

# The Lambë programming language

D. Plaindoux

March 27, 2021

## 1 Kind level

$$\begin{array}{lcl} \kappa & = & \star \\ & | & \kappa \rightarrow \kappa \\ & | & K \end{array}$$

## 2 Type level

$$\begin{array}{lcl} \underline{c} & \in & \mathcal{C} \quad \text{Constructor names} \\ \alpha & \in & \mathcal{I} \quad \text{Variable names} \end{array}$$

$$\begin{array}{lcl} \tau & = & \alpha \quad \text{Variable or Constant} \\ & | & \tau \rightarrow \tau \quad \text{Function Type} \\ & | & \tau \multimap \tau \quad \text{Method Type} \\ & | & \tau + \tau \quad \text{Sum type} \\ & | & \tau \ \tau \quad \text{Type Application} \\ & | & \Lambda(\alpha : \kappa). \tau \quad \text{Type Abstraction} \\ & | & \forall(\alpha : \kappa). \tau \quad \text{Universal Quantification} \\ & | & \exists(\alpha : \kappa). \tau \quad \text{Existential Quantification} \\ & | & \mu(\alpha : \kappa). \tau \quad \text{Type Recursion} \\ & | & \underline{c}S \quad \text{Type Constructor} \\ & | & \Gamma \quad \text{Trait Type} \\ & | & \tau.m \quad \text{Trait Type Usage} \end{array}$$

### 3 Trait level

$$\begin{aligned}
m_i &\in \mathcal{I} \\
K &= \{m_i : \kappa_i\}_I \\
T &= \{m_i \triangleq \tau_i\}_I \\
S &= \{m_i : \tau_i\}_I \\
W &= \{\Gamma_i\}_I \\
\Gamma &= \langle K, T, S, W \rangle \\
\Gamma_\emptyset &= \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle \\
- + - &: \Gamma \rightarrow \Gamma \rightarrow \Gamma \\
\Gamma_\emptyset + \Gamma' &= \Gamma' \\
\Gamma' + \Gamma_\emptyset &= \Gamma' \\
\langle K, T, S, W \rangle + \Gamma &= \langle K, T, S, W \cup \Gamma \rangle \\
-[-]_\kappa &: \Gamma \rightarrow \kappa + \perp \\
\langle \{n_i : k_i\}_I, \rightarrow, \rightarrow, \rightarrow \rangle [n]_\kappa &= k_j & \exists j \in I, n_j = n \\
\langle \rightarrow, \rightarrow, \rightarrow, \Gamma_I \rangle [n]_\kappa &= k & \exists i \in I, \Gamma_i[n]_\kappa = k \\
-[-]_\tau &: \Gamma \rightarrow \tau + \perp \\
\langle \rightarrow, \{n_i \triangleq t_i\}_I, \rightarrow, \rightarrow \rangle [n]_\tau &= t_j & \exists j \in I, n_j = n \\
\langle \rightarrow, \rightarrow, \rightarrow, \Gamma_I \rangle [n]_\tau &= t & \exists i \in I, \Gamma_i[n]_\tau = t \\
-[-]_\sigma &: \Gamma \rightarrow \sigma + \perp \\
\langle \rightarrow, \rightarrow, \{n_i : t_i\}_I, \rightarrow \rangle [n]_\sigma &= t_j & \exists j \in I, n_j = n \\
\langle \rightarrow, \rightarrow, \rightarrow, \Gamma_I \rangle [n]_\sigma &= t & \exists i \in I, \Gamma_i[n]_\sigma = t \\
& \quad s
\end{aligned}$$

### 4 Expression level

$\underline{c}$	$\in \mathcal{C}$	Constructor names
$\alpha$	$\in \mathcal{I}$	Variable names
$\epsilon$	$= \alpha$	Variable
	$\lambda \alpha. \epsilon$	Function
	$\zeta. \epsilon$	Method
	$\epsilon \epsilon$	Application
	<b>let</b> $\alpha : \tau = \epsilon$ <b>in</b> $\epsilon$	Typed let binding
	<b>let</b> $\alpha = \epsilon$ <b>in</b> $\epsilon$	Let binding
	<b>when</b> $(\alpha). \{\tau_i \triangleright \epsilon_i\}_I$	Smart cast
	$\{\tau, \epsilon\}$	Pack
	<b>let</b> $\{\tau, \alpha\} = \epsilon$ <b>in</b> $\epsilon$	Unpack
	$\Sigma$	Trait term
	$\epsilon.m$	Trait term Usage

With the expression trait defined by:

$$\begin{aligned} M &= \{m_i \triangleq \epsilon_i\}_I \\ \Sigma &= \Gamma \circledast M \end{aligned}$$

