STLC with Logic: Poor Man's Axi

August 21, 2024

Intro

These slides propose a basic calculus that could serve as a Poor Man's Axi. The system is Simply Typed Lambda Calculus with a first-order logic on top of it that let's us reason about program equality. At the end, we extend the system with booleans and natural numbers.

Grammar

Terms:

$e ::=$

$x \mid \lambda x.e \mid e_1\ e_2 \mid$
$(e_1, e_2) \mid \texttt{outl}\ e \mid \texttt{outr}\ e \mid$
$\texttt{inl}\ e \mid \texttt{inr}\ e \mid \texttt{case}\ e\ \texttt{of}\ (x_1.e_1, x_2.e_2) \mid$
$\texttt{unit} \mid \texttt{elim}_{\mathbf{0}}\ e$

Types:

$A, B ::= A \to B \mid A \times B \mid A + B \mid \texttt{Unit} \mid \texttt{Empty}$

Propositions:

$P, Q ::=$

$\top \mid \bot \mid \neg P \mid P \vee Q \mid P \wedge Q \mid P \Rightarrow Q \mid P \Leftrightarrow Q \mid$
$\forall x : A.P \mid \exists x : A.P \mid$
$e_1 =_A e_2$

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 Γ, term e is of type A

Computational equality judgement:
$\Gamma \vdash e_1 \equiv e_2 : A$ – in typing context Γ, terms e_1 and e_2 are computationally equal. Intuitively, e_1 and e_2 compute to the same normal form.

Well-formed proposition judgement:
$\Gamma \vdash P\ \texttt{prop}$ – in the typing context Γ, proposition P is well-formed.

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

Typing – basics

We treat typing contexts Γ 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

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

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

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

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

$$\frac{}{\Gamma \vdash \mathtt{unit} : \mathtt{Unit}} \qquad \frac{\Gamma \vdash e : \mathtt{Empty}}{\Gamma \vdash \mathtt{elim_0}\ e : A}$$

Computation – basics

We represent computation using the non-directed computational equality relation, which might be a bit unintuitive, but is a common practice.

Computational equality is congruence relation, i.e. an equivalence relation (reflexive, symmetric, transitive) which preserves all term constructors and contains computation rules (this is where computation happens) and uniqueness rules (which tell us when two terms of a given type are equal, but are harder to interpret as computation steps). Note that computational equality is typed, i.e. there's a separate computational equality relation for each type.

Computational equality – equivalence relation

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

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

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

Computational equality – congruence rules

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

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

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

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

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

$$\frac{}{\Gamma \vdash \mathtt{unit} \equiv \mathtt{unit} : \mathtt{Unit}} \qquad \frac{\Gamma \vdash e \equiv e' : \mathtt{Empty}}{\Gamma \vdash \mathtt{elim_0}\ e \equiv \mathtt{elim_0}\ e' : A}$$

Computational equality – computation rules

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

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

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

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

Computational equality – uniqueness rules

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

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

$$\frac{\Gamma \vdash e_1 : \mathtt{Unit} \quad \Gamma \vdash e_2 : \mathtt{Unit}}{\Gamma \vdash e_1 \equiv e_2 : \mathtt{Unit}}$$

$$\frac{\Gamma \vdash e_1 : \mathtt{Empty} \quad \Gamma \vdash e_2 : \mathtt{Empty}}{\Gamma \vdash e_1 \equiv e_2 : \mathtt{Empty}}$$

Logic – well-formed proposition judgement

The role of the well-formed proposition judgement is twofold: to ensure that propositions are well-scoped (i.e. they don't contain free variables), and that propositional equality is formed only from well-typed terms. This judgement depends only on the typing context.

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 – true proposition judgement

The true proposition judgement depends on two contexts – a typing context Γ and an assumption context Δ. 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

The rules for connectives are entirely standard.

