プログラミング言語周りノート

-

2022 年 1 月 1 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) λ-Calculus 8

2.2 Simply Typed λ-Calculus 9

2.3 WIP: System-T 12

2.4 WIP: PCF 13

2.5 System-F 14

2.6 System-F ω 19

2.7 λ μ-Calculus 26

2.8 WIP: Lambda Bar Mu Mu Tilde Calculus 30

2.9 WIP: π-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

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

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

は,

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

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

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

は,

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

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

集合 (*set*) について，以下の表記を用いる．

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

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

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

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

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

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

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

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

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

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

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

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

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

第 2 章

Basic Calculus

2.1 WIP: (Untyped) λ-Calculus

2.2 Simply Typed λ-Calculus

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

2.2.1 Syntax

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

Convention:

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

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

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

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

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

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

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

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

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

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

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

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

□

定理 2 (α-Equality Does Not Touch Free Variables). $e_1 \equiv_\alpha e_2$ ならば，$fv(e_1) = fv(e_2)$. □

2.2.2 Typing Semantics

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

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

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

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

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

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

2.2.3 Evaluation Semantics (Call-By-Value)

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

Small Step:

$\boxed{e \Rightarrow e'}$

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

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

Big Step:

$\boxed{e \Downarrow v}$

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

定理 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 $v = \mathrm{nf}(\Rightarrow, e)$ が存在し，

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

□

2.2.4 Equational Reasoning

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

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

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

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

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

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

定理 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

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

Convention:

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

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

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

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

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

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

$$\frac{[\overline{x'} \leftarrow \overline{e'}](x) = e}{x[\overline{x'} \leftarrow \overline{e'}] = e} \qquad \frac{x \notin \mathrm{dom}([\overline{x'} \leftarrow \overline{e'}])}{x[\overline{x'} \leftarrow \overline{e'}] = x}$$
$$\frac{e_1[\overline{x'} \leftarrow \overline{e'}] = e_1'' \quad e_2[\overline{x'} \leftarrow \overline{e'}] = e_2''}{(e_1\ e_2)[\overline{x'} \leftarrow \overline{e'}] = e_1''\ e_2''} \qquad \frac{e([\overline{x'} \leftarrow \overline{e'}] \restriction_{\mathrm{dom}([\overline{x'} \leftarrow \overline{e'}]) \setminus \{x\}}) = e''}{(\lambda x : \tau . e)[\overline{x'} \leftarrow \overline{e'}] = \lambda x : \tau . e''}$$
$$\frac{e[\overline{x'} \leftarrow \overline{e'}] = e''}{(e\ \tau)[\overline{x'} \leftarrow \overline{e'}] = e''\ \tau} \qquad \frac{e[\overline{x'} \leftarrow \overline{e'}] = e''}{(\Lambda t . e)[\overline{x'} \leftarrow \overline{e'}] = \Lambda t . e''}$$

Type Free Variable:

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

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

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

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

Type Substitution:

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

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

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

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

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

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

α-Equality:

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

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

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

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

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

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

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

□

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

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

□

定理 10 (α-Equality Does Not Touch Free Variables).

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

□

2.5.2 Typing Semantics

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

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

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

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

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

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

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

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

2.5.3 Evaluation Semantics (Call-By-Value)

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

Small Step:

$\boxed{e \Rightarrow e'}$

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

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

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

Big Step:

$\boxed{e \Downarrow v}$

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

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

定理 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 $v = \mathrm{nf}(\Rightarrow, e)$ が存在し，

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

□

2.5.4 Equational Reasoning

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

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

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

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

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

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

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

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

定理 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 τ_1 and τ_2:

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

Admissible typing rule:

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

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

Admissible equality:

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

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

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

Existential Type

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

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

Admissible typing rule:

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

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

Admissible equality:

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

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

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

2.6 System-F ω

Alias: F ω, $\lambda\omega$ [RRD14]

2.6.1 Syntax

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

Convention:

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

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

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

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

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

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

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

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

Type Free Variable:

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

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

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

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

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

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

Type Substitution:

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

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

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

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

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

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

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

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

α-Equality:

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

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

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

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

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

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

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

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

□

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

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

□

定理 18 (α-Equality Does Not Touch Free Variables).

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

□

2.6.2 Typing Semantics

Kinding:

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

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

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

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

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

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

Type equivalence:

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

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

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

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

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

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

定理 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:

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

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

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

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

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

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

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

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

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

2.6.3 Evaluation Semantics (Call-By-Value)

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

Small Step:

$\boxed{e \Rightarrow e'}$

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

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

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

Big Step:

$\boxed{e \Downarrow v}$

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

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

定理 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 $v = \mathrm{nf}(\Rightarrow, e)$ が存在し，

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

□

2.6.4 Equational Reasoning

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

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

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

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

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

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

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

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

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

定理 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 τ_1 and τ_2:

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

Admissible kinding:

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

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

Admissible type equality:

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

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

Admissible typing:

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

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

Admissible equality:

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

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

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

Existential Type

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

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

Admissible kinding:

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

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

Admissible type equality:

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

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

Admissible typing rule:

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

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

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

Admissible equality:

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

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

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

2.7 λ μ-Calculus

Alias: λ μ [Sel01][Roc05]

2.7.1 Syntax

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

Environment Reference:

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

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

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

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

2.7.2 Typing Semantics

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

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

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

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

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

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

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

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

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

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

2.7.3 Equivalence

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

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

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

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

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

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

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

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

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

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

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

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

2.7.4 Elaboration (Call-By-Value)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Abbreviation:

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

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

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，

$$V(\Gamma) \stackrel{\mathrm{def}}{=} \begin{cases} V(\Gamma'), x_{x'} : V_{\tau'} & (\Gamma = \Gamma', x' : \tau') \\ \cdot & (\Gamma = \cdot) \end{cases}$$

