Typed Lambda Caculus $\lambda^{\times,+,\rightarrow}$

《程序设计语言理论》

张昱

0551-63603804，yuzhang@ustc.edu.cn

中国科学技术大学
计算机科学与技术学院

References

- PFPL
  - Chapters: 4 Statics, 5 Dynamics, 6 Type Safety, 7 Evaluation Dynamics
  - Chapters: 10 Product Types, 11 Sum Types (Void & Unit, Boolean, Enumerate, Options)
- TAPL
- The algebra (and calculus!) of algebraic data types

Outline

- $L^{\text{int,bool}}$
- $\lambda^{\rightarrow}$ (typed lambda calculus)
- Algebraic Data Types
  - Product type
  - Sum type

中国科学技术大学
University of Science and Technology of China

$L^{\text{int,bool}}$：Statics and Dynamics

☐ Syntax

☐ Semantics

☐ Type Safety

$L^{int,bool}_{-}$ Syntax

☐ Syntax

Type τ ::=	$\texttt{int}$	integer
	$\texttt{bool}$	boolean
Term t ::=	$\texttt{z}$	zero
	$s(t)$	successor
	$\texttt{true}$	constant true
	$\texttt{false}$	constant false
	$\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3$	if expression
	$\texttt{iszero}(t)$	zero check

$\mathcal{L}^{\text{int,bool}}$ semantics

Semantics: judgements, rules

- Dynamics
  - Values: judgement **a val**

$$\frac{}{z\ \mathrm{val}}\ (\mathrm{V\text{-}z}) \qquad \frac{t\ \mathrm{val}}{s(t)\ \mathrm{val}}\ (\mathrm{V\text{-}s})$$

Introduction form
of int type
(引入int类型的值)

$$\frac{}{\mathrm{false\ val}}\ (\mathrm{V\text{-}F}) \qquad \frac{}{\mathrm{true\ val}}\ (\mathrm{V\text{-}T})$$

Introduction form
of bool type
(引入bool类型的值)

interpreter.ml

$\mathcal{L}^{\text{int,bool}}$- semantics

Structural Dynamics

按语法结构来归纳计算到closed value

☐ Dynamics

- **Small-step semantics (*structural*)**

Judgement: $t \mapsto t'$

$$\frac{t \mapsto t'}{s(t) \mapsto s(t')}\ (\text{D-}\textsc{s}) \qquad \frac{t_1 \mapsto t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \mapsto \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3}\ (\text{D-}\textsc{if}_1)$$

$$\frac{}{\texttt{if true then } t_2 \texttt{ else } t_3 \mapsto t_2}\ (\text{D-}\textsc{if}_2) \qquad \frac{}{\texttt{if false then } t_2 \texttt{ else } t_3 \mapsto t_3}\ (\text{D-}\textsc{if}_3)$$

$$\frac{t \mapsto t'}{\texttt{iszero}(t) \mapsto \texttt{iszero}(t')}\ (\text{D-}\textsc{iszero}_1) \qquad \frac{}{\texttt{iszero}(z) \mapsto \texttt{true}}\ (\text{D-}\textsc{iszero}_2)$$

$$\frac{}{\texttt{iszero}(s(t)) \mapsto \texttt{false}}\ (\text{D-}\textsc{iszero}_3)$$

Search transitions
决定指令执行次序

Instruction transitions
基础的求值步

interpreter.ml

$L^{\text{int,bool}}$- semantics

Semantics: judgements, rules

- Dynamics
  - **Example**

$$
\begin{array}{lll}
 & \texttt{if iszero}(z)\ \texttt{then}\ s(z)\ \texttt{else}\ z & \\
\mapsto & \texttt{if true then}\ s(z)\ \texttt{else}\ z & (\text{D-}\texttt{iszero}_1) \\
\mapsto & s(z) & (\text{D-}\texttt{if}_2)
\end{array}
$$

