I wanted to try out making a programming language vaguely 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. Satisfies judgements stating a value satisfies a predicate ought to correspond to squares.

The language of "predicates" is based off the Simply Typed Lambda Calculus (in category theory terms is closed/has exponential objects) and the language of "values" has product types (in category theory terms is Cartesian.)

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.

Core Calculus

Types $t ::= 1 \mid t \times t$

Sorts $s ::= t? \mid t!$

Values $v ::= x \mid \pi_1 v \mid \pi_2 v \mid (v, v)$

Predicates $p ::= x \mid p\, p \mid \mu(x : t?).p$

Environment $\Gamma ::= \cdot \mid x : s, \Gamma$

Substitutions $\sigma ::= \cdot \mid v \models x, \sigma$

I need a better name for the abstraction for predicates. It's a little like the μ abstraction $\bar{\lambda}\mu\tilde{\mu}$ but different.

Typing judgments.

$$\frac{}{\Gamma \vdash \mathbf{tt} : 1!}$$

$$\frac{\Gamma \vdash v : t_0 \times t_1!}{\Gamma \vdash \pi_1 v : t_0!}$$

$$\frac{\Gamma \vdash v : t_0 \times t_1!}{\Gamma \vdash \pi_2 v : t_1!}$$

$$\frac{\Gamma \vdash v_0 : t_0! \quad \Gamma \vdash v_1 : t_1!}{\Gamma \vdash (v_0, v_1) : t_0 \times t_1!}$$

$$\frac{\Gamma \vdash p_0 : t_0 \times t_1? \quad \Gamma \vdash p_1 : t_0?}{\Gamma \vdash p_0\, p_1 : t_1?}$$

$$\frac{x : t_0?, \Gamma \vdash p : t_1?}{\Gamma \vdash \mu(x : t_0?).p : t_0 \times t_1?}$$

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

$$\frac{v_0 \models p\ [\sigma]}{\pi_1(v_0, v_1) \models p\ [\sigma]}$$

$$\frac{v_1 \models p\ [\sigma]}{\pi_2(v_0, v_1) \models p\ [\sigma]}$$

$$\frac{v \models [x := p_1]p_0\ [\sigma]}{v \models (\mu(x : t?).p_0)\, p_1\ [\sigma]}$$

$$\frac{v_1 \models p\ [v_0 \models x,\, \sigma]}{(v_0, v_1) \models \mu(x : t?).p\ [\sigma]}$$

Example

$$(v, v) \models \mu(x : t?).x\ [\sigma]$$

Disjoint Unions

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.

Types $t ::= \ldots \mid 0 \mid t + t$

Values $v ::= \ldots \mid i_1\, t\, v \mid i_2\, t\, v \mid \mathbf{match}\, v\, \mathbf{with}\, |i_1 x \Rightarrow v\ |i_2 x \Rightarrow v\ \mathbf{end}$

Predicates $p ::= \ldots \mid \mathbf{false} \mid \mathbf{left}\, p \mid \mathbf{right}\, p \mid [p; p]$

Typing judgments.

$$\frac{\Gamma \vdash v : t_0!}{\Gamma \vdash i_1\, t_1\, v : t_0 + t_1!}$$

$$\frac{\Gamma \vdash v : t_1!}{\Gamma \vdash i_2\, t_0\, v : t_0 + t_1!}$$

$$\frac{\Gamma \vdash v_0 : t_0 + t_1! \quad x_0 : t_0!, \Gamma \vdash v_1 : t_2! \quad x_1 : t_1!, \Gamma \vdash v_2 : t_2!}{\Gamma \vdash \mathbf{match}\, v_0\, \mathbf{with}\, |i_1 x_0 \Rightarrow v_1\ |i_2 x_1 \Rightarrow v_2\ \mathbf{end} : t_2!}$$

$$\frac{}{\Gamma \vdash \mathbf{false} : 0?}$$

$$\frac{\Gamma \vdash p : t_0 + t_1?}{\Gamma \vdash \mathbf{left}\, p : t_0?}$$

$$\frac{\Gamma \vdash p : t_0 + t_1?}{\Gamma \vdash \mathbf{right}\, p : t_1?}$$

$$\frac{\Gamma \vdash p_0 : t_0? \quad \Gamma \vdash p_1 : t_1?}{\Gamma \vdash [p_0; p_1] : t_0 + t_1?}$$

Satisfies

$$\frac{[x_0 := v_0]v_1 \models p\ [\sigma]}{\mathbf{match}\, i_1\, t\, v_0\, \mathbf{with}\, | i_1 x_0 \Rightarrow v_1\ | i_2 x_1 \Rightarrow v_2\ \mathbf{end} \models p\ [\sigma]}$$

$$\frac{[x_1 := v_0]v_2 \models p\ [\sigma]}{\mathbf{match}\, i_2\, t\, v_0\, \mathbf{with}\, | i_1 x_0 \Rightarrow v_1\ | i_2 x_1 \Rightarrow v_2\ \mathbf{end} \models p\ [\sigma]}$$

$$\frac{v \models p_0\ [\sigma]}{v \models \mathbf{left}\,[p_0; p_1]\ [\sigma]}$$

$$\frac{v \models p_1\ [\sigma]}{v \models \mathbf{right}\,[p_0; p_1]\ [\sigma]}$$

$$\frac{v \models p_0\ [\sigma]}{i_1\, t\, v \models [p_0; p_1]\ [\sigma]}$$

$$\frac{v \models p_1\ [\sigma]}{i_2\, t\, v \models [p_0; p_1]\ [\sigma]}$$

Dependent Sums

If product of sets becomes an exponential in the predicate calculus then dependent sums ought to become 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. It's messy if you don't want full dependent types.

Types $t ::= \ldots \mid x \mid \Sigma(x : \mathbf{type}).t$

Sorts $s ::= \ldots \mid \mathbf{type}$

Values $v ::= \ldots \mid \langle x := t, v\rangle$

Predicates $p ::= \ldots \mid p\, t \mid \mathbf{M}(x : \mathbf{type}).p$

Substitutions $\sigma ::= \ldots \mid t \models x,\ \sigma$

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

$$\frac{\Gamma \vdash t_0 : \mathbf{type} \quad x : \mathbf{type}, \Gamma \vdash v : t_1!}{\Gamma \vdash \langle x := t_0, v\rangle : \Sigma(x : \mathbf{type}).t_0!}$$

$$\frac{\Gamma \vdash p : \Sigma(x : \mathbf{type}).t_1? \quad \Gamma \vdash t_0 : \mathbf{type}}{\Gamma \vdash p_0\, t_0 : [x := t_0]t_1?}$$

$$\frac{x : \mathbf{type}, \Gamma \vdash p : t?}{\Gamma \vdash \mathbf{M}(x : \mathbf{type}).p : \Sigma(x : \mathbf{type}).t?}$$

I can't figure out satisfaction at all.

$$\frac{v \models [x := t]p\ [\sigma]}{v \models (\mathbf{M}(x : \mathbf{type}).p)\, t\ [\sigma]}$$

$$\frac{v \models p\ [t \models x,\ \sigma]}{\langle x := t, v\rangle \models \mathbf{M}(x : \mathbf{type}).p\ [\sigma]}$$