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

□

2.7.5 Elaboration (Call-By-Name)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Abbreviation:

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

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，

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

□

2.8 WIP: Lambda Bar Mu Mu Tilde Calculus

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

2.9 WIP: π-Calculus

第 3 章

Modules and Phase Distinction

3.1 Light-Weight F-ing modules

[RRD14]

3.1.1 Internal Language

Having same power as System F ω

Syntax:

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

Abbreviation:

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

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

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

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

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

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

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

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

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

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

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

Kinding:

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

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

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

Type equivalence:

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

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

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

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

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

Typing:

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

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

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

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

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

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

Reduction:

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

$\boxed{e \Rightarrow e'}$

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

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

Equivalence:

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

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

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

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

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

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

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

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

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

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

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

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

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

3.1.2 Syntax

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

3.1.3 Signature

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

Atomic Signature:

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

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

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

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

Admissible kinding:

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

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

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

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

Admissible type equivalence:

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

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

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

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

Admissible typing:

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

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

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

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

Admissible equivalence:

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

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

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

$$\frac{\Gamma \vdash \Sigma_1 \equiv \Sigma_2 : \Omega}{\Gamma \vdash [\Sigma_1] \equiv [\Sigma_2] : [= \Sigma_1]}\ \text{Eq-Cong-A-Sig}$$

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:

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

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

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

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

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

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

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

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

Module:

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

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

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

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

Bindings:

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

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

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

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

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

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

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

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

Path:

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

Use M-Dot.

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

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

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

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

3.2 F-ing modules

[RRD14]

3.2.1 Internal Language

See 第 3.1.1 小節.

3.2.2 Syntax

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

3.2.3 Signature

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

Atomic Signature:

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

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

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

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

Admissible kinding:

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

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

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

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

Admissible type equivalence:

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

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

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

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

Admissible typing:

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

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

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

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

Admissible equivalence:

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

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

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

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

3.2.4 (Generative) Elaboration

Signature:

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

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

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

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

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

Declarations:

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

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

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

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

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

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

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

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

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

Matching:

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

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

Subtyping:

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

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

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

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

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

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

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

Module:

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

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

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

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

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

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

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

Bindings:

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

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

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

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

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

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

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

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

Path:

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

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

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

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

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

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

3.2.5 Modules as First-Class Values

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

Rootedness:

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

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

Rooted ordering:

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

Signature normalization:

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

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

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

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

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

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

Type:

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

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

Expression:

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

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

Module:

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

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

3.2.6 Elaboration with Applicative Functor

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

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

Abbreviation:

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

Effect combining:

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

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

Subeffects:

$\boxed{\varphi_1 \leq \varphi_2}$

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

Signature:

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

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

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

Subtyping:

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

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

Module:

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

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

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

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

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

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

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

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

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

定理 30 (Typing for module elaboration).

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

□

Bindings:

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

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

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

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

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

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

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

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

Path:

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

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

Expression:

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

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

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

$$
\begin{array}{rcll}
e & ::= & \epsilon & \text{(epsilon)} \\
& | & \sigma & \text{(terminal)} \\
& | & A & \text{(non-terminal)} \\
& | & ee & \text{(sequence)} \\
& | & e \ / \ e & \text{(alternative)} \\
& | & e^* & \text{(repetition)} \\
& | & !\,e & \text{(not predicate)} \\
\sigma & \in & \Sigma & \\
A & \in & N &
\end{array}
$$

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．

Σ 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

$$\frac{}{\langle \epsilon, x \rangle \rightarrow \mathrm{s}(\epsilon)}$$

$$\frac{}{\langle \sigma, \sigma x \rangle \rightarrow \mathrm{s}(\sigma)} \qquad \frac{\sigma \neq \sigma'}{\langle \sigma, \sigma' x \rangle \rightarrow \mathrm{f}} \qquad \frac{}{\langle \sigma, \epsilon \rangle \rightarrow \mathrm{f}}$$

$$\frac{A \leftarrow e \in R \quad \langle e, x \rangle \rightarrow o}{\langle A, x \rangle \rightarrow o}$$

$$\frac{\langle e_1, x_1 x_2 y \rangle \rightarrow \mathrm{s}(x_1) \quad \langle e_2, x_2 y \rangle \rightarrow \mathrm{s}(x_2)}{\langle e_1 e_2, x_1 x_2 y \rangle \rightarrow \mathrm{s}(x_1 x_2)} \qquad \frac{\langle e_1, x \rangle \rightarrow \mathrm{f}}{\langle e_1 e_2, x \rangle \rightarrow \mathrm{f}} \qquad \frac{\langle e_1, x_1 y \rangle \rightarrow \mathrm{s}(x_1) \quad \langle e_2, y \rangle \rightarrow \mathrm{f}}{\langle e_1 e_2, x_1 y \rangle \rightarrow \mathrm{f}}$$

$$\frac{\langle e_1, xy \rangle \rightarrow \mathrm{s}(x)}{\langle e_1 \ / \ e_2, xy \rangle \rightarrow \mathrm{s}(x)} \qquad \frac{\langle e_1, x \rangle \rightarrow \mathrm{f} \quad \langle e_2, x \rangle \rightarrow o}{\langle e_1 \ / \ e_2, x \rangle \rightarrow o}$$