interpreter.ml

$\mathcal{L}^{\text{int,bool}}_{-}$ semantics

Semantics: judgements, rules

- ☐ **Statics: Type system (typing rules)**
  - **Judgement** $t : \tau$ 项是良类型的(well-typed)

$$\frac{}{\mathtt{z} : \mathtt{int}}\ (\text{T-z}) \qquad \frac{t : \mathtt{int}}{s(t) : \mathtt{int}}\ (\text{T-s}) \qquad \frac{}{\mathtt{true} : \mathtt{bool}}\ (\text{T-TRUE}) \qquad \frac{}{\mathtt{false} : \mathtt{bool}}\ (\text{T-FALSE})$$

$$\frac{t_1 : \mathtt{bool} \qquad t_2 : \tau \qquad t_3 : \tau}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 : \tau}\ (\text{T-IF}) \qquad \frac{t : \mathtt{int}}{\mathtt{iszero}(t) : \mathtt{bool}}\ (\text{T-ISZERO})$$

Elimination rules of int/bool type：该类型上的表达式可以进行的操作

Introduction rules of int/bool type: 引入该类型的值

interpreter.ml

$\mathcal{L}^{\text{int,bool}}$- semantics

Semantics: judgements, rules

- **Example**

$$\begin{array}{lr}
\quad \mathtt{if}\ \mathtt{iszero}(z)\ \mathtt{then}\ s(z)\ \mathtt{else}\ z & \\
\longmapsto \mathtt{if}\ \mathtt{true}\ \mathtt{then}\ s(z)\ \mathtt{else}\ z & (\text{D-}\mathtt{iszero}_1) \\
\longmapsto s(z) & (\text{D-}\mathtt{if}_2)
\end{array}$$

$$\dfrac{\dfrac{\dfrac{}{z : \mathtt{int}}\,(\text{T-z})}{\mathtt{iszero}(z) : \mathtt{bool}}\,(\text{T-ISZERO}) \qquad \dfrac{\dfrac{}{z : \mathtt{int}}\,(\text{T-z})}{s(z) : \mathtt{int}}\,(\text{T-s}) \qquad \dfrac{}{z : \mathtt{int}}\,(\text{T-z})}{\mathtt{if}\ \mathtt{iszero}(z)\ \mathtt{then}\ s(z)\ \mathtt{else}\ z : \mathtt{int}}\,(\text{T-IF})$$

interpreter.ml

$L^{\text{int,bool}}$- soundness

A language is *safe*/*sound* if satisfying two theorems

- **Progress**

  if $t : \tau$ then either $t\ \mathrm{val}$ or $\exists t'$ such that $t \mapsto t'$

- **Preservation**

  if $t : \tau$ and $t \mapsto t'$ then $t' : \tau$

Prove the theorems by *induction* on the *typing rules*

interpreter.ml

$L^{\rightarrow}$: Typed Lambda Calculus

- ☐ Syntax
- ☐ Semantics
- ☐ Type Safety

λ→: Syntax and Values

λ→: typed lambda calculus

- ☐ **Syntax**

$$\text{Type } \tau ::= \text{int} \mid \tau_1 \to \tau_2$$

$$\text{Term } t ::= n \mid t_1 + t_2 \mid x \mid \lambda(x:\tau).t' \mid t_1\, t_2$$

- ☐ **Values**

$$\frac{}{n \text{ val}}\,(\text{V-n}) \qquad \frac{}{\lambda(x:\tau).t \text{ val}}\,(\text{V-fn})$$

λ→: Syntax

λ→: typed lambda calculus

- ☐ **Syntax**

$$\text{Type } \tau ::= \text{int} \mid \tau_1 \rightarrow \tau_2$$

$$\text{Term } t ::= n \mid t_1 + t_2 \mid x \mid \lambda(x : \tau).t' \mid t_1 t_2$$

$\lambda^{\to}$: Dynamics

Type $\tau ::= \text{int} \mid \tau_1 \to \tau_2$
Term $t ::= n \mid t_1 + t_2 \mid x \mid \lambda(x:\tau).t' \mid t_1 t_2$

