# STLC with Logic: Poor Man's Future Axi

August 21, 2024

# Grammar

Terms:

$$
\begin{aligned}
e ::= \quad & x \mid \lambda x.e \mid e_1\ e_2 \mid \\
& (e_1, e_2) \mid \texttt{outl}\ e \mid \texttt{outr}\ e \mid \\
& \texttt{inl}\ e \mid \texttt{inr}\ e \mid \texttt{case}\ e\ \texttt{of}\ (x_1.e_1, x_2.e_2) \mid \\
& () \mid \texttt{elim}_{\mathbf{0}}\ e
\end{aligned}
$$

Types:

$$A, B ::= A \to B \mid A \times B \mid A + B \mid \mathbf{1} \mid \mathbf{0}$$

Propositions:

$$
\begin{aligned}
P, Q ::= \quad & \top \mid \bot \mid \neg P \mid P \vee Q \mid P \wedge Q \mid P \Rightarrow Q \mid P \Leftrightarrow Q \mid \\
& \forall x : A.P \mid \exists x : A.P \mid \\
& e_1 =_A e_2
\end{aligned}
$$

# Contexts

Typing contexts:
$\Gamma ::= \cdot \mid \Gamma, x : A$

Assumption contexts:
$\Delta ::= \cdot \mid \Delta, P$

We make use of two kinds of contexts. Typing contexts tell us what the type of a variable is. They are used during typechecking and to see if a proposition is well-formed (since propositions can depend on term variables). Assumption contexts tell us what assumption were made. They are used during proof checking.

# Judgements

Typing judgement:
$\Gamma \vdash e : A$ – in typing context $\Gamma$, term $e$ is of type $A$

Computational equality judgement:
$\Gamma \vdash e_1 \equiv e_2 : A$ – in typing context $\Gamma$, terms $e_1$ and $e_2$ are computationally equal. This is the judgment we use to represent computation.

Well-formed proposition judgement:
$\Gamma \vdash P\ \texttt{prop}$ – in the typing context $\Gamma$, proposition $P$ is well-formed. We need this judgement to prevent forming propositions that contain free variables.

True proposition judgement:
$\Gamma \mid \Delta \vdash P$ – in typing context $\Gamma$ and propositional context $\Delta$, proposition $P$ holds.

# Typing – basics

We treat typing contexts $\Gamma$ as sets, so that there is no need for the so-called structural rules. The basic rule for typing is that variables have whatever type the typing context tells us.

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}$$

# Typing – main rules

$$\frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x.e : A \to B} \qquad \frac{\Gamma \vdash f : A \to B \quad \Gamma \vdash a : A}{\Gamma \vdash f\ a : B}$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \times B} \qquad \frac{\Gamma \vdash e : A \times B}{\Gamma \vdash \mathtt{outl}\ e : A} \qquad \frac{\Gamma \vdash e : A \times B}{\Gamma \vdash \mathtt{outr}\ e : B}$$

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \mathtt{inl}\ e : A + B} \qquad \frac{\Gamma \vdash e : B}{\Gamma \vdash \mathtt{inr}\ e : A + B}$$

$$\frac{\Gamma \vdash e : A + B \quad \Gamma, a : A \vdash e_1 : C \quad \Gamma, b : B \vdash e_2 : C}{\Gamma \vdash \mathtt{case}\ e\ \mathtt{of}\ (a.e_1, b.e_2) : C}$$

$$\frac{}{\Gamma \vdash () : \mathbf{1}} \qquad \frac{\Gamma \vdash e : \mathbf{0}}{\Gamma \vdash \mathtt{elim_0}\ e : A}$$

# Computation – basics

Computational equality is a congruence relation, i.e. an equivalence relation (reflexive, symmetric, transitive) which preserves all term constructors.

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash e \equiv e : A}$$

$$\frac{\Gamma \vdash e_1 \equiv e_2 : A}{\Gamma \vdash e_2 \equiv e_1 : A}$$

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

# Computation – congruence rules

$$\frac{\Gamma, x : A \vdash e \equiv e' : B}{\Gamma \vdash \lambda x.e \equiv \lambda x.e' : A \to B} \qquad \frac{\Gamma \vdash f \equiv f' : A \to B \quad \Gamma \vdash a \equiv a' : A}{\Gamma \vdash f\ a \equiv f'\ a' : B}$$

$$\frac{\Gamma \vdash a \equiv a' : A \quad \Gamma \vdash b \equiv b' : B}{\Gamma \vdash (a, b) \equiv (a', b') : A \times B}$$

$$\frac{\Gamma \vdash e \equiv e' : A \times B}{\Gamma \vdash \mathtt{outl}\ e \equiv \mathtt{outl}\ e' : A} \qquad \frac{\Gamma \vdash e \equiv e' : A \times B}{\Gamma \vdash \mathtt{outr}\ e \equiv \mathtt{outr}\ e' : B}$$

