epsilon

$$\overline{\llbracket \epsilon/e \rrbracket} = \overline{\llbracket \epsilon \rrbracket}$$

$$\overline{\llbracket e\epsilon \rrbracket} = \overline{\llbracket e \rrbracket} \quad \overline{\llbracket \epsilon e \rrbracket} = \overline{\llbracket e \rrbracket}$$

#### Repetition

$$M ::= eM \mid \epsilon$$

$$\overline{\llbracket e^* \rrbracket} = \overline{\llbracket M \rrbracket}$$

### 14.2.4 Producing Analysis

$$s ::= 0 \mid 1, \quad o ::= s \mid f$$

- $\epsilon \rightarrow 0$
- $\sigma \rightarrow 1$
- $\sigma \rightarrow f$
- $A \leftarrow e \in R, \quad e \rightarrow o \text{ ならば } A \rightarrow o$
- $e_1 \rightarrow 0, \quad e_2 \rightarrow 0 \text{ ならば } e_1e_2 \rightarrow 0$
- $e_1 \rightarrow 1, \quad e_2 \rightarrow s \text{ ならば } e_1e_2 \rightarrow 1$
- $e_1 \rightarrow s, \quad e_2 \rightarrow 1 \text{ ならば } e_1e_2 \rightarrow 1$
- $e_1 \rightarrow f \text{ ならば } e_1e_2 \rightarrow f$
- $e_1 \rightarrow s, \quad e_2 \rightarrow f \text{ ならば } e_1e_2 \rightarrow f$
- $e_1 \rightarrow s \text{ ならば } e_1 / e_2 \rightarrow s$
- $e_1 \rightarrow f, \quad e_2 \rightarrow o \text{ ならば } e_1 / e_2 \rightarrow o$
- $e \rightarrow 1 \text{ ならば } e^* \rightarrow 1$
- $e \rightarrow f \text{ ならば } e^* \rightarrow f$
- $e \rightarrow s \text{ ならば } !e \rightarrow f$

- $e \rightarrow f$  ならば  $e \rightarrow 0$

定理 32.

- $\langle e, x \rangle \rightarrow s(\epsilon)$  ならば,  $e \rightarrow 0$
- $\langle e, xy \rangle \rightarrow s(x)$ ,  $x \neq \epsilon$  ならば,  $e \rightarrow 1$
- $\langle e, x \rangle \rightarrow f$  ならば,  $e \rightarrow f$

□

系 33.  $e \not\rightarrow 0$  ならば,  $\langle e, xy \rangle \not\rightarrow s(x)$  かつ  $\langle e, xy \rangle \not\rightarrow f$

□

## 14.3 Haskell Parsing with PEG

[Sim10]

### 14.3.1 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   ::= "\x\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 ::= "." | ":" | "::" | "=" | "\\ " | "<-" | "->" | "@" | "~" | "=>"

```

```

        tycon  ::= conid
        modid  ::= (conid "."*) conid
        qvarid ::= (modid "."?) varid
        qconid ::= (modid "."?) conid
        qtycon ::= (modid "."?) conid
        qvarsym ::= (modid "."?) varsym
        qconsym ::= (modid "."?) consym