## 5 Illustration

### 5.1 Algebraic datatype

```
type Nil      = data Nil
type Cons a = data Cons (head:a) (tail:List a)
type List a = Nil | Cons a
```

#### *Type kind*

```
Nil   : ★
Cons  : ★ → ★
List  : ★ → ★
```

#### *Type definition*

```
Nil    $\triangleq$  Nil{ }
Cons   $\triangleq$   $\Lambda(\alpha : \star). \text{Cons}\{\text{head} : \alpha; \text{tail} : \text{List } \alpha\}$ 
List   $\triangleq$   $\Lambda(\alpha : \star). \mu(\xi : \star). \text{Nil}\{\} + \text{Cons}\{\text{head} : \alpha; \text{tail} : \xi\}$ 
```

#### *Function definition*

```
Nil   : Nil
Cons  :  $\forall(\alpha : \star). \alpha \rightarrow \text{List } \alpha \rightarrow \text{Cons } \alpha$ 
```

### 5.2 Function signature

```
sig emptyList : forall a. unit -> List a
sig isEmpty   : forall a. self -> bool for List a
```

```
emptyList :  $\forall(\alpha : \star). \text{unit} \rightarrow \text{List } \alpha$ 
isEmpty   :  $\forall(\alpha : \star). \text{List } \alpha \rightarrow \text{bool}$ 
```

### 5.3 Closed trait

```
trait Access a for List a {
  sig head : self -> Option a
}
```

which is equivalent to the following type definition:

```
type Access a = trait for List a {
  sig head : self -> Option a
}
```

#### *Type kind*

Access :  $\star \rightarrow \{\}$

#### *Type definition*

Access  $\triangleq \Lambda(\alpha : \star). \langle \emptyset, \emptyset, \{\text{head} : \text{List } \alpha \multimap \text{Option } \alpha\}, \emptyset \rangle$

### 5.4 Open trait

```
trait Set a {
  sig new : self
  sig contains : self -> a -> bool
}
```

#### *Type kind*

Set :  $\star \rightarrow \star$

#### *Type definition*

Set  $\triangleq \Lambda(\alpha : \star). \exists(\text{self} : \star). \langle \emptyset, \emptyset, \{\text{new} : \text{self}, \text{contains} : \text{self} \multimap \alpha \rightarrow \text{bool}\}, \emptyset \rangle$

### 5.5 Trait with and abstract type

```
trait Pure a {
  kind t = type -> type
  sig pure : a -> t a
}
```

#### *Type kind*

Pure :  $\star \rightarrow \{t : \star \rightarrow \star\}$

#### *Type definition*

$$\text{Pure} \triangleq \Lambda(\alpha : \star). \exists(\mathbf{t} : \star \rightarrow \star). \langle \emptyset, \emptyset, \{\text{pure} : \alpha \rightarrow \mathbf{t} \alpha\}, \emptyset \rangle$$

## 5.6 Trait with requirement

```
trait Applicative (t:type->type) with Functor t {
  sig pure : forall a.a -> t a
}
```

$$\text{Applicative} \triangleq \Lambda(\mathbf{t} : \star \rightarrow \star). \langle \emptyset, \emptyset, \{\text{pure} : \forall(\alpha : \star). \alpha \rightarrow \mathbf{t} \alpha\}, \{\text{Functor } \mathbf{t}\} \rangle$$

## 5.7 Function signature with requirement

```
sig eq : forall a. List a -> List a -> bool with Equatable a
```

$$\text{eq} : \langle \{\text{eq} : \star\}, \{\text{eq} \triangleq \forall(\alpha : \star). \text{List } \alpha \rightarrow \text{List } \alpha \rightarrow \text{bool}\}, \emptyset, \{\text{Equatable } \alpha\} \rangle.\text{eq}$$

# 6 Type system

## 6.1 $\Gamma$ and projections

$$\begin{aligned} \mathcal{K}_\downarrow[-] &: \Gamma \rightarrow K \\ \mathcal{K}_\uparrow[-] &: K \rightarrow \Gamma \\ \mathcal{T}_\downarrow[-] &: \Gamma \rightarrow T \\ \mathcal{T}_\uparrow[-] &: T \rightarrow \Gamma \\ \mathcal{S}_\downarrow[-] &: \Gamma \rightarrow S \\ \mathcal{S}_\uparrow[-] &: S \rightarrow \Gamma \\ \mathcal{W}_\downarrow[-] &: \Gamma \rightarrow W \end{aligned}$$