☐ Dynamics

- Values

$$\frac{}{n \text{ val}}\text{(V-n)} \qquad \frac{}{\lambda(x:\tau).t \text{ val}}\text{(V-fn)}$$

- Dynamics

$$\frac{t_1 \mapsto t_1'}{t_1 t_2 \mapsto t' t_2}\text{(D-app}_1\text{)} \qquad \frac{}{(\lambda(x:\tau).t_1)t_2 \mapsto [t_2 / x]t_1}\text{(D-app}_2\text{)}$$

$$\frac{t_1 \mapsto t_1'}{t_1 + t_2 \mapsto t_1' + t_2}\text{(D-add}_1\text{)} \qquad \frac{t_1 \text{ val} \quad t_2 \mapsto t_2'}{t_1 + t_2 \mapsto t_1 + t_2'}\text{(D-add}_2\text{)}$$

$$\frac{n_3 = n_1 + n_2}{n_1 + n_2 \mapsto n_3}\text{(D-add}_3\text{)}$$

$\lambda^{\to}$: Statics

$\text{Type } \tau ::= \text{int} \mid \tau_1 \to \tau_2$

$\text{Term } t ::= n \mid t_1 + t_2 \mid x \mid \lambda(x : \tau).t' \mid t_1 t_2$

□ Statics

Γ: typing context

$$\frac{}{n : \text{int}}\ (\text{T-n})$$

$$\frac{\Gamma \vdash t_1 : \text{int} \qquad \Gamma \vdash t_2 : \text{int}}{\Gamma \vdash t_1 + t_2 : \text{int}}\ (\text{T-add})$$

$$\frac{\boxed{\Gamma, x : \tau_1} \vdash t : \tau_2}{\Gamma \vdash \lambda(\boxed{x : \tau_1}).t : \tau_1 \to \tau_2}\ (\text{T-fn})$$

$$\frac{\Gamma \vdash t_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\ t_2 : \tau_2}\ (\text{T-app})$$

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ (\text{T-var})$$

Introduction rule
of function type
(引入该类型的值)

Elimination rule
of function type
(该类型的表达式
上可以进行的操作)

$L^{\times,+,\to}$： Algebraic Data Types

- ☐ Syntax
- ☐ Semantics
- ☐ Type Safety

Algebraic Data Types

- □ Product types
- □ Sum types (variants)

$\text{Type } \tau ::= \cdots \mid \tau_1 \times \tau_2$

$\mid \tau_1 + \tau_2$

$\text{Term } t ::= \cdots \mid (t_1, t_2)$ ordered pair序偶

$\mid t.d$ projection 投影

injection 注入：注入d标记项t得到τ类型值 $\mid \text{inj } t = d \text{ as } \tau$

case分析：如果t是由L标记项得到的，则记L标记项为x_1，按t_1计算；若由R标记项得到，则记R标记项为x_2，按t_2计算 $\mid \text{case } t\ \{x_1 \rightsquigarrow t_1 \mid x_2 \rightsquigarrow t_2\}$

injection 注入

$\text{Direction } d ::= L \mid R$

Product Types: Examples

$$\lambda(x : \text{int} \times \text{int}).x.L + x.R$$

$$\lambda(x : (\text{int} \rightarrow \text{int}) \times \text{int}).x.L\, x.R$$

$$(1, (2, (3, 4))) : \text{int} \times (\text{int} \times (\text{int} \times \text{int}))$$

Product Types: Examples

$$\lambda(x : \mathrm{int} \times \mathrm{int}).x.L + x.R$$

- 函数: 对pair中的两个元素求和

$$\lambda(x : (\mathrm{int} \to \mathrm{int}) \times \mathrm{int}).x.L\ x.R$$

- 函数: 接收由一个函数和一个整数组成的pair, 以该整数为参数调用这个函数

$$(1,(2,(3,4))) : \mathrm{int} \times (\mathrm{int} \times (\mathrm{int} \times \mathrm{int}))$$