$$\frac{\langle e, x_1 x_2 y \rangle \rightarrow \mathrm{s}(x_1) \quad \langle e^*, x_2 y \rangle \rightarrow \mathrm{s}(x_2)}{\langle e^*, x_1 x_2 y \rangle \rightarrow \mathrm{s}(x_1 x_2)} \qquad \frac{\langle e, x \rangle \rightarrow \mathrm{f}}{\langle e^*, x \rangle \rightarrow \mathrm{s}(\epsilon)}$$

$$\frac{\langle e, x \rangle \rightarrow \mathrm{f}}{\langle !e, x \rangle \rightarrow \mathrm{s}(\epsilon)} \qquad \frac{\langle e, xy \rangle \rightarrow \mathrm{s}(x)}{\langle !e, xy \rangle \rightarrow \mathrm{f}}$$

$$
\begin{aligned}
[\![(\Sigma, N, R, e_0)]\!] &= [\![e_0]\!] \\
[\![e]\!] &= \{x \in \Sigma^* \mid \langle e, x \rangle \rightarrow \mathrm{s}(x)\}
\end{aligned}
$$

14.2.3 Equivalence

Abbreviations

$$\begin{aligned} \&\, e &= !(!\, e) && \text{(and predicate)} \\ e^{+} &= ee^{*} && \text{(positive repetition)} \\ e^{?} &= e/\epsilon && \text{(optional)} \end{aligned}$$

Associativity

$$\frac{}{[\![e_1/(e_2/e_3)]\!] = [\![(e_1/e_2)/e_3]\!]}$$

$$\frac{}{[\![e_1(e_2e_3)]\!] = [\![(e_1e_2)e_3]\!]}$$

Epsilon

$$\frac{}{[\![\epsilon/e]\!] = [\![\epsilon]\!]}$$

$$\frac{}{[\![e\epsilon]\!] = [\![e]\!]} \qquad \frac{}{[\![\epsilon e]\!] = [\![e]\!]}$$

Repetition

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

$$\frac{}{[\![e^{*}]\!] = [\![M]\!]}$$

14.2.4 Producing Analysis

$s ::= 0 \mid 1,\ \ o ::= \mathrm{s} \mid \mathrm{f}$

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

- $e \rightharpoonup \mathrm{f}$ ならば $!e \rightharpoonup 0$

定理 32.

- $\langle e, x \rangle \to \mathrm{s}(\epsilon)$ ならば，$e \rightharpoonup 0$
- $\langle e, xy \rangle \to \mathrm{s}(x)$，$x \neq \epsilon$ ならば，$e \rightharpoonup 1$
- $\langle e, x \rangle \to \mathrm{f}$ ならば，$e \rightharpoonup \mathrm{f}$

$\square$

系 33. $e \not\rightharpoonup o$ ならば，$\langle e, xy \rangle \not\to \mathrm{s}(x)$ かつ $\langle e, xy \rangle \not\to \mathrm{f}$ $\square$