$$\frac{\Gamma \vdash e \equiv e' : A}{\Gamma \vdash \mathtt{inl}\ e \equiv \mathtt{inl}\ e' : A + B} \qquad \frac{\Gamma \vdash e \equiv e' : B}{\Gamma \vdash \mathtt{inr}\ e \equiv \mathtt{inr}\ e' : A + B}$$

$$\frac{\Gamma \vdash e \equiv e' : A + B \quad \Gamma, a : A \vdash e_1 \equiv e_1' : C \quad \Gamma, b : B \vdash e_2 \equiv e_2' : C}{\Gamma \vdash \mathtt{case}\ e\ \mathtt{of}\ (a.e_1, b.e_2) \equiv \mathtt{case}\ e'\ \mathtt{of}\ (a.e_1', b.e_2') : C}$$

$$\frac{}{\Gamma \vdash () \equiv () : \mathbf{1}} \qquad \frac{\Gamma \vdash e \equiv e' : \mathbf{0}}{\Gamma \vdash \mathtt{elim}_{\mathbf{0}}\ e \equiv \mathtt{elim}_{\mathbf{0}}\ e' : A}$$

# Computation – main rules

$$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash a : A}{\Gamma \vdash (\lambda x.b)\ a \equiv b\,[x := a] : B}$$

$$\frac{\Gamma \vdash f : A \to B \quad \Gamma \vdash g : A \to B \quad \Gamma, x : A \vdash f\ x \equiv g\ x : B}{\Gamma \vdash f \equiv g : A \to B}$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \mathtt{outl}\ (a, b) \equiv a : A} \qquad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \mathtt{outr}\ (a, b) \equiv b : B}$$

$$\frac{\Gamma \vdash e_1, e_2 : A \times B \quad \Gamma \vdash \mathtt{outl}\ e_1 \equiv \mathtt{outl}\ e_2 : A \quad \Gamma \vdash \mathtt{outr}\ e_1 \equiv \mathtt{outr}\ e_2}{\Gamma \vdash e_1 \equiv e_2 : A \times B}$$

$$\frac{\Gamma \vdash a : A \quad \Gamma, x : A \vdash e_1 : C \quad \Gamma, y : B \vdash e_2 : C}{\Gamma \vdash \mathtt{case}\ (\mathtt{inl}\ a)\ \mathtt{of}\ (x.e_1, y.e_2) \equiv e_1\,[x := a] : C}$$

$$\frac{\Gamma \vdash b : B \quad \Gamma, x : A \vdash e_1 : C \quad \Gamma, y : B \vdash e_2 : C}{\Gamma \vdash \mathtt{case}\ (\mathtt{inr}\ b)\ \mathtt{of}\ (x.e_1, y.e_2) \equiv e_2\,[y := b] : C}$$

# Computation – unit and empty

The uniqueness rules for unit and empty are a bit unusual: we assert that all terms of these types are computationally equal.

$$\frac{\Gamma \vdash e_1 : \mathbf{1} \quad \Gamma \vdash e_2 : \mathbf{1}}{\Gamma \vdash e_1 \equiv e_2 : \mathbf{1}}$$

$$\frac{\Gamma \vdash e_1 : \mathbf{0} \quad \Gamma \vdash e_2 : \mathbf{0}}{\Gamma \vdash e_1 \equiv e_2 : \mathbf{0}}$$

# Logic – well-formed propositions

$$\frac{}{\Gamma \vdash \top \ \mathtt{prop}} \qquad \frac{}{\Gamma \vdash \bot \ \mathtt{prop}}$$

$$\frac{\Gamma \vdash P \ \mathtt{prop}}{\Gamma \vdash \neg P \ \mathtt{prop}}$$

$$\frac{\Gamma \vdash P \ \mathtt{prop} \quad \Gamma \vdash Q \ \mathtt{prop}}{\Gamma \vdash P \vee Q \ \mathtt{prop}} \qquad \frac{\Gamma \vdash P \ \mathtt{prop} \quad \Gamma \vdash Q \ \mathtt{prop}}{\Gamma \vdash P \wedge Q \ \mathtt{prop}}$$

$$\frac{\Gamma \vdash P \ \mathtt{prop} \quad \Gamma \vdash Q \ \mathtt{prop}}{\Gamma \vdash P \Rightarrow Q \ \mathtt{prop}} \qquad \frac{\Gamma \vdash P \ \mathtt{prop} \quad \Gamma \vdash Q \ \mathtt{prop}}{\Gamma \vdash P \Leftrightarrow Q \ \mathtt{prop}}$$

$$\frac{\Gamma, x : A \vdash P \ \mathtt{prop}}{\Gamma \vdash \forall x : A.P \ \mathtt{prop}} \qquad \frac{\Gamma, x : A \vdash P \ \mathtt{prop}}{\Gamma \vdash \exists x : A.P \ \mathtt{prop}}$$

$$\frac{\Gamma \vdash e_1 : A \quad \Gamma \vdash e_2 : A}{\Gamma \vdash e_1 =_A e_2 \ \mathtt{prop}}$$