- 由pairs产生n元组

Product Types: Semantics

- Statics
  - Introduction rule

$$\frac{\Gamma \vdash t_1 : \tau_1 \qquad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash (t_1, t_2) : \tau_1 \times \tau_2}\text{(T-pair)}$$

  - Elimination rules

$$\frac{\Gamma \vdash t : \tau_1 \times \tau_2}{\Gamma \vdash t.L : \tau_1}\text{(T-project-L)} \qquad \frac{\Gamma \vdash t : \tau_1 \times \tau_2}{\Gamma \vdash t.R : \tau_2}\text{(T-project-R)}$$

Product Types: Semantics

- ☐ Dynamics

$$\frac{t_1 \mapsto t_1'}{(t_1, t_2) \mapsto (t_1', t_2)}\,(\text{D-pair}_1) \qquad \frac{t_1\ \text{val} \qquad t_2 \mapsto t_2'}{(t_1, t_2) \mapsto (t_1, t_2')}\,(\text{D-pair}_2)$$

$$\frac{t_1\ \text{val} \qquad t_2\ \text{val}}{(t_1, t_2)\ \text{val}}\,(\text{D-pair}_3)$$

$$\frac{t \mapsto t'}{t.d \mapsto t'.d}\,(\text{D-proj}_1) \qquad \frac{(t_1, t_2)\ \text{val}}{(t_1, t_2).L \mapsto t_1}\,(\text{D-proj}_2)$$

$$\frac{(t_1, t_2)\ \text{val}}{(t_1, t_2).R \mapsto t_2}\,(\text{D-proj}_3)$$

Record Types

Syntactic Sugar

- ☐ **Syntax**

$$\text{Type } \tau ::= \cdots \mid \{l_1 : \tau_1, \cdots, l_k : \tau_k\}$$
$$\text{Term } t ::= \cdots \mid (l_1 = t_1, \cdots, l_k = t_k)$$
$$\mid t.l_i \quad (1 \le i \le k)$$

- ☐ **Statics**

- ☐ **Dynamics**

Sum Types: Examples

$$(\mathbf{inj}\ \mathbf{l} = \boldsymbol{L}\ \mathbf{as}\ \mathbf{int} + (\mathbf{int} \rightarrow \mathbf{int})) : \mathbf{int} + (\mathbf{int} \rightarrow \mathbf{int})$$

$$\mathbf{case}\ (\mathbf{inj}\ \mathbf{l} = \boldsymbol{L}\ \mathbf{as}\ \mathbf{int} + (\mathbf{int} \rightarrow \mathbf{int}))\{\boldsymbol{x}_1 \rightsquigarrow \boldsymbol{x}_1 + \mathbf{1} \mid \boldsymbol{x}_2 \rightsquigarrow \boldsymbol{x}_2\ \mathbf{2}\} \mapsto \mathbf{l} + \mathbf{1}$$

Sum Types: Examples

$$(\mathrm{inj}\ \mathrm{l} = L\ \mathrm{as}\ \mathrm{int}+(\mathrm{int} \to \mathrm{int})) : \mathrm{int}+(\mathrm{int} \to \mathrm{int})$$

- 将左标记项包装成一个sum类型的值

$$\mathrm{case}\ (\mathrm{inj}\ \mathrm{l} = L\ \mathrm{as}\ \mathrm{int}+(\mathrm{int} \to \mathrm{int}))\{x_1 \rightsquigarrow x_1+1 \mid x_2 \rightsquigarrow x_2\ 2\} \mapsto \mathrm{l}+1$$

- 用case算子对sum值分情况处理，当前是左标记形成的sum值，故按 $x_1 \rightsquigarrow x_1+1$ 计算，得到 $\mathrm{l}+1$