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    ::= "\r\n" | "\r" | "\n" | "\f"
comment    ::= dashes (!symbol any*)? newline
dashes     ::= "-" ("-")+
opencom    ::= "{-"
closecom   ::= "-}"
ncomment   ::= opencom ANYs (ncomment ANYs)* closecom
ANYs       ::= !(ANY* (opencom | closecom) ANY*) ANY*
ANY        ::= graphic | whitechar
any        ::= graphic | " " | "\t"
graphic    ::= small | large | symbol | digit | special | "\"" | "'"
small      ::= "a" | "b" | ··· | "z" | (Unicode lowercase letter) | "_"
large      ::= "A" | "B" | ··· | "Z" | (Unicode uppercase letter) | (Unicode titlecase letter)
symbol     ::= "!" | "#" | "$" | "%" | "&" | "*" | "+" | "." | "/" | "<" | "=" | ">"
            |  "?" | "@" | "\\" | "^" | "|" | "-" | "~" | ":"
            |  !(symbol | "_" | "\"" | "'") uniSymbol
uniSymbol  ::= (Unicode symbol) | (Unicode punctuation)
digit      ::= "0" | "1" | ··· | "9" | (Unicode decimal digit)
octit      ::= "0" | "1" | ··· | "7"
hexit      ::= digit | "A" | ··· | "F" | "a" | ··· | "f"
varid      ::= !(reservedid !other) small other*
conid      ::= large other*
other      ::= small | large | digit | "'"
reservedid ::= "case" | "class" | "data" | "default" | "deriving" | "do" | "else"
            |  "foreign" | "if" | "import" | "in" | "infix" | "infixl" | "infixr"
            |  "instance" | "let" | "module" | "newtype" | "of" | "then" | "type"
            |  "where" | "_"
varsym     ::= !((reservedop | dashes) !symbol | ":") symbol+
consym     ::= !(reservedop !symbol) ":" symbol+
reservedop ::= ".." | ":" | "::" | "=" | "\\" | "<-" | "->" | "@" | "~" | "=>"
```

$$
\begin{array}{rcl}
\mathit{modid} & ::= & (\mathit{conid}\ \texttt{"."})^{*}\ \mathit{conid} \\
\mathit{qvarid} & ::= & (\mathit{modid}\ \texttt{"."})^{?}\ \mathit{varid} \\
\mathit{qconid} & ::= & (\mathit{modid}\ \texttt{"."})^{?}\ \mathit{conid} \\
\mathit{qvarsym} & ::= & (\mathit{modid}\ \texttt{"."})^{?}\ \mathit{varsym} \\
\mathit{qconsym} & ::= & (\mathit{modid}\ \texttt{"."})^{?}\ \mathit{consym}
\end{array}
$$

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

14.3.2 Preprocess for Layout

$$
L(s) = \begin{cases}
L_1(r', s) & (s = t : s', \mathrm{pos}(t) = (r', c'), \mathrm{isflt}(t)) \\
\{c'\} : \langle c' \rangle : L_1(r', s) & (s = t : s', \mathrm{pos}(t) = (r', c'), \mathrm{isfl}\not{t}(t)) \\
\{0\} : \epsilon & (s = \epsilon)
\end{cases}
$$

$$
L_1(r, s) = \begin{cases}
\langle c' \rangle : L_2(r', c', t, s') & (s = t : s', \mathrm{pos}(t) = (r', c'), r \neq r') \\
L_2(r', c', t, s') & (s = t : s', \mathrm{pos}(t) = (r', c'), r = r') \\
\epsilon & (s = \epsilon)
\end{cases}
$$

$$
L_2(r_1, c_1, t_1, s) = \begin{cases}
t_1 : t_2 : L_1(r_2, s') & (\mathrm{islt}(t_1), s = t_2 : s', \mathrm{pos}(t_2) = (r_2, c_2), t_2 = \texttt{"\{"}) \\
t_1 : \{c_2\} : \langle c_2 \rangle : t_2 : L_1(r_2, s') & (\mathrm{islt}(t_1), s = t_2 : s', \mathrm{pos}(t_2) = (r_2, c_2), t_2 \neq \texttt{"\{"}) \\
t_1 : \{0\} : \epsilon & (\mathrm{islt}(t_1), s = \epsilon) \\
t_1 : L_1(r_1, s) & (\mathrm{isl}\not{t}(t_1))
\end{cases}
$$

$$
\mathrm{islft}(t) = \begin{cases}
\top & (t = \texttt{"module"}) \\
\bot & (\text{otherwise})
\end{cases}
$$

$$
\mathrm{islt}(t) = \begin{cases}
\top & (t = \texttt{"let"}) \\
\top & (t = \texttt{"where"}) \\
\top & (t = \texttt{"do"}) \\
\top & (t = \texttt{"of"}) \\
\bot & (\text{otherwise})
\end{cases}
$$

14.3.3 PEG with Layout Tokens

$$
\begin{array}{rcl}
\mathit{module} & ::= & \texttt{"module"}\ \mathit{modid}\ \mathit{exports}^{?}\ \texttt{"where"}\ \mathit{body} \\
 & | & \mathit{body} \\
\mathit{body} & ::= & \mathit{expbo}\ \mathit{bodyinl}\ \mathit{expbc} \\
 & | & \mathit{impbo}\ \mathit{bodyinl}\ \mathit{impbc} \\
\mathit{bodyinl} & ::= & \mathit{impdecls}\ \mathit{semi}\ \mathit{topdecls} \\
 & | & \mathit{impdecls} \\
 & | & \mathit{topdecls}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{exports} & ::= & \texttt{"("}\ (\mathit{export}\ \texttt{","})^{*}\ \mathit{export}^{?}\ \texttt{")"} \\