## 6.2 Kind inclusion

---

$$\frac{}{\star \subseteq_\kappa \star}(\text{refl}_\star) \quad \frac{}{K \subseteq_\kappa \star}(\top_\star) \quad \frac{k_3 \subseteq_\kappa k_1 \quad k_2 \subseteq_\kappa k_4}{k_1 \rightarrow k_2 \subseteq_\kappa k_3 \rightarrow k_4}(\rightarrow_\star)$$

$$\frac{\forall j \in J, \exists i \in I, n_i = n'_j \quad k_i \subseteq k'_j}{\{n_i : k_i\}_I \subseteq_\kappa \{n'_j : k'_j\}_J}(\text{trait}_\star)$$


---

### 6.3 Type rules

---

$$\begin{array}{c}
\frac{\Gamma[n]_{\kappa} = k' \quad k \subseteq_{\kappa} k'}{\Gamma \vdash n :_{\kappa} k}(\text{Identity}) \quad \frac{\Gamma \vdash t_1 :_{\kappa} \star \quad \Gamma \vdash t_2 :_{\kappa} \star}{\Gamma \vdash t_1 \rightarrow t_2 :_{\kappa} \star}(\rightarrow\text{-type}) \\
\\
\frac{\Gamma \vdash t_1 :_{\kappa} \star \quad \Gamma \vdash t_2 :_{\kappa} \star}{\Gamma \vdash t_1 \multimap t_2 :_{\kappa} \star}(\multimap\text{-type}) \quad \frac{\Gamma \vdash t_1 :_{\kappa} \star \quad \Gamma \vdash t_2 :_{\kappa} \star}{\Gamma \vdash t_1 + t_2 :_{\kappa} \star}(+\text{-type}) \\
\\
\frac{\Gamma \vdash t_1 :_{\kappa} k' \rightarrow k \quad \Gamma \vdash t_2 :_{\kappa} k'}{\Gamma \vdash t_1 \ t_2 :_{\kappa} k}(\text{apply-type}) \\
\\
\frac{k_1 \subseteq_{\kappa} k \quad \Gamma \oplus \mathcal{K}_{\uparrow}[\{a : k\}] \vdash t :_{\kappa} k_2}{\Gamma \vdash \Lambda(a : k).t :_{\kappa} k_1 \rightarrow k_2}(\Lambda\text{-type}) \quad \frac{\Gamma \oplus \mathcal{K}_{\uparrow}[\{a : k\}] \vdash t :_{\kappa} k}{\Gamma \vdash \forall(a : k).t :_{\kappa} k}(\forall\text{-type}) \\
\\
\frac{\Gamma \oplus \mathcal{K}_{\uparrow}[\{a : k\}] \vdash t :_{\kappa} k}{\Gamma \vdash \exists(a : k).t :_{\kappa} k}(\exists\text{-type}) \quad \frac{\Gamma \oplus \mathcal{K}_{\uparrow}[\{a : k\}] \vdash t :_{\kappa} k}{\Gamma \vdash \mu(a : k).t :_{\kappa} k}(\mu\text{-type}) \\
\\
\frac{\Gamma' = \langle K, T, S, W \rangle \quad K \subseteq K' \quad \forall(n, t) \in T, \Gamma' \vdash t :_{\kappa} K[n] \quad \forall(-, s) \in S, \Gamma' \vdash s :_{\kappa} \star \quad \forall w \in W, \Gamma_{\emptyset} \vdash w :_{\kappa} \{\}}{\Gamma \vdash \Gamma' :_{\kappa} K'}(\text{trait-type}) \\
\\
\frac{\forall i \in I, \Gamma \vdash S[m_i] : \star}{\Gamma \vdash \underline{c}S_I :_{\kappa} \star}(\text{const-type}) \quad \frac{\Gamma \vdash t_1 :_{\kappa} K \quad \mathcal{K}_{\uparrow}[K'] \vdash n : k}{\Gamma \vdash t_1.n :_{\kappa} k}(\text{use-type})
\end{array}$$


---

### 6.4 Type reduction

---

$$\begin{array}{c}
\frac{\Gamma \vdash t_1 \longrightarrow \Lambda(x : k).t_4 \quad \Gamma \vdash t_2 : k}{\Gamma \vdash t_1 \ t_2 \longrightarrow t_4[t_2/x]}(\text{red-apply}) \\
\\
\frac{\Gamma[n]_{\tau} = t' \quad \Gamma \vdash t' \longrightarrow t''}{\Gamma \vdash n \longrightarrow t''}(\text{red-var}) \quad \frac{\Gamma \vdash t[\mu(\alpha).t/\alpha] \longrightarrow t'}{\Gamma \vdash \mu(\alpha).t \longrightarrow t'}(\text{red-}\mu) \\
\\
\frac{\Gamma \vdash t_1 \longrightarrow \Gamma' \quad \Gamma' \oplus \Gamma \vdash n \longrightarrow t_2}{\Gamma \vdash t_1.n \longrightarrow t_2}(\text{red-access-var}) \quad \frac{}{\Gamma \vdash t \longrightarrow t}(\text{id})
\end{array}$$


