The Lambë programming language

D. Plaindoux

April 1, 2021

1 Kind level

$$
\begin{array}{rcl}
\kappa & = & \star \\
& \mid & \kappa \rightarrow \kappa \\
& \mid & K
\end{array}
$$

2 Type level

$$
\begin{array}{rcll}
\underline{c} & \in & \mathcal{C} & \text{Constructor names} \\
\alpha & \in & \mathcal{I} & \text{Variable names} \\
\\
\tau & = & \alpha & \text{Variable or Constant} \\
& \mid & \tau \rightarrow \tau & \text{Function Type} \\
& \mid & \tau \looparrowright \tau & \text{Method Type} \\
& \mid & \tau + \tau & \text{Sum type} \\
& \mid & \tau\ \tau & \text{Type Application} \\
& \mid & \Lambda(\alpha : \kappa).\tau & \text{Type Abstraction} \\
& \mid & \forall(\alpha : \kappa).\tau & \text{Universal Quantification} \\
& \mid & \exists(\alpha : \kappa).\tau & \text{Existential Quantification} \\
& \mid & \mu(\alpha : \kappa).\tau & \text{Type Recursion} \\
& \mid & \underline{c}S & \text{Type Constructor} \\
& \mid & \Gamma & \text{Trait Type} \\
& \mid & \tau.m & \text{Trait Type Usage}
\end{array}
$$

3 Trait level

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

4 Expression level

$$
\begin{array}{rcll}
\underline{c} & \in & \mathcal{C} & \text{Constructor names} \\
\alpha & \in & \mathcal{I} & \text{Variable names} \\
 & & & \\
\epsilon & = & \alpha & \text{Variable} \\
 & | & \lambda\alpha.\epsilon & \text{Function} \\
 & | & \zeta.\epsilon & \text{Method} \\
 & | & \epsilon\ \epsilon & \text{Application} \\
 & | & \texttt{let}\ \alpha\ \texttt{=}\ \epsilon\ \texttt{in}\ \epsilon & \text{Let binding} \\
 & | & \texttt{when}(\alpha).\{\tau_i \rhd \epsilon_i\}_I & \text{Smart cast} \\
 & | & \{\tau, \epsilon\} & \text{Pack} \\
 & | & \texttt{let}\ \{\tau, \alpha\}\ \texttt{=}\ \epsilon\ \texttt{in}\ \epsilon & \text{Unpack} \\
 & | & \Sigma & \text{Trait term} \\
 & | & \epsilon.m & \text{Trait term Usage}
\end{array}
$$

With the expression trait defined by:

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

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

$$
\begin{array}{rcl}
\texttt{Nil} & : & \star \\
\texttt{Cons} & : & \star \to \star \\
\texttt{List} & : & \star \to \star
\end{array}
$$

Type definition

$$
\begin{array}{rcl}
\texttt{Nil} & \triangleq & \underline{\texttt{Nil}}\{\} \\
\texttt{Cons} & \triangleq & \Lambda(\alpha:\star).\underline{\texttt{Cons}}\{\texttt{head}:\alpha;\texttt{tail}:\texttt{List}\ \alpha\} \\
\texttt{List} & \triangleq & \Lambda(\alpha:\star).\mu(\xi:\star).\underline{\texttt{Nil}}\{\} + \underline{\texttt{Cons}}\{\texttt{head}:\alpha;\texttt{tail}:\xi\}
\end{array}
$$

Function definition

$$
\begin{array}{rcl}
\texttt{Nil} & : & \texttt{Nil} \\
\texttt{Cons} & : & \forall(\alpha:\star).\alpha \to \texttt{List}\ \alpha \to \texttt{Cons}\ \alpha
\end{array}
$$

5.2 Algebraic datatype ... again

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

Type kind

$$
\texttt{List} \ : \ \star \to \star
$$

Type definition

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

Function definition

$$\begin{aligned}\texttt{Nil} &: \forall(\alpha:\star).\texttt{List}\ \alpha \\ \texttt{Cons} &: \forall(\alpha:\star).\alpha \to \texttt{List}\ \alpha \to \texttt{List}\ \alpha\end{aligned}$$

5.3 Function signature

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

$$\begin{aligned}\texttt{emptyList} &: \forall(\alpha:\star).\texttt{unit} \to \texttt{List}\ \alpha \\ \texttt{isEmpty} &: \forall(\alpha:\star).\texttt{List}\ \alpha \nrightarrow \texttt{bool}\end{aligned}$$

5.4 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

$$\texttt{Access} : \star \to \{\}$$

Type definition

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

5.5 Open trait

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

Type kind

$$\texttt{Set} : \star \to \star$$

Type definition

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

5.6 Trait with and abstract type

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

Type kind

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

Type definition

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

5.7 Trait with requirement

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

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

5.8 Function signature with requirement

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

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

6 Type system

6.1 Γ and projections

$$
\begin{array}{rcl}
\mathcal{K}_{\downarrow}[\_] & : & \Gamma \to K \\
\mathcal{K}_{\uparrow}[\_] & : & K \to \Gamma \\
\mathcal{T}_{\downarrow}[\_] & : & \Gamma \to T \\
\mathcal{T}_{\uparrow}[\_] & : & T \to \Gamma \\
\mathcal{S}_{\downarrow}[\_] & : & \Gamma \to S \\
\mathcal{S}_{\uparrow}[\_] & : & S \to \Gamma \\
\mathcal{W}_{\downarrow}[\_] & : & \Gamma \to W
\end{array}
$$