\mathit{export} & ::= & \mathit{qvar} \\
 & | & \mathit{qtycon}\ (\texttt{"("}\ (\texttt{".."}\ |\ (\mathit{cname}\ \texttt{","})^{*}\ \mathit{cname}\ |)\ \texttt{")"})^{?} \\
 & | & \texttt{"module"}\ \mathit{modid} \\
\mathit{impdecl} & ::= & \texttt{"import"}\ \texttt{"qualified"}^{?}\ \mathit{modid}\ (\texttt{"as"}\ \mathit{modid})^{?}\ \mathit{impspec}^{?} \\
 & | & \\
\mathit{impspec} & ::= & \texttt{"("}\ (\mathit{import}\ \texttt{","})^{*}\ \mathit{import}^{?}\ \texttt{")"} \\
 & | & \texttt{"hiding"}\ \texttt{"("}\ (\mathit{import}\ \texttt{","})^{*}\ \mathit{import}^{?}\ \texttt{")"} \\
\mathit{import} & ::= & \mathit{var} \\
 & | & \mathit{tycon}\ (\texttt{"("}\ (\texttt{".."}\ |\ (\mathit{cname}\ \texttt{","})^{*}\ \mathit{cname}\ |)\ \texttt{")"})^{?} \\
\mathit{cname} & ::= & \mathit{var}\ |\ \mathit{con}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{topdecls} & ::= & (\mathit{topdecl}\ \mathit{semi})^{*}\ \mathit{topdecl}\ | \\
\mathit{topdecl} & ::= & \texttt{"type"}\ \mathit{simpletype}\ \texttt{"="}\ \mathit{type} \\
 & | & \texttt{"data"}\ (\mathit{context}\ \texttt{"=>"})^{?}\ \mathit{simpletype}\ (\texttt{"="}\ \mathit{constrs})^{?}\ \mathit{deriving}^{?} \\
 & | & \texttt{"newtype"}\ (\mathit{context}\ \texttt{"=>"})^{?}\ \mathit{simpletype}\ \texttt{"="}\ \mathit{newconstr}\ \mathit{deriving}^{?} \\
 & | & \texttt{"class"}\ (\mathit{scontext}\ \texttt{"=>"})^{?}\ \mathit{tycon}\ \mathit{tyvar}\ (\texttt{"where"}\ \mathit{cdecls})^{?} \\
 & | & \texttt{"instance"}\ (\mathit{scontext}\ \texttt{"=>"})^{?}\ \mathit{qtycon}\ \mathit{inst}\ (\texttt{"where"}\ \mathit{idecls})^{?} \\
 & | & \texttt{"default"}\ \texttt{"("}\ ((\mathit{type}\ \texttt{","})^{*}\ \mathit{type}\ |)\ \texttt{")"} \\
 & | & \texttt{"foreign"}\ \mathit{fdecl} \\
 & | & \mathit{decl}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{decls} & ::= & \mathit{expbo}\ \mathit{declsinl}\ \mathit{expbc} \\
 & | & \mathit{impbo}\ \mathit{declsinl}\ \mathit{impbc} \\
\mathit{declsinl} & ::= & (\mathit{decl}\ \mathit{semi})^{*}\ \mathit{decl}\ | \\
\mathit{decl} & ::= & \mathit{gendecl} \\
 & | & (\mathit{funlhs}\ |\ \mathit{pat})\ \mathit{rhs} \\
\mathit{cdecls} & ::= & \mathit{expbo}\ \mathit{cdeclsinl}\ \mathit{expbc} \\
 & | & \mathit{impbo}\ \mathit{cdeclsinl}\ \mathit{impbc} \\
\mathit{cdeclsinl} & ::= & (\mathit{cdecl}\ \mathit{semi})^{*}\ \mathit{cdecl}\ | \\
\mathit{cdecl} & ::= & \mathit{gendecl} \\
 & | & (\mathit{funlhs}\ |\ \mathit{var})\ \mathit{rhs} \\
\mathit{idecls} & ::= & \mathit{expbo}\ \mathit{ideclsinl}\ \mathit{expbc} \\
 & | & \mathit{impbo}\ \mathit{ideclsinl}\ \mathit{impbc} \\
\mathit{ideclsinl} & ::= & (\mathit{idecl}\ \mathit{semi})^{*}\ \mathit{idecl}\ | \\
\mathit{idecl} & ::= & (\mathit{funlhs}\ |\ \mathit{var})\ \mathit{rhs} \\
 & | & \\
\mathit{gendecl} & ::= & \mathit{vars}\ \texttt{"::"}\ (\mathit{context}\ \texttt{"=>"})^{?}\ \mathit{type} \\
 & | & \mathit{fixity}\ \mathit{integer}^{?}\ \mathit{ops} \\
 & | & \\
\mathit{ops} & ::= & (\mathit{op}\ \texttt{","})^{*}\ \mathit{op} \\
\mathit{vars} & ::= & (\mathit{var}\ \texttt{","})^{*}\ \mathit{var} \\
\mathit{fixity} & ::= & \texttt{"infixl"}\ |\ \texttt{"infixr"}\ |\ \texttt{"infix"}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{type} & ::= & \mathit{btype}\ (\texttt{"->"}\ \mathit{type})^{?} \\
\mathit{btype} & ::= & \mathit{btype}^{?}\ \mathit{atype} \\
\mathit{atype} & ::= & \mathit{gtycon} \\
 & | & \mathit{tyvar} \\
 & | & \texttt{"("}\ (\mathit{type}\ \texttt{","})^{+}\ \mathit{type}\ \texttt{")"} \\
 & | & \texttt{"["}\ \mathit{type}\ \texttt{"]"} \\
 & | & \texttt{"("}\ \mathit{type}\ \texttt{")"} \\
\mathit{gtycon} & ::= & \mathit{qtycon} \\
 & | & \texttt{"("}\ \texttt{")"} \\
 & | & \texttt{"["}\ \texttt{"]"} \\
 & | & \texttt{"("}\ \texttt{"->"}\ \texttt{")"} \\
 & | & \texttt{"("}\ \texttt{","}^{+}\ \texttt{")"}
\end{array}
$$