decimal ::= digit+
octal   ::= octit+
hexadecimal ::= hexit+
integer ::= decimal
        | "0o" octal | "0O" octal
        | "0x" hexadecimal | "0X" hexadecimal
float   ::= decimal "." decimal exponent?
        | decimal exponent
exponent ::= ("e" | "E") ("+" | "-") decimal
char     ::= "'" (!("'" | "\\") graphic | " " | !"\\&" escape) "'"
string   ::= "\"" (!("\"" | "\\") graphic | " " | escape | gap)* "\""
escape   ::= "\\(charesc | ascii | decimal | "o" octal | "x" hexadecimal)
charesc  ::= "a" | "b" | "f" | "n" | "r" | "t" | "v" | "\\ " | "\" " | "' " | "&"
ascii    ::= "^" cntrl | "NUL" | "SOH" | "STX" | "ETX" | "EOT" | "ENQ" | "ACK" | "BEL" | "BS"
        | "HT" | "LF" | "VT" | "FF" | "CR" | "SO" | "SI" | "DLE" | "DC1" | "DC2" | "DC3"
        | "DC4" | "NAK" | "SYN" | "ETB" | "CAN" | "EM" | "SUB" | "ESC" | "FS" | "GS" | "RS"
        | "US" | "SP" | "DEL"
cntrl    ::= "A" | "B" | ... | "Z" | "@" | "[" | "\\ " | "]" | "^" | "_"
gap      ::= "\\ " whitechar+ "\\ "

```

### 14.3.2 Preprocess for Layout

$$\begin{aligned}
 L(s) &= \begin{cases} L_1(r', s) & (s = t : s', \text{pos}(t) = (r', c'), \text{isflt}(t)) \\ \{c'\} : \langle c' \rangle : L_1(r', s) & (s = t : s', \text{pos}(t) = (r', c'), \text{isflt}(t)) \\ \{0\} : \epsilon & (s = \epsilon) \end{cases} \\
 L_1(r, s) &= \begin{cases} \langle c' \rangle : L_2(r', c', t, s') & (s = t : s', \text{pos}(t) = (r', c'), r \neq r') \\ L_2(r', c', t, s') & (s = t : s', \text{pos}(t) = (r', c'), r = r') \\ \epsilon & (s = \epsilon) \end{cases} \\
 L_2(r_1, c_1, t_1, s) &= \begin{cases} t_1 : t_2 : L_1(r_2, s') & (\text{islt}(t_1), s = t_2 : s', \text{pos}(t_2) = (r_2, c_2), t_2 = "\{") \\ t_1 : \{c_2\} : \langle c_2 \rangle : t_2 : L_1(r_2, s') & (\text{islt}(t_1), s = t_2 : s', \text{pos}(t_2) = (r_2, c_2), t_2 \neq "\{") \\ t_1 : \{0\} : \epsilon & (\text{islt}(t_1), s = \epsilon) \\ t_1 : L_1(r_1, s) & (\text{islt}(t_1)) \end{cases} \\
 \text{isflt}(t) &= \begin{cases} \top & (t = \text{"module"}) \\ \perp & (\text{otherwise}) \end{cases} \\
 \text{islt}(t) &= \begin{cases} \top & (t = \text{"let"}) \\ \top & (t = \text{"where"}) \\ \top & (t = \text{"do"}) \\ \top & (t = \text{"of"}) \\ \perp & (\text{otherwise}) \end{cases}
 \end{aligned}$$

## 14.3.3 PEG with Layout Tokens

```

module ::= "module" modid exports? "where" body
        | body
body ::= expbo bodyinl expbc
        | impbo bodyinl impbc
bodyinl ::= impdecls semi topdecls
        | impdecls
        | topdecls

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 ::= gendecl
        | (funlhs | pat) rhs
cdecls ::= expbo cdeclsinl expbc
        | impbo cdeclsinl impbc
cdeclsinl ::= (cdecl semi)* cdecl |
cdecl ::= gendecl
        | (funlhs | var) rhs
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 ::= qtycon
           | "(" 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 ::= fexp? aexp
aexp ::= literal
       | "(" exp ")"
       | "(" (exp ",")+ exp ")"
       | "[" (exp ",")* exp "]"
       | "[" exp ("," exp)? ".." exp? "]"
       | "[" exp "|" (qual ",")* qual "]"
       | "(" infixexp qop ")"
       | "(" !("-" infixexp) qop infixexp ")"
       | qcon expbo ((fbind ",")* fbind |) expbc
       | !(qcon "{") aexp expbo ((fbind ",")* fbind |) expbc
       | qvar
       | 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

gcon  ::= "(" ")"
      | "[" "]"
      | "(" " , " " "
      | qcon
var    ::= varid | "(" varsym ")"
qvar   ::= qvarid | "(" qvarsym ")"
con    ::= conid | "(" consym ")"
qcon   ::= qconid | "(" gconsym ")"
varop  ::= varsym | "^" varid "^"
qvarop ::= qvarsym | "^" qvarid "^"
conop  ::= consym | "^" conid "^"
qvarop ::= gconsym | "^" qconid "^"
op     ::= varop | conop
qop    ::= qvarop | qconop
gconsym ::= ":" | qconsym

