# 1 Evaluation Order

**Definition of Class and Constructor**

$$
\begin{aligned}
\text{parentClass} &= \lambda \text{r} : \text{parentRep} \,.\, \lambda \text{self} : \text{parent} \,.\, \langle\langle \mathit{pmethods} \rangle\rangle \\
\text{childClass} &= \lambda \text{r} : \text{childRep} \,.\, \lambda \text{self} : \text{child} \,.\, \text{let super} = \text{parentClass r self in } \langle\langle \mathit{cmethods} \rangle\rangle \\
\text{newChild} &= \lambda_ : \text{Unit} \,.\, \text{let r} = \langle\langle \mathit{cvars} \rangle\rangle \text{ in fix } (\text{childClass r})
\end{aligned}
$$

$$
\begin{aligned}
\text{parentClass} &: \text{parentRep} \to \text{parent} \to \text{parent} \\
\text{childClass} &: \text{childRep} \to \text{child} \to \text{child} \\
\text{newChild} &: \text{Unit} \to \text{child}
\end{aligned}
$$

**Construction of Object**

$$
\begin{aligned}
\text{ic} &= \text{newChild unit} \\
&= \text{fix } (\text{childClass } \langle \mathit{cvars} \rangle) \\
&= \text{fix } (\lambda \,\text{self} : \text{child} \,.\, \text{let super} = \text{parentClass } \langle \mathit{cvars} \rangle \text{ self in } \langle \mathit{cmethods} \rangle) \\
&= \underbrace{\text{fix } \langle f \rangle}_{A} \\
&= [\text{self} \to \text{fix} \langle f \rangle] \langle f'\mathit{s\ body} \rangle \\
&= \text{let super} = \text{parentClass } \langle \mathit{cvars} \rangle (\text{fix} \langle f \rangle) \text{ in } \langle \mathit{cmethods}' \rangle \\
&= \text{let super} = (\lambda \text{self} : \text{parent} \,.\, \langle \mathit{pmethods} \rangle) \underbrace{(\text{fix} \langle f \rangle)}_{B} \text{ in } \langle \mathit{cmethods}' \rangle
\end{aligned}
$$

A を評価するためには結局 B を評価する必要があり、これは発散する。



# 2 Dummy lambda-abstraction

**Definition of Class and Constructor**

$$
\begin{aligned}
\text{parentClass} &= \lambda \text{r} : \text{parentRep} \,.\, \lambda \text{self} : \text{unit} \to \text{parent} \,.\, \lambda_ : \text{Unit} \,.\, \langle\langle \mathit{pmethods} \rangle\rangle \\
\text{childClass} &= \lambda \text{r} : \text{childRep} \,.\, \lambda \text{self} : \text{unit} \to \text{child} \,.\, \lambda_ : \text{Unit} \,. \\
&\qquad \text{let super} = \text{parentClass r self unit in } \langle\langle \mathit{cmethods} \rangle\rangle \\
\text{newChild} &= \lambda_ : \text{Unit} \,.\, \text{let r} = \langle\langle \mathit{cvars} \rangle\rangle \text{ in fix } (\text{childClass r}) \text{ unit}
\end{aligned}
$$

$$
\begin{aligned}
\text{parentClass} &: \text{parentRep} \to (\text{Unit} \to \text{parent}) \to \text{Unit} \to \text{parent} \\
\text{childClass} &: \text{childRep} \to (\text{Unit} \to \text{child}) \to \text{Unit} \to \text{child} \\
\text{newChild} &: \text{Unit} \to \text{child}
\end{aligned}
$$