```
     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
```

$$
\begin{array}{rcl}
\mathit{exp} & ::= & \mathit{infixexp}\ \texttt{"::"}\ (\mathit{context}\ \texttt{"=>"})^{?}\ \mathit{type} \\
 & | & \mathit{infixexp} \\
\mathit{infixexp} & ::= & \texttt{"-"}\ \mathit{infixexp} \\
 & | & \mathit{lexp}\ \mathit{qop}\ \mathit{infixexp} \\
 & | & \mathit{lexp} \\
\mathit{lexp} & ::= & \texttt{"\textbackslash\textbackslash"}\ \mathit{apat}^{+}\ \texttt{"->"}\ \mathit{exp} \\
 & | & \texttt{"let"}\ \mathit{decls}\ \texttt{"in"}\ \mathit{exp} \\
 & | & \texttt{"if"}\ \mathit{exp}\ \mathit{semi}^{?}\ \texttt{"then"}\ \mathit{exp}\ \mathit{semi}^{?}\ \texttt{"else"}\ \mathit{exp} \\
 & | & \texttt{"case"}\ \mathit{exp}\ \texttt{"of"}\ \mathit{casealts} \\
 & | & \texttt{"do"}\ \mathit{dostmts} \\
 & | & \mathit{fexp} \\
\mathit{fexp} & ::= & \mathit{fexp}^{?}\ \mathit{aexp} \\
\mathit{aexp} & ::= & \mathit{literal} \\
 & | & \texttt{"("}\ \mathit{exp}\ \texttt{")"} \\
 & | & \texttt{"("}\ (\mathit{exp}\ \texttt{","})^{+}\ \mathit{exp}\ \texttt{")"} \\
 & | & \texttt{"["}\ (\mathit{exp}\ \texttt{","})^{*}\ \mathit{exp}\ \texttt{"]"} \\
 & | & \texttt{"["}\ \mathit{exp}\ (\texttt{","}\ \mathit{exp})^{?}\ \texttt{".."}\ \mathit{exp}^{?}\ \texttt{"]"} \\
 & | & \texttt{"["}\ \mathit{exp}\ \texttt{"|"}\ (\mathit{qual}\ \texttt{","})^{*}\ \mathit{qual}\ \texttt{"]"} \\
 & | & \texttt{"("}\ \mathit{infixexp}\ \mathit{qop}\ \texttt{")"} \\
 & | & \texttt{"("}\ !(\texttt{"-"}\ \mathit{infixexp})\ \mathit{qop}\ \mathit{infixexp}\ \texttt{")"} \\
 & | & \mathit{qcon}\ \mathit{expbo}\ ((\mathit{fbind}\ \texttt{","})^{*}\ \mathit{fbind}\ |)\ \mathit{expbc} \\
 & | & !(\mathit{qcon}\ \texttt{"\{"})\ \mathit{aexp}\ \mathit{expbo}\ ((\mathit{fbind}\ \texttt{","})^{*}\ \mathit{fbind}\ |)\ \mathit{expbc} \\
 & | & \mathit{qvar} \\
 & | & \mathit{gcon}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{qual} & ::= & \mathit{pat}\ \texttt{"<-"}\ \mathit{exp} \\
 & | & \texttt{"let"}\ \mathit{decls} \\
 & | & \mathit{exp} \\
\mathit{casealts} & ::= & \mathit{expbo}\ \mathit{alts}\ \mathit{expbc} \\
 & | & \mathit{impbo}\ \mathit{alts}\ \mathit{impbc} \\
\mathit{alts} & ::= & (\mathit{alt}\ \mathit{semi})^{*}\ \mathit{alt} \\
\mathit{alt} & ::= & \mathit{pat}\ \texttt{"->"}\ \mathit{exp}\ (\texttt{"where"}\ \mathit{decls})^{?} \\
 & | & \mathit{pat}\ \mathit{gdpat}\ (\texttt{"where"}\ \mathit{decls})^{?} \\
 & | & \\
\mathit{gdpat} & ::= & \mathit{guards}\ \texttt{"->"}\ \mathit{exp}\ \mathit{gdpat}^{?} \\
\mathit{dostmts} & ::= & \mathit{expbo}\ \mathit{stmts}\ \mathit{expbc} \\