expbo  ::= [l]      "{" [0 : l]
expbc  ::= [0 : l] "}" [l]
impbo  ::= [m : l] {n} [n : m : l | n < m]
      | [m : l] {n} [(n + 1) : m : l | n ≥ m]
      | [ε] {n} [n : ε | n > 0]
impbc  ::= [m : l] ε [l | m > 0]
semi   ::= ";"
      | [m : l] ⟨n⟩ [m : l | m = n]

```

$$\text{skip}(l, t) = \begin{cases} \text{true} & (l = \langle m \rangle : l' \wedge t = \langle n \rangle \wedge m < n) \\ \text{false} & (\text{otherwise}) \end{cases}$$





## 第 15 章

# Analysis and Optimizations



## 第 16 章

# Meta-Programming and Multi-Stage Programming



## 第 17 章

# Generic Programming



## 第 18 章

# Advanced Calculus





## 第 19 章

### Some Notes of Quell Ideas

## 19.1 WIP: Implementation Note of PEG Parser

Normalizing

$$\begin{aligned}
 e_{\text{RHS}} &::= e_1 / \cdots / e_n / \epsilon & (n \in \mathbb{N}) \\
 &| e_1 / \cdots / e_n & (n \in \mathbb{N}_{\geq 1}) \\
 e &::= !(u_1 \cdots u_n) & (n \in \mathbb{N}_{\geq 1}) \\
 &| \&(u_1 \cdots u_n) & (n \in \mathbb{N}_{\geq 1}) \\
 &| u_1 \cdots u_n & (n \in \mathbb{N}_{\geq 1}) \\
 u &::= \sigma \\
 &| A
 \end{aligned}$$

$$\begin{aligned}
 \text{norm}(N, []) &= (N, \emptyset) \\
 \text{norm}(N, [A \leftarrow e] + X) &= (N_2, \{A \leftarrow \text{alt}(a)\} \cup X_1 \cup X_2) \\
 &(\text{norm}(N, e) = (a, N_1, X_1), \text{norm}(N_1, X) = (N_2, X_2))
 \end{aligned}$$

$$\begin{aligned}
 \text{norm}(N, \epsilon) &= ([\epsilon], N, \emptyset) \\
 \text{norm}(N, \sigma) &= ([\sigma], N, \emptyset) \\
 \text{norm}(N, A) &= ([A], N, \emptyset) \\
 \text{norm}(N, e_1 e_2) &= (\text{seq}(a_1, a_2), N_2, X_1 \cup X_2) & (\text{norm}(N, e_1) = (a_1, N_1, X_1), \text{norm}(N_1, e_2) = (a_2, N_2, X_2)) \\
 \text{norm}(N, e_1 / e_2) &= (a_1 + a_2, N_2, X_1 \cup X_2) & (\text{norm}(N, e_1) = (a_1, N_1, X_1), \text{norm}(N_1, e_2) = (a_2, N_2, X_2)) \\
 \text{norm}(N, e^*) &= ([M], N' \uplus \{M\}, X \cup \{M \leftarrow AM / \epsilon\}) & (\text{norm}(N \uplus \{A\}, [A \leftarrow e]) = (N', X)) \\
 \text{norm}(N, \& e) &= ([M], N' \uplus \{M\}, X \cup \{M \leftarrow \&A\}) & (\text{norm}(N + \{A\}, [A \leftarrow e]) = (N', X)) \\
 \text{norm}(N, !e) &= ([M], N' \uplus \{M\}, X \cup \{M \leftarrow !A\}) & (\text{norm}(N + \{A\}, [A \leftarrow e]) = (N', X))
 \end{aligned}$$

$$\begin{aligned}
 \text{seq}(a_1, a_2) &= [e_1 e_2 \mid e_1 \leftarrow a_1, e_2 \leftarrow a_2] \\
 \text{alt}([e_1, \dots, e_n]) &= e_1 / \cdots / e_m & (\forall i < m. e_i \neq \epsilon, e_m = \epsilon) \\
 \text{alt}([e_1, \dots, e_n]) &= e_1 / \cdots / e_n & (\forall i. e_i \neq \epsilon)
 \end{aligned}$$

$$\begin{aligned}
 \text{norm}((\Sigma, N, R, e_0)) &= (\Sigma, N', R', S) \\
 (R = \{A_1 \leftarrow e_1, \dots, A_n \leftarrow e_n\}, \text{norm}(N \uplus \{S\}, [S \leftarrow e_0, A_1 \leftarrow e_1, \dots, A_n \leftarrow e_n])) &= (N', R')
 \end{aligned}$$

Machine

State:

- a rule
- current position in rule

Transition:

- $\sigma$
- 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、メモリし、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

$$\begin{array}{lcl}
 E & ::= & CA \\
 & | & \epsilon \\
 A & ::= & aB \\
 & | & a \\
 B & ::= & bA \\
 & | & b \\
 C & ::= & !abab \\
 & | & \& ab
 \end{array}$$



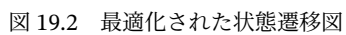


図 19.2 最適化された状態遷移図

## 19.2 Quell Syntax and Identations

### 19.2.1 Syntax

## 19.3 Quell Modules

### 19.3.1 Syntax

$$\begin{array}{ll}
 e & ::= \dots \\
 & \quad | \text{ letrec}\{B\} \text{ in } e \\
 & \quad | P \\
 \tau & ::= \dots \\
 & \quad | P \\
 P & ::= M \\
 M & ::= x \\
 & \quad | \{B\} \\
 & \quad | M.x \\
 & \quad | \text{ fun } x : S. M \\
 & \quad | x \ x \\
 & \quad | x : S \\
 B & ::= x = e \\
 & \quad | \text{ type } t = T \\
 & \quad | \text{ module } x = M \\
 & \quad | \text{ use } B \\
 & \quad | \epsilon \\
 & \quad | B; B \\
 T & ::= \lambda x. T \\
 & \quad | \tau \\
 S & ::= P \\
 & \quad | \{D\} \\
 & \quad | (x : S) \rightarrow S \\
 & \quad |
 \end{array}$$





## 参考文献

- [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.
- [GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. Cambridge University Press, apr 1989.
- [Knu65] Donald E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6):607–639, dec 1965.
- [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.