**Construction of Object**

$$
\begin{aligned}
\text{ic} &= \text{newChild unit} \\
&= \text{fix } (\text{childClass } \langle \mathit{cvars} \rangle) \text{ unit} \\
&= \text{fix } (\lambda \,\text{self} : \text{unit} \to \text{child} \,.\, \lambda_ : \text{Unit} \,. \\
&\qquad \text{let super} = \text{parentClass } \langle \mathit{cvars} \rangle \text{ self unit in } \langle \mathit{cmethods} \rangle) \text{ unit} \\
&= \underbrace{\text{fix } \langle f \rangle}_{A} \text{ unit} \\
&= [\text{self} \to \text{fix} \langle f \rangle] \langle f'\mathit{s\ body} \rangle \text{ unit} \\
&= (\lambda_ : \text{Unit} \,.\, \text{let super} = \text{parentClass } \langle \mathit{cvars} \rangle (\text{fix} \langle f \rangle) \text{ unit in } \langle \mathit{cmethods}' \rangle) \text{ unit} \\
&= \text{let super} = \text{parentClass } \langle \mathit{cvars} \rangle (\text{fix} \langle f \rangle) \text{ unit in } \langle \mathit{cmethods}' \rangle \\
&= \text{let super} = (\lambda \text{self} : \text{unit} \to \text{parent} \,.\, \lambda_ : \text{Unit} \,.\, \langle \mathit{pmethods} \rangle) \\
&\qquad \underbrace{(\text{fix} \langle f \rangle)}_{B} \text{ unit in } \langle \mathit{cmethods}' \rangle \\
&= \text{let super} = (\lambda \text{self} : \text{unit} \to \text{parent} \,.\, \lambda_ : \text{Unit} \,.\, \langle \mathit{pmethods} \rangle) \\
&\qquad \underbrace{(\lambda_ : \text{Unit} \,.\, \text{let super} = \text{parentClass } \langle \mathit{cvars} \rangle (\text{fix} \langle f \rangle) \text{ unit in } \langle \mathit{cmethods}' \rangle)}_{value} \text{ unit} \\
&\qquad \text{in } \langle \mathit{cmethods}' \rangle \\
&= \text{let super} = (\lambda_ : \text{Unit} \,.\, \langle \mathit{pmethods}' \rangle) \text{ unit in } \langle \mathit{cmethods}' \rangle \\
&= \text{let super} = \langle \mathit{pmethods}' \rangle \text{ in } \langle \mathit{cmethods}' \rangle \\
&= \langle \mathit{cmethods}'' \rangle
\end{aligned}
$$

この場合の A と B はそれのみで評価すると value になる。$\langle \mathit{cmethods}'' \rangle$ は value であるから、オブジェクトを発散せずに作成することができる。

## 3 Reference

**Wrong Definition of Class and Constructor**

次のように定義を行うと間違いである。なぜなら、childRep <: parentRep は成立するが、Ref child <: Ref parent は成立しない。(参照の Subtyping は同じ型でしか成立しない。)

$$
\begin{aligned}
\text{parentClass} &= \lambda \text{r} : \text{parentRep} \,.\, \lambda \text{self} : \text{Ref parent} \,.\, \langle\langle \mathit{pmethods} \rangle\rangle \\
\text{childClass} &= \lambda \text{r} : \text{childRep} \,.\, \lambda \text{self} : \text{Ref child} \,. \\
&\quad \text{let super} = \text{parentClass r self in} \langle\langle \mathit{cmethods} \rangle\rangle \\
\text{newChild} &= \lambda _ : \text{Unit} \,.\, \text{let r} = \langle\langle \mathit{cvars} \rangle\rangle \text{ in} \\
&\quad\quad \text{let cAux} = \text{ref } \langle \mathit{dummy} \rangle \text{ in} \\
&\quad (\text{cAux} := (\text{childClass r cAux}); \text{!cAux})
\end{aligned}
$$

parentClass に渡される参照は普通実際のところ読み込みにしか使用しない。また、Source T とは T 型のオブジェクトをセル (メモリ) から読みとれる性質を表しており、Source は covariant かつ Ref T <: Source T である。このことから上記で定義した関数の引数の型を Ref T から Source T に変更すれば、型エラーを起こさず、その型でも望むべき性質は実装可能である。[1]

**Definition of Class and Constructor**

$$
\begin{aligned}
\text{parentClass} &= \lambda \text{r} : \text{parentRep} \,.\, \lambda \text{self} : \text{Source parent} \,.\, \langle\langle \mathit{pmethods} \rangle\rangle \\
\text{childClass} &= \lambda \text{r} : \text{childRep} \,.\, \lambda \text{self} : \text{Source child} \,. \\
&\quad \text{let super} = \text{parentClass r self in} \langle\langle \mathit{cmethods} \rangle\rangle \\
\text{newChild} &= \lambda _ : \text{Unit} \,.\, \text{let r} = \langle\langle \mathit{cvars} \rangle\rangle \text{ in} \\
&\quad\quad \text{let cAux} = \text{ref } \langle \mathit{dummy} \rangle \text{ in} \\
&\quad (\text{cAux} := (\text{childClass r cAux}); \text{!cAux})
\end{aligned}
$$

$$
\begin{aligned}
\text{parentClass} &: \text{parentRep} \to (\text{Source parent}) \to \text{parent} \\
\text{childClass} &: \text{childRep} \to (\text{Source child}) \to \text{child} \\
\text{newChild} &: \text{Unit} \to \text{child}
\end{aligned}
$$

[1] Ref は Source と Sink の融合した性質である。(15.5) 参照