---

## 6.5 Type inclusion

---

$$\frac{\Gamma \vdash t :_{\kappa} \star}{\Gamma \vdash t \subseteq t} (\text{sub-refl}) \quad \frac{\forall j \in J, \exists i \in I, m_i = m'_j, \Gamma \vdash t_i \subseteq t'_j}{\Gamma \vdash \underline{c}\{m_i : t_i\}_I \subseteq \underline{c}\{m'_i : t'_i\}_J} (\text{sub-const})$$

$$\frac{\Gamma \vdash t_1 \longrightarrow t_3 \quad \Gamma \vdash t_3 \subseteq t_2}{\Gamma \vdash t_1 \subseteq t_2} (\text{app-l}) \quad \frac{\Gamma \vdash t_2 \longrightarrow t_3 \quad \Gamma \vdash t_1 \subseteq t_3}{\Gamma \vdash t_1 \subseteq t_2} (\text{app-r})$$

$$\frac{\Gamma \vdash t_3 \subseteq t_1 \quad \Gamma \vdash t_2 \subseteq t_4}{\Gamma \vdash t_1 \rightarrow t_2 \subseteq t_3 \rightarrow t_4} (\text{sub-}\rightarrow) \quad \frac{\Gamma \vdash t_3 \curlywedge t_1 \quad \Gamma \vdash t_2 \curlywedge t_4}{\Gamma \vdash t_1 \curlywedge t_2 \subseteq t_3 \curlywedge t_4} (\text{sub-}\curlywedge)$$

$$\frac{\Gamma \vdash t_1 \subseteq t_3 \quad \Gamma \vdash t_2 \subseteq t_3}{\Gamma \vdash t_1 + t_2 \subseteq t_3} (\text{sub-}+l) \quad \frac{\Gamma \vdash t_1 \subseteq t_2}{\Gamma \vdash t_1 \subseteq t_2 + t_3} (\text{sub-}+r1)$$

$$\frac{\Gamma \vdash t_1 \subseteq t_3}{\Gamma \vdash t_1 \subseteq t_2 + t_3} (\text{sub-}+r2) \quad \frac{\Gamma \oplus \mathcal{K}_{\uparrow}[\{a : \star\}] \vdash t_1 \subseteq t_2}{\Gamma \vdash \mu(a).t_1 \subseteq \mu(a).t_2} (\text{sub-}\mu)$$

$$\frac{\Gamma \vdash t_1[\mu(a).t_1/a] \subseteq t_2}{\Gamma \vdash \mu(a).t_1 \subseteq t_2} (\text{sub-}\mu-l) \quad \frac{\Gamma \vdash t_1 \subseteq t_2[\mu(a).t_2/a]}{\Gamma \vdash t_1 \subseteq \mu(a).t_2} (\text{sub-}\mu-r)$$

$$\frac{k' \subseteq_{\kappa} k \quad \Gamma \oplus \mathcal{K}_{\uparrow}[\{x : k\}] \vdash t_1 \subseteq t_2}{\Gamma \vdash \Lambda(x : k).t_1 \subseteq \Lambda(x : k').t_2} (\text{sub-}\Lambda)$$

$$\frac{k \subseteq_{\kappa} k' \quad \Gamma \oplus \mathcal{K}_{\uparrow}[\{x : k\}] \vdash t_1 \subseteq t_2}{\Gamma \vdash \forall(x : k).t_1 \subseteq \forall(x : k').t_2} (\text{sub-}\forall)$$

$$\frac{k \subseteq_{\kappa} k' \quad \Gamma \oplus \mathcal{K}_{\uparrow}[\{x : k\}] \vdash t_1 \subseteq t_2}{\Gamma \vdash \exists(x : k).t_1 \subseteq \exists(x : k').t_2} (\text{sub-}\exists)$$

$$\frac{\begin{array}{l} \mathcal{K}_{\downarrow}[\Gamma_1] \subseteq \mathcal{K}_{\downarrow}[\Gamma_2] \\ \forall(n, t) \in \mathcal{T}_{\downarrow}[\Gamma_1], \exists(n', t') \in \mathcal{T}_{\downarrow}[\Gamma_2], n = n', \Gamma \vdash t' \subseteq t \\ \forall(n, t) \in \mathcal{S}_{\downarrow}[\Gamma_1], \exists(n', t') \in \mathcal{S}_{\downarrow}[\Gamma_2], n = n', \Gamma \vdash t' \subseteq t \end{array}}{\Gamma \vdash \Gamma_1 \subseteq \Gamma_2} (\text{sub-trait})$$