Sum Types: Statics

- Statics
  - Introduction rules

$$\frac{\Gamma \vdash t : \tau_1}{\Gamma \vdash \text{inj } t = L \text{ as } \tau_1 + \tau_2 : \tau_1 + \tau_2}\ (\text{T-inject-L})$$

$$\frac{\Gamma \vdash t : \tau_2}{\Gamma \vdash \text{inj } t = R \text{ as } \tau_1 + \tau_2 : \tau_1 + \tau_2}\ (\text{T-inject-R})$$

  - Elimination rule

$$\frac{\Gamma \vdash t : \tau_1 + \tau_2 \qquad \Gamma, x_1 : \tau_1 \vdash t_1 : \tau \qquad \Gamma, x_2 : \tau_2 \vdash t_2 : \tau}{\Gamma \vdash \text{case } t\ \{x_1 \rightsquigarrow t_1 \mid x_2 \rightsquigarrow t_2\} : \tau}\ (\text{T-case})$$

Sum Types: Dynamics

- Dynamics
  - Inject

$$\frac{t \mapsto t'}{\mathrm{inj}\ t = d\ \mathrm{as}\ \tau \mapsto \mathrm{inj}\ t' = d\ \mathrm{as}\ \tau}\,(\text{D-inject}_1)$$

$$\frac{t\ \mathrm{val}}{\mathrm{inj}\ t = d\ \mathrm{as}\ \tau\ \mathrm{val}}\,(\text{D-inject}_2)$$

  - case

$$\frac{t \mapsto t'}{\mathrm{case}\ t\ \{x_1 \leadsto t_1 \mid x_2 \leadsto t_2\} \mapsto \mathrm{case}\ t'\ \{x_1 \leadsto t_1 \mid x_2 \leadsto t_2\}}\,(\text{D-case}_1)$$

$$\frac{t\ \mathrm{val}}{\mathrm{case}\ \mathrm{inj}\ t = L\ \mathrm{as}\ \tau\ \{x_1 \leadsto t_1 \mid x_2 \leadsto t_2\} \mapsto [t / x_1]t_1}\,(\text{D-case}_2)$$

$$\frac{t\ \mathrm{val}}{\mathrm{case}\ \mathrm{inj}\ t = R\ \mathrm{as}\ \tau\ \{x_1 \leadsto t_1 \mid x_2 \leadsto t_2\} \mapsto [t / x_2]t_2}\,(\text{D-case}_3)$$

Type Algebra

- □ **void and unit types**

$$\text{Type } \tau ::= \cdots \mid \text{void} \mid \text{unit}$$

$$\text{Term } t ::= \cdots \mid \text{null}$$

void是无参的和类型
unit是无参的积类型
null 是unit 类型的唯一值
void类型没有值

- ■ 相等的类型(|τ|表示τ类型值的个数)

$$|\tau \times \text{unit}| = |\tau| \qquad |\tau + \text{void}| = |\tau|$$

- ■ 若τ= bool,有bool x unit 的项(true, null), (false, null)
  - □ 将unit 加入pair ,对数据结构并未增加额外的信息

类型 $\text{bool} + \text{void}$ 有两个值 $\text{inj true} = L \text{ as bool} + \text{void}$

$$\text{inj false} = L \text{ as bool} + \text{void}$$

[The algebra (and calculus!) of algebraic data types]

Some Useful Sum Types

- Enumerate types

例：扑克牌的花色

- 类型　　$\mathbf{card} \triangleq \mathbf{unit}+(\mathbf{unit}+(\mathbf{unit}+\mathbf{unit}))$
- 引入形式（值）：　$\mathbf{hearts} \mid \mathbf{spades} \mid \mathbf{diamonds} \mid \mathbf{clubs}$

$\mathbf{hearts} \triangleq \mathbf{inj}\ \mathbf{null} = L\ \mathbf{as}\ \mathbf{card}$

$\mathbf{spades} \triangleq \mathbf{inj}\ (\mathbf{inj}\ \mathbf{null} = L\ \mathbf{as}\ (\mathbf{unit}+(\mathbf{unit}+\mathbf{unit}))) = R\ \mathbf{as}\ \mathbf{card}$