6.2 Kind inclusion

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

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

6.3 Type rules

$$\frac{\Gamma[n]_\kappa = k' \quad k \subseteq_\kappa k'}{\Gamma \vdash n :_\kappa k}(\text{Identity}) \qquad \frac{\Gamma \vdash t_1 :_\kappa \star \quad \Gamma \vdash t_2 :_\kappa \star}{\Gamma \vdash t_1 \to t_2 :_\kappa \star}(\to\text{-type})$$

$$\frac{\Gamma \vdash t_1 :_\kappa \star \quad \Gamma \vdash t_2 :_\kappa \star}{\Gamma \vdash t_1 \hookrightarrow t_2 :_\kappa \star}(\hookrightarrow\text{-type}) \qquad \frac{\Gamma \vdash t_1 :_\kappa \star \quad \Gamma \vdash t_2 :_\kappa \star}{\Gamma \vdash t_1 + t_2 :_\kappa \star}(+\text{-type})$$

$$\frac{\Gamma \vdash t_1 :_\kappa k' \to k \quad \Gamma \vdash t_2 :_\kappa k'}{\Gamma \vdash t_1\ t_2 :_\kappa k}(\text{apply-type})$$

$$\frac{k_1 \subseteq_\kappa k \quad \Gamma \oplus \mathcal{K}_\uparrow[\{a : k\}] \vdash t :_\kappa k_2}{\Gamma \vdash \Lambda(a : k).t :_\kappa k_1 \to k_2}(\Lambda\text{-type}) \qquad \frac{\Gamma \oplus \mathcal{K}_\uparrow[\{a : k\}] \vdash t :_\kappa k}{\Gamma \vdash \forall(a : k).t :_\kappa k}(\forall\text{-type})$$

$$\frac{\Gamma \oplus \mathcal{K}_\uparrow[\{a : k\}] \vdash t :_\kappa k}{\Gamma \vdash \exists(a : k).t :_\kappa k}(\exists\text{-type}) \qquad \frac{\Gamma \oplus \mathcal{K}_\uparrow[\{a : k\}] \vdash t :_\kappa k}{\Gamma \vdash \mu(a : k).t :_\kappa k}(\mu\text{-type})$$

$$\frac{\begin{array}{c}\Gamma' = \langle K, T, S, W\rangle \quad K \subseteq K' \quad \forall(n,t) \in T, \Gamma' \vdash t :_\kappa K[n] \\ \forall(\_, s) \in S, \Gamma' \vdash s :_\kappa \star \quad \forall w \in W, \Gamma_\emptyset \vdash w :_\kappa \{\}\end{array}}{\Gamma \vdash \Gamma' :_\kappa K'}(\text{trait-type})$$

$$\frac{\forall i \in I, \Gamma \vdash S[m_i] : \star}{\Gamma \vdash \underline{\mathsf{c}}S_I :_\kappa \star}(\text{const-type}) \qquad \frac{\Gamma \vdash t_1 :_\kappa K \quad \mathcal{K}_\uparrow[K'] \vdash n : k}{\Gamma \vdash t_1.n :_\kappa k}(\text{use-type})$$

6.4 Type reduction

$$\frac{\Gamma \vdash t_1 \longrightarrow \Lambda(x : k).t_3 \quad \Gamma \vdash t_2 : k \quad \Gamma \vdash t_3[t_2/x] \longrightarrow t_4}{\Gamma \vdash t_1\ t_2 \longrightarrow t_4}(\text{red-apply})$$

$$\frac{\Gamma[n]_\tau = t' \quad \Gamma \vdash t' \longrightarrow t''}{\Gamma \vdash n \longrightarrow t''}(\text{red-var}) \qquad \frac{\Gamma \vdash t[\mu(\alpha).t/\alpha] \longrightarrow t'}{\Gamma \vdash \mu(\alpha).t \longrightarrow t'}(\text{red-}\mu)$$

$$\frac{\Gamma \vdash t_1 \longrightarrow \Gamma' \quad \Gamma' \oplus \Gamma \vdash n \longrightarrow t_2}{\Gamma \vdash t_1.n \longrightarrow t_2}(\text{red-access-var}) \qquad \frac{}{\Gamma \vdash t \longrightarrow t}(\text{id})$$

6.5 Type inclusion

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

$$\frac{\Gamma \vdash t_1 \longrightarrow t_3 \quad \Gamma \vdash t_3 \subseteq t_2}{\Gamma \vdash t_1 \subseteq t_2}(\text{app-l}) \qquad \frac{\Gamma \vdash t_2 \longrightarrow t_3 \quad \Gamma \vdash t_1 \subseteq t_4}{\Gamma \vdash t_1 \subseteq t_2}(\text{app-r})$$

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

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

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

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

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

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

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

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

6.6 Expression rules

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

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

$$\frac{\Gamma \oplus \mathcal{S}_\uparrow[\{n : t_1\}] \vdash a : t_2}{\Gamma \vdash \lambda n.a : t_1 \to t_2}(\text{abstr}) \qquad \frac{\Gamma \oplus \mathcal{S}_\uparrow[\{\texttt{self} : t_1\}] \vdash a : t_2}{\Gamma \vdash \zeta.a : t_1 \looparrowright t_2}(\text{meth})$$

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

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

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

$$\frac{\Gamma \vdash t_1 :_\kappa k \quad \Gamma \vdash e : t_2[t_1/a]}{\Gamma \vdash \{t_1, e\} : \exists(a : k).t_2}(\exists\text{-intro}) \qquad \frac{\Gamma \vdash e_1 : t1 \quad \Gamma \oplus \mathcal{T}_\uparrow[\{a : t_1\}] \vdash e_2 : t_2}{\Gamma \vdash \texttt{let}\ a\ \texttt{=}\ e_1\ \texttt{in}\ e_2 : t_2}(\text{let-in})$$

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

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