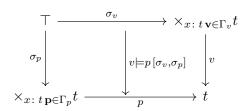
I wanted to try out making a proof assistant corresponding to the double category **Rel**.

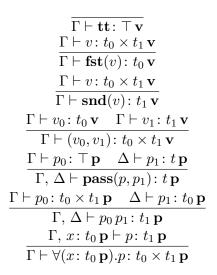
You have a calculus of "values" corresponding to one edge of **Rel** and a sort of relational calculus of "predicates" corresponding to the other edge. Squares ought to correspond to judgements stating a value satisfies a predicate.

The language of "values" handling Cartesian product of sets has product types (in category theory terms is Cartesian.)

The language of "predicates" ought to be more complicated. **Rel** is a closed monoidal category over Cartesian product of sets. One has an isomorphism $\mathbf{Rel}(A, B \otimes C) \sim \mathbf{Rel}(A \otimes B, C)$. So some sort of substructural type theory ought to be the basic framework. I have a hunch explicit unification of logical variables corresponds to explicit duplication of resources but this is a guess.

This is the core framework. I've been thinking about further extensions but I want to see how far I can get characterizing **Rel** with as few language features as possible. Later on I give a few possible extensions.





When a value satisfies a predicate (I need a better symbol here.)

$$\begin{array}{c} v_0 \models p \quad [\sigma] \\ \hline \mathbf{fst}(v_0, v_1) \models p \quad [\sigma] \\ \hline v_1 \models p \quad [\sigma] \\ \hline \mathbf{snd}(v_0, v_1) \models p \quad [\sigma] \\ \hline v \models \mathbf{pass}(p_0, p_1) \quad [\sigma] \\ \hline \mathbf{tt} \models p_0 \ [\sigma] \\ \hline v \models [x \leftarrow p_1] p_0 \quad [\sigma] \\ \hline v \models (\forall (x \colon t \ \mathbf{p}).p_0) \ p_1 \quad [\sigma] \\ \hline v_1 \models p \quad [\sigma, v_0 \models x] \\ \hline (v_0, v_1) \models \forall (x \colon t \ \mathbf{p}).p \quad [\sigma] \end{array}$$

Core Calculus

 $\begin{array}{lll} \textbf{Types} & & & & & & & \\ \textbf{Sorts} & & & & & s ::= t \ \mathbf{p} \ | \ t \ \mathbf{v} \\ \textbf{Values} & & & v ::= x \ | \ \mathbf{tt} \ | \ \mathbf{fst}(v) \ | \ \mathbf{snd}(v) \ | \ (v,v) \\ \textbf{Predicates} & & p ::= x \ | \ \mathbf{pass}(p,p) \ | \ p \ p \ | \ \forall (x : t \ \mathbf{p}).p \\ \textbf{Environment} & & & \Gamma ::= \cdot \ | \ \Gamma, \ x : s \\ \textbf{Substitutions} & & \sigma ::= \cdot \ | \ \sigma, \ v \models x \end{array}$

I need a better name for the abstraction for predicates. It's a little like the μ abstraction from the $\bar{\lambda}\mu\bar{\mu}$ calculus but different. I called it \forall because it's opposite to application/composition in **Rel**which is existential quantification.

The core calculus is based off multiplicative linear type theory.

Typing judgments.

Examples

Pattern matching on equality

$$(v,v) \models \forall (x \colon t \mathbf{p}).x$$

Transposition

$$\forall (p\colon t\times t\times \top)(x\colon t)(y\colon t).p\,y\,x$$

Disjoint Unions

Disjoint unions in set become Cartesian product/coproduct in **Rel**.

I am fairly confident in a simple extension to disjoint unions of sets which are sum types in the value calculus and product types in the predicate calculus.

$$\begin{array}{ll} \textbf{Types} & t ::= \dots \mid \bot \mid t+t \\ \textbf{Values} & v ::= \dots \mid \mathbf{absurd}_t(v) \mid \mathbf{inl}_t(v) \mid \mathbf{inr}_t(v) \mid \\ & \mathbf{match} \ v \ \begin{cases} v & \leftarrow \mathbf{inl}(x) \\ v & \leftarrow \mathbf{inr}(x) \end{cases} \\ \textbf{Predicates} & p ::= \dots \mid \mathbf{false} \mid \mathbf{left}(p) \mid \mathbf{right}(p) \mid \\ & [p;p] \end{array}$$

Typing judgments.