---

## 6.6 Expression rules

---

$$\frac{\Gamma[e]_\sigma = t' \quad \Gamma \vdash t \subseteq t'}{\Gamma \vdash e : t}(\text{id}) \quad \frac{\Gamma \vdash n : t_1 \rightarrow t_2 \quad \Gamma \vdash a : t_3 \quad \Gamma \vdash t_3 \subseteq t_1}{\Gamma \vdash n \ a : t_2}(\text{app})$$

$$\frac{\Gamma \vdash n : t_1 \multimap t_2 \quad \Gamma \vdash a : t_3 \quad \Gamma \vdash t_3 \subseteq t_1}{\Gamma \vdash a \ n : t_2}(\text{call})$$

$$\frac{\Gamma \oplus \mathcal{S}_\uparrow[\{n : t_1\}] \vdash a : t_2}{\Gamma \vdash \lambda n. a : t_1 \rightarrow t_2}(\text{abstr}) \quad \frac{\Gamma \oplus \mathcal{S}_\uparrow[\{\mathbf{self} : t_1\}] \vdash a : t_2}{\Gamma \vdash \zeta. a : t_1 \multimap t_2}(\text{meth})$$

$$\frac{\Gamma \vdash r : \underline{c}S \quad \mathcal{S}_\uparrow[S] \oplus \Gamma \vdash n : t}{\Gamma \vdash r.n : t}(\text{access}) \quad \frac{\Gamma \vdash r : \Gamma' \quad \Gamma' \oplus \Gamma \vdash n : t}{\Gamma \vdash r.n : t}(\text{use})$$

$$\frac{\Gamma \vdash e : \forall(a : k). t_1 \quad \Gamma \vdash t_2 :_\kappa k}{\Gamma \vdash e : t_1[t_2/a]}(\forall\text{-elim}) \quad \frac{\mathcal{K}_\uparrow[\{a : k\}] \oplus \Gamma \vdash e : t}{\Gamma \vdash e : \forall(a : k). t}(\forall\text{-intro})$$

$$\frac{\Gamma \vdash e_1 : \exists(a : k). t_1 \quad \mathcal{K}_\uparrow[\{A : k\}] \oplus \mathcal{S}_\uparrow[\{a : t_1\}] \oplus \Gamma \vdash e_2 : t_2 \quad A \notin \mathbf{ftv}(t_2)}{\Gamma \vdash \mathbf{let} \ \{A, a\} = e_1 \ \mathbf{in} \ e_2 : t_2}(\exists\text{-elim})$$

$$\frac{\Gamma \vdash t_1 :_\kappa k \quad \Gamma \vdash e : t_2[t_1/a]}{\Gamma \vdash \{t_1, e\} : \exists(a : k). t_2}(\exists\text{-intro}) \quad \frac{\Gamma \vdash e_1 : t_1 \quad \Gamma \oplus \mathcal{T}_\uparrow[\{a : t_1\}] \vdash e_2 : t_2}{\Gamma \vdash \mathbf{let} \ a = e_1 \ \mathbf{in} \ e_2 : t_2}(\text{let-in})$$

$$\frac{\Gamma \vdash e_1 : t_1 \quad \Gamma \oplus \mathcal{T}_\uparrow[\{a : t_1\}] \vdash e_2 : t_2}{\Gamma \vdash \mathbf{let} \ a : t_1 = e_1 \ \mathbf{in} \ e_2 : t_2}(\text{let-in})$$

$$\frac{\forall i \in I, \Gamma \vdash a : t_i \quad \mathcal{S}_\uparrow[\{a : t_i\}] \oplus \Gamma \vdash e_i : t}{\Gamma \vdash \mathbf{when}(a). \{t_i \triangleright e_i\}_I : t}(\text{when})$$

$$\frac{\Gamma \vdash \Gamma_1 \subseteq \Gamma_2 \quad \Gamma_1[m_i]_\sigma = t_i \quad \forall i \in I, \Gamma_1 \oplus \Gamma \vdash e_i : t_i}{\Gamma \vdash \Gamma_1 \otimes \{m_i \triangleq e_i\}_I : \Gamma_2}(\text{trait})$$


---