 & | & \mathit{impbo}\ \mathit{stmts}\ \mathit{impbc} \\
\mathit{stmts} & ::= & \mathit{stmt}^{*}\ \mathit{exp}\ \mathit{semi}^{?} \\
\mathit{stmt} & ::= & \mathit{exp}\ \mathit{semi} \\
 & | & \mathit{pat}\ \texttt{"<-"}\ \mathit{exp}\ \mathit{semi} \\
 & | & \texttt{"let"}\ \mathit{decls}\ \mathit{semi} \\
 & | & \mathit{semi} \\
\mathit{fbind} & ::= & \mathit{qvar}\ \texttt{"="}\ \mathit{exp}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{pat} & ::= & \mathit{lpat}\ \mathit{qconop}\ \mathit{pat} \\
 & | & \mathit{lpat} \\
\mathit{lpat} & ::= & \texttt{"-"}\ (\mathit{integer}\ |\ \mathit{float}) \\
 & | & \mathit{gcon}\ \mathit{apat}^{+} \\
 & | & \mathit{apat} \\
\mathit{apat} & ::= & \mathit{var}\ (\texttt{"@"}\ \mathit{apat})^{?} \\
 & | & \mathit{literal} \\
 & | & \texttt{"\_"} \\
 & | & \texttt{"("}\ \mathit{pat}\ \texttt{")"} \\
 & | & \texttt{"("}\ (\mathit{pat}\ \texttt{","})^{+}\ \mathit{pat}\ \texttt{")"} \\
 & | & \texttt{"["}\ (\mathit{pat}\ \texttt{","})^{*}\ \mathit{pat}\ \texttt{"]"} \\
 & | & \texttt{"\textasciitilde"}\ \mathit{apat} \\
 & | & \mathit{qcon}\ \mathit{expbo}\ ((\mathit{fpat}\ \texttt{","})^{*}\ \mathit{fpat}\ |)\ \mathit{expbc} \\
 & | & \mathit{gcon} \\
\mathit{fpat} & ::= & \mathit{qvar}\ \texttt{"="}\ \mathit{pat}
\end{array}
$$

$$
\begin{array}{rcl}
gcon & ::= & \texttt{"("}\ \texttt{")"} \\
 & | & \texttt{"["}\ \texttt{"]"} \\
 & | & \texttt{"("}\ \texttt{","}^{+}\ \texttt{")"} \\
 & | & qcon \\
var & ::= & varid \mid \texttt{"("}\ varsym\ \texttt{")"} \\
qvar & ::= & qvarid \mid \texttt{"("}\ qvarsym\ \texttt{")"} \\
con & ::= & conid \mid \texttt{"("}\ consym\ \texttt{")"} \\
qcon & ::= & qconid \mid \texttt{"("}\ gconsym\ \texttt{")"} \\
varop & ::= & varsym \mid \texttt{"`"}\ varid\ \texttt{"`"} \\
qvarop & ::= & qvarsym \mid \texttt{"`"}\ qvarid\ \texttt{"`"} \\
conop & ::= & consym \mid \texttt{"`"}\ conid\ \texttt{"`"} \\
qvarop & ::= & gconsym \mid \texttt{"`"}\ qconid\ \texttt{"`"} \\
op & ::= & varop \mid conop \\
qop & ::= & qvarop \mid qconop \\
gconsym & ::= & \texttt{":"} \mid qconsym \\
tyvar & ::= & varid \\
tycon & ::= & conid \\
qtycon & ::= & qconid
\end{array}
$$

$$
\begin{array}{rclll}
expbo & ::= & [l] & \texttt{"\{"} & [0 : l] \\
expbc & ::= & [0 : l] & \texttt{"\}"} & [l] \\
impbo & ::= & [m : l] & \{n\} & [n : m : l \mid n < m] \\
 & | & [m : l] & \{n\} & [(n+1) : m : l \mid n \geq m] \\
 & | & [\epsilon] & \{n\} & [n : \epsilon \mid n > 0] \\
impbc & ::= & [m : l] & \epsilon & [l \mid m > 0] \\
semi & ::= & & \texttt{";"} & \\