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

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

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

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

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

Logic – substitution

To express rules for quantifiers, we need the operation of substituting a term for a variable in a proposition. Our notation is $P\,[x := e]$ for proposition P in which variable x was substituted with term e. To define it, we would of course also need an analogous operation for substituting in terms. Unfortunately, I'm too lazy (and short on time) to define them here, but it shouldn't be hard for you to define it yourself.

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 that can be substitued in proofs. Note that we handle reflexivity by referring to computational equality.

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

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

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

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

Logic – reasoning by cases on terms

Note that so far, we haven't got any rules that allow reasoning by cases on terms. For example, we might want to reason by cases not on a disjunction, but on a term $e : A + B$. To be able to do this, we need to add some more rules. Note that these rules are needed only for positive types (i.e. sums and the empty type), because for negative types the uniqueness rules suffice. Also, there's a slight discrepancy in the presentation between empty and sums, but don't worry about it.

$$\frac{\Gamma \vdash e : \mathtt{Empty}}{\Gamma \mid \Delta \vdash P}$$

$$\frac{\Gamma, a : A \mid \Delta \vdash P\,[x := \mathtt{inl}\ a] \quad \Gamma, b : B \mid \Delta \vdash P\,[x := \mathtt{inr}\ b]}{\Gamma \mid \Delta \vdash \forall x : A + B.P}$$

Logic – classical logic

There are many ways to add classical logic to the system, but we'll have a rule which basically says that we can reason by cases on any proposition.

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

New types – intro

Our current menagerie of type constructors isn't very expressive. In fact, since the only base types are unit and empty, all we can do is finite types and functions between them. Let's what we need to do to add a new type constructor to our language. Note that to save space, we will omit the congruence rules for computational equality.

More types – booleans

$$\frac{}{\Gamma \vdash \texttt{true} : \texttt{Bool}} \qquad \frac{}{\Gamma \vdash \texttt{false} : \texttt{Bool}}$$

$$\frac{\Gamma \vdash e : \texttt{Bool} \quad \Gamma \vdash e_1 : A \quad \Gamma \vdash e_2 : A}{\Gamma \vdash \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 : A}$$

$$\frac{\Gamma \vdash e_1 : A \quad \Gamma \vdash e_2 : A}{\Gamma \vdash \texttt{if true then } e_1 \texttt{ else } e_2 \equiv e_1 : A}$$

$$\frac{\Gamma \vdash e_1 : A \quad \Gamma \vdash e_2 : A}{\Gamma \vdash \texttt{if false then } e_1 \texttt{ else } e_2 \equiv e_2 : A}$$

$$\frac{\Gamma \mid \Delta \vdash P\,[b := \texttt{true}] \quad \Gamma \mid \Delta \vdash P\,[b := \texttt{false}]}{\Gamma \mid \Delta \vdash \forall b : \texttt{Bool}.P}$$

More types – natural numbers

$$\frac{}{\Gamma \vdash \mathtt{zero} : \mathbb{N}} \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathtt{succ}\ n : \mathbb{N}}$$

$$\frac{\Gamma \vdash z : A \quad \Gamma \vdash s : A \to A \quad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathtt{elim}_{\mathbb{N}}\ z\ s\ n : A}$$

$$\frac{\Gamma \vdash z : A \quad \Gamma \vdash s : A \to A}{\Gamma \vdash \mathtt{elim}_{\mathbb{N}}\ z\ s\ \mathtt{zero} \equiv z : A}$$

$$\frac{\Gamma \vdash z : A \quad \Gamma \vdash s : A \to A \quad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathtt{elim}_{\mathbb{N}}\ z\ s\ (\mathtt{succ}\ n) \equiv s\ (\mathtt{elim}_{\mathbb{N}}\ z\ s\ n) : A}$$

$$\frac{\Gamma \mid \Delta \vdash P\,[n := \mathtt{zero}] \quad \Gamma \mid \Delta, P\,[n := n'] \vdash P\,[n := \mathtt{succ}\ n']}{\Gamma \mid \Delta \vdash \forall n : \mathbb{N}.P}$$

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?
- Define addition of natural numbers and prove that it is associative and commutative.
- Write an interesting program and prove an interesting theorem about it.