$$\frac{\Gamma \vdash v \colon \bot \mathbf{v}}{\Gamma \vdash \mathbf{absurd}_t(v) \colon t \mathbf{v}} \\ \frac{\Gamma \vdash v \colon t_0 \mathbf{v}}{\Gamma \vdash \mathbf{inl}_{t_1}(v) \colon t_0 + t_1 \mathbf{v}} \\ \frac{\Gamma \vdash v \colon t_1 \mathbf{v}}{\Gamma \vdash \mathbf{inr}_{t_0}(v) \colon t_0 + t_1 \mathbf{v}}$$

$$\frac{\Gamma \vdash v_0 \colon t_0 + t_1 \mathbf{v} \ \Gamma, \ x_0 \colon t_0 \mathbf{v} \vdash v_1 \colon t_2 \mathbf{v} \ \Gamma, \ x_1 \colon t_1 \mathbf{v} \vdash v_2 \colon t_2 \mathbf{v}}{\Gamma \vdash \mathbf{match} \ v_0 \begin{cases} v_1 & \leftarrow \mathbf{inl}(x_0) \\ v_2 & \leftarrow \mathbf{inr}(x_1) \end{cases} \colon t_2 \mathbf{v}} \quad \text{predication of the product of the product$$

$$\begin{array}{c} \overline{\Gamma \vdash \mathbf{false} \colon \bot \, \mathbf{p}} \\ \underline{\Gamma \vdash p \colon t_0 + t_1 \, \mathbf{p}} \\ \overline{\Gamma \vdash \mathbf{left}(p) \colon t_0 \, \mathbf{p}} \\ \underline{\Gamma \vdash p \colon t_0 + t_1 \, \mathbf{p}} \\ \overline{\Gamma \vdash \mathbf{right}(p) \colon t_1 \, \mathbf{p}} \\ \underline{\Gamma \vdash p_0 \colon t_0 \, \mathbf{p} \quad \Gamma \vdash p_1 \colon t_1 \, \mathbf{p}} \\ \underline{\Gamma \vdash [p_0; p_1] \colon t_0 + t_1 \, \mathbf{p}} \end{array}$$

Dependent Sums

If product of sets becomes an internal hom in the predicate calculus then dependent sums ought to become a little like Π types. So the predicate calculus effectively becomes like System-F.

Some things become awkward to interpret here though.

I also really can't figure out unpacking. messy if you don't want full dependent types.

$$\begin{array}{lll} \text{Types} & t ::= \ldots \mid x \mid \mathbf{head}(v) \mid \Sigma(x \colon \ ^*).t \\ \text{Sorts} & s ::= \ldots \mid \ ^* \\ \text{Values} & v ::= \ldots \mid \mathbf{tail}(v) \mid \langle x \leftarrow t, v \rangle \\ \text{Predicates} & p ::= \ldots \mid pt \mid \mathbf{M}(x \colon \ ^*).p \\ \text{Substitutions} & \sigma ::= \ldots \mid \sigma, t \models x \end{array}$$

Not really good at the typing judgements for dependent sum types.

Satisfies

$$\frac{\Gamma \vdash v \colon \Sigma(x \colon \ ^*).t\,\mathbf{v}}{\Gamma \vdash \mathbf{head}(v) \colon \ ^*}$$

$$\frac{\Gamma \vdash v \colon \Sigma(x \colon \ ^*).t\,\mathbf{v}}{\Gamma \vdash \mathbf{tail}(v) \colon [x \leftarrow \mathbf{head}(v)]t\,\mathbf{v}}$$

$$\frac{\Gamma \vdash t_0 \colon \ ^* \quad \Gamma,\, x \colon \ ^* \vdash v \colon t_1\,\mathbf{v}}{\Gamma \vdash \langle x \leftarrow t_0, v \rangle \colon \Sigma(x \colon \ ^*).t_0\,\mathbf{v}}$$

$$\frac{\Gamma \vdash p \colon \Sigma(x \colon \ ^*).t_1\,\mathbf{p} \quad \Gamma \vdash t_0 \colon \ ^*}{\Gamma \vdash p_0\,t_0 \colon [x \leftarrow t_0]t_1\,\mathbf{p}}$$

$$\frac{\Gamma,\, x \colon \ ^* \vdash p \colon t\,\mathbf{p}}{\Gamma \vdash \mathbf{M}(x \colon \ ^*).p \colon \Sigma(x \colon \ ^*).t\,\mathbf{p}}$$

I can't figure out satisfaction at all.

$$\begin{aligned} t &\models p \quad [\sigma] \\ \hline \mathbf{head}(\langle x \leftarrow t, v \rangle) &\models p \quad [\sigma] \\ \hline (x \leftarrow t]v &\models p \quad [\sigma] \\ \hline \mathbf{tail}(\langle x \leftarrow t, v \rangle) &\models p \quad [\sigma] \\ \hline v &\models [x \leftarrow t]p \quad [\sigma] \\ \hline v &\models (\mathbf{M}(x \colon *).p) t \quad [\sigma] \\ \hline v &\models p \ [\sigma, t \models x] \\ \hline \langle x \leftarrow t, v \rangle &\models \mathbf{M}(x \colon *).p \quad [\sigma] \end{aligned}$$

The Future?

Satisifies judgments correspond to thin squares. Moving to more generally categories such as **Span** or **Prof** or **Vect** for matrix math requires an interpretation of squares carrying constructive content.