 & | & [m : l] & \langle n \rangle & [m : l \mid m = n]
\end{array}
$$

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

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

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

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

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

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

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

Machine

State:

- a rule
- current position in rule

Transition:

- σ
- EOS
- otherwise

Output:

with backpoint　バックポイントを設置し，バックポイントに戻った時の次の遷移を指定する．fail した場合一番直近の backpoint まで入力状態とスタックを戻す．reduce 時取り除かれる．

enter　非終端記号を参照する．メモ化されている場合その値を使う．それ以外の場合，reduce 時戻ってくる状態を記録し，次の状態に遷移する．

goto　次の状態に遷移する．

shift　入力を 1 つ消費し，次の状態に遷移する．

reduce　規則に沿ってスタックから要素を取り出してまとめ，メモし，スタックに新たに入れた後，enter 時に記録された状態に遷移する．

Optimization

1. unify transitions.
2. look ahead backpoints.

Example

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

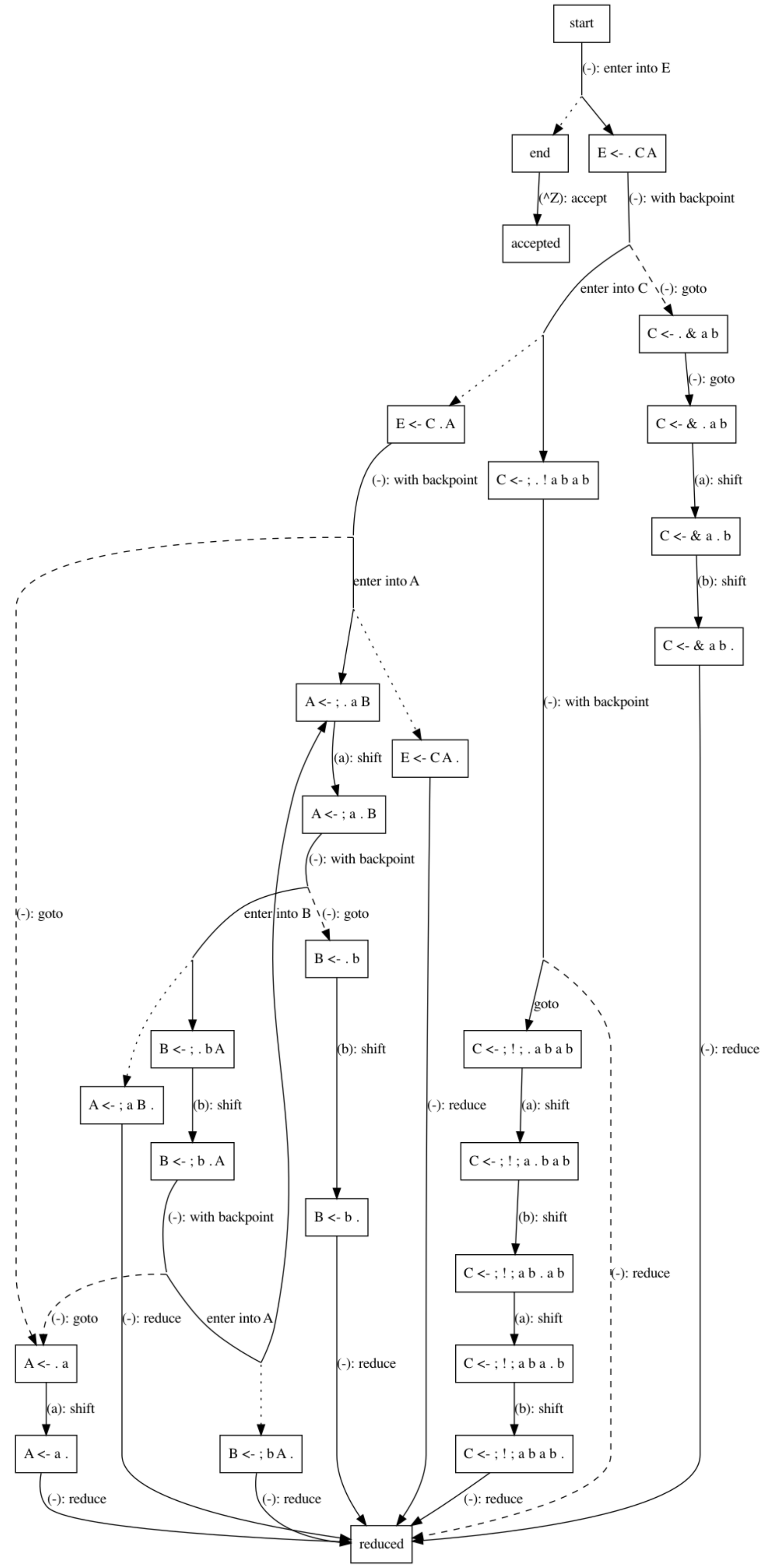
start
(-): enter into E
end
E <- . C A
(^Z): accept
(-): with backpoint
accepted
enter into C
(-): goto
C <- . & a b
(-): goto
E <- C . A
C <- & . a b
(-): with backpoint
C <- ; . ! a b a b
(a): shift
C <- & a . b
enter into A
(b): shift
C <- & a b .
A <- ; . a B
(-): with backpoint
E <- C A .
(a): shift
A <- ; a . B
(-): with backpoint
(-): goto
enter into B
(-): goto
B <- . b
goto
B <- ; . b A
(b): shift
C <- ; ! ; . a b a b
(-): reduce
A <- ; a B .
(b): shift
(-): reduce
(a): shift
B <- ; b . A
C <- ; ! ; a . b a b
(-): with backpoint
B <- b .
(b): shift
C <- ; ! ; a b . a b
(-): reduce
(-): goto
(-): reduce
enter into A
(a): shift
A <- . a
(-): reduce
C <- ; ! ; a b a . b
(a): shift
(b): shift
A <- a .
B <- ; b A .
C <- ; ! ; a b a b .
(-): reduce
(-): reduce
(-): reduce
reduced

図 19.1　状態遷移図

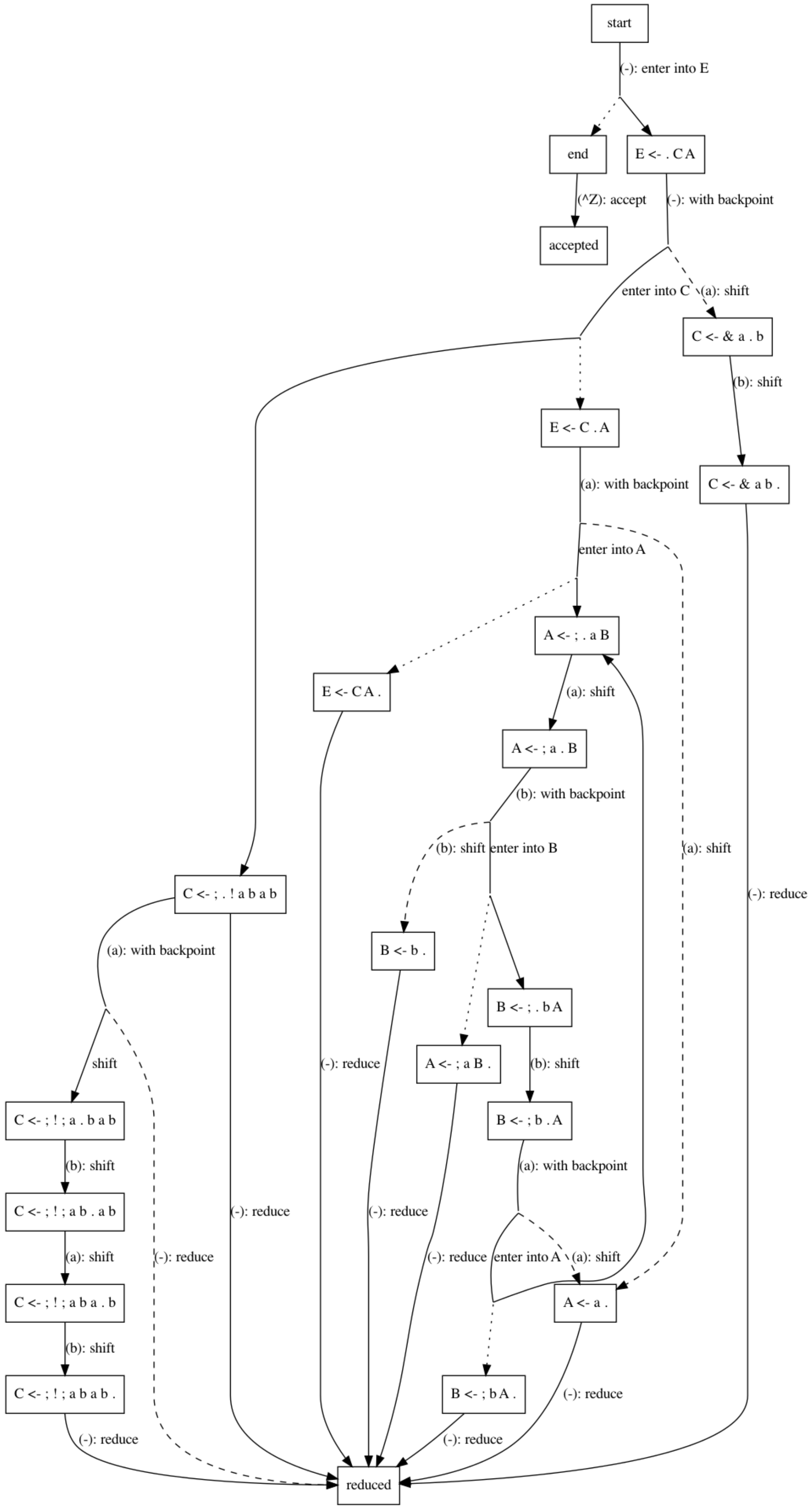

図 19.2　最適化された状態遷移図

19.2 Quell Syntax and Identations

19.2.1 Syntax

19.3 Quell Modules

19.3.1 Syntax

$$
\begin{array}{rcl}
e & ::= & \cdots \\
  & | & \mathrm{letrec}\{B\}\ \mathrm{in}\ e \\
  & | & P \\
\tau & ::= & \cdots \\
  & | & P \\
P & ::= & M \\
M & ::= & x \\
  & | & \{B\} \\
  & | & M.x \\
  & | & \mathrm{fun}\, x : S.\, M \\
  & | & x\ x \\
  & | & x : S \\
B & ::= & x = e \\
  & | & \mathrm{type}\, t = T \\
  & | & \mathrm{module}\, x = M \\
  & | & \mathrm{use}\, B \\
  & | & \epsilon \\
  & | & B; B \\
T & ::= & \lambda x.\, T \\
  & | & \tau \\
S & ::= & P \\
  & | & \{D\} \\
  & | & (x : S) \to S \\
  & | &
\end{array}
$$

参考文献

[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.