# Logic – basics

The truth judgement depends on two contexts – a typing context $\Gamma$ and an assumption context $\Delta$. We treat both of them as sets, which means we don't need any structural rules. We treat negation and equivalence as defined, so that no rules are needed to handle them. We define $\neg P$ to be $P \Rightarrow \bot$ and $P \Leftrightarrow Q$ to be $P \Rightarrow Q \wedge Q \Rightarrow P$. The basic rule of our logic is that we can use assumptions from the assumption context.

$$\frac{P \in \Delta}{\Gamma \mid \Delta \vdash P}$$

# Logic – connectives

$$\frac{\Gamma \mid \Delta, P \vdash Q}{\Gamma \mid \Delta \vdash P \Rightarrow Q} \qquad \frac{\Gamma \mid \Delta \vdash P \Rightarrow Q \quad \Gamma \mid \Delta \vdash P}{\Gamma \mid \Delta \vdash Q}$$

$$\frac{\Gamma \mid \Delta \vdash P \quad \Gamma \mid \Delta \vdash Q}{\Gamma \mid \Delta \vdash P \wedge Q} \qquad \frac{\Gamma \mid \Delta \vdash P \wedge Q}{\Gamma \mid \Delta \vdash P} \qquad \frac{\Gamma \mid \Delta \vdash P \wedge Q}{\Gamma \mid \Delta \vdash Q}$$

$$\frac{\Gamma \mid \Delta \vdash P}{\Gamma \mid \Delta \vdash P \vee Q} \qquad \frac{\Gamma \mid \Delta \vdash Q}{\Gamma \mid \Delta \vdash P \vee Q}$$

$$\frac{\Gamma \mid \Delta \vdash P \vee Q \quad \Gamma \mid \Delta, P \vdash R \quad \Gamma \mid \Delta, Q \vdash R}{\Gamma \mid \Delta \vdash R}$$

$$\frac{}{\Gamma \mid \Delta \vdash \top} \qquad \frac{\Gamma \mid \Delta \vdash \bot}{\Gamma \mid \Delta \vdash P}$$

# Logic – quantifiers

$$\frac{\Gamma, x : A \mid \Delta \vdash P}{\Gamma \mid \Delta \vdash \forall x : A.P} \qquad \frac{\Gamma \mid \Delta \vdash \forall x : A.P \quad \Gamma \vdash a : A}{\Gamma \mid \Delta \vdash P\,[x := a]}$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \mid \Delta \vdash P\,[x := a]}{\Gamma \mid \Delta \vdash \exists x : A.P}$$

$$\frac{\Gamma \mid \Delta \vdash \exists x : A.P \quad \Gamma, x : A \mid \Delta, P \vdash R}{\Gamma \mid \Delta \vdash R}$$

# Logic – equality

Propositional equality is an equivalence relation (note that we handle reflexivity by referring to computational equality) that can be substitued in proofs.

$$\frac{\Gamma \vdash e_1 \equiv e_2 : A}{\Gamma \mid \Delta \vdash e_1 =_A e_2}$$

$$\frac{\Gamma \mid \Delta \vdash e_1 =_A e_2}{\Gamma \mid \Delta \vdash e_2 =_A e_1}$$

$$\frac{\Gamma \mid \Delta \vdash e_1 =_A e_2 \quad \Gamma \mid \Delta \vdash e_2 =_A e_3}{\Gamma \mid \Delta \vdash e_1 =_A e_3}$$

$$\frac{\Gamma \mid \Delta \vdash e_1 =_A e_2 \quad \Gamma, x : A \vdash P \ \mathtt{prop} \quad \Gamma \mid \Delta \vdash P\,[x := e_1]}{\Gamma \mid \Delta \vdash P\,[x := e_2]}$$

# Logic – classical logic

There are many ways to add classical logic to the system, so we pick one at random.

$$\frac{\Gamma \vdash P\ \mathtt{prop} \quad \Gamma \mid \Delta, P \vdash R \quad \Gamma \mid \Delta, \neg P \vdash R}{\Gamma \mid \Delta \vdash R}$$

# Exercises

Just seeing a system like this won't be enough to convince anybody that it makes sense. Therefore, doing some exercises would be advised:

- Assume that $A$ and $B$ are arbitrary types. Define functions $\texttt{swap} : A \times B \to B \times A$ and $\texttt{sweep} : A + B \to B + A$ and prove that they are involutive. Are they computationally involutive, i.e. involutive up to computational equality?
- Can you prove that every term of a product type is a pair?
- Can you prove that every term of a sum type is either `inl` $a$ or `inr` $b$ for some $a$ and $b$?
- Extend the system with a type of booleans.
- Extend the system with a type of natural numbers.
- Write an interesting program and prove an interesting theorem about it.