……

- 消去形式

$\mathbf{case}\ \mathbf{hearts}\ \mathbf{as}\ \mathbf{card}\ \{f_1 \mid f_2 \mid f_3 \mid f_4\} = f_1\ *$

$\mathbf{case}\ \mathbf{spades}\ \mathbf{as}\ \mathbf{card}\ \{f_1 \mid f_2 \mid f_3 \mid f_4\} = f_2\ *$

……

Some Useful Sum Types

Option types

- 类型 $\text{option}_\tau \triangleq \text{unit}+\tau$
- 引入形式（值）： $\text{null} \mid \text{just}(M)$

$$\text{null} \triangleq \text{inj null} = L \text{ as option}_\tau$$

$$\text{just}(M) \triangleq \text{inj } M = R \text{ as option}_\tau, \quad M:\tau$$

- 消去形式

$$\text{ifnull}_\tau : \text{option}_\tau \to (\text{unit} \to \rho) \to (\tau \to \rho) \to \rho$$

$$\text{ifnull}_\tau \ \text{null} \ \{\lambda\_ : \text{unit}.e_1 \mid \lambda x : \tau.e_2\} \mapsto e_1$$

$$\text{ifnull}_\tau \ \text{just}(M) \ \{\lambda\_ : \text{unit}.e_1 \mid \lambda x : \tau.e_2\} \mapsto [M / x]e_2$$

Some Useful Sum Types

理解空指针错误——option类型的意义之一

- **在OO语言中，所有对象都是引用(指针)，对象的引用可能为空，不能通过空引用来访问对象的域类型**
- **如何避免空指针错误?**

  一些语言提供空指针的检测函数 null : $\tau \rightarrow$bool

  if null(e) then …*error* … else …*ok* …
- **但是空指针异常仍然普遍, 原因: 1)缺少空指针检测; 2)极少在程序的异常处进行空指针检测**
- **解决：用option$_\tau$描述类型为τ的可选值类型，其值或者为τ类型的值，或者为空.**

  消去形式 ifnull$_\tau$ e {λ_:unit. …*error*…| λx:τ. …*ok*…}

Reduction Strategies

- □ 操作语义与符号解释器
  - ■ **不确定的解释器**:每一步可选择任意的子表达式进行归约
  - ■ **确定的解释器**：每一步选择一个特定的”下一步归约”
  - ■ **并行的解释器**:把几个归约同时作用于不相交的子表达式
- □ 确定的归约：归约策略
  - ■ **归约策略**：是项到项的部分函数F，它具有性质：如果$F(e)=e'$，那么$e \rightarrow e'$.
  - ■ **求值函数**：

$$eval_F(e_1) = \begin{cases} e_1 & \text{if } F(e_1) \text{ is not defined} \\ e_2 & \text{if } F(e_1) = e_1' \text{ and } eval_F(e_1') = e_2 \end{cases}$$

Reduction Strategies

三种归约策略

例，若$e_1 \to e_1'$且$e_2 \to e_2'$，则$(\lambda x{:}\sigma.e_1)\ e_2$可以归约成

- $[e_2/\ x]\ e_1$：最左最外归约，最左归约，惰性归约. 对于函数应用，称为按名调用

  若e_1中有形参x的多次出现，用e_2代换x会使函数体中有多个e_2，从而e_2的归约会重复多次.

- $(\lambda x{:}\sigma.\ e_1)\ e_2'$：急切归约.对于函数应用，称为按值调用

  若e_2无范式，则$(\lambda x{:}\sigma.\ e_1)\ e_2$的急切归约不会终止.

  若e_2在最终的结果中不使用，则最左归约方式因无需归约e_2而终止.

- $(\lambda x{:}\sigma.e_1')\ e_2$：在提供$e_2$前试图“优化”函数$\lambda x{:}\